12-2018

A Comparative Study of Project Management System Web Applications Built on ASP.Net Core and Laravel MVC Frameworks

Alon Poudel
alonpoudel@gmail.com

Recommended Citation

Poudel, Alon, "A Comparative Study of Project Management System Web Applications Built on ASP.Net Core and Laravel MVC Frameworks" (2018). *Culminating Projects in Computer Science and Information Technology*. 27.
https://repository.stcloudstate.edu/csit_etds/27

A Comparative Study of Project Management System Web Applications Built on ASP.Net Core and Laravel MVC Frameworks

by

Alon Poudel

A Starred Paper

Submitted to the Graduate faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Computer Science

January, 2019

Starred Paper Committee:
Bryant A. Julstrom
Jie H. Meichsner
Channa J. Kumarage

Abstract

With rapid advancement in the field of computer science, the ways we use and interact with web applications have changed immensely. Developers must create web applications for browsers, cell phones, and search engines that are accessible and easy to use in various devices. Therefore, the efficiency of software development is critical. Software Design Patterns are an essential part of software development which is intended to solve real-world problems by creating templates of best practices. Design patterns bring clarity, cost-effectiveness, and better communication in the software development cycle. They also improve the development speed, support features, and usage, and they reduce expenses. Documentation and maintenance of established web applications frameworks are major advantages of software design patterns.

The study Is of Model-View-Controller (MVC) software design patterns. It analyzes and compares ASP.Net Core and Laravel PHP web application development frameworks. MVC facilitates reuse of code and separation of application layers. It explains the development experience of Project Management Web application on ASP.Net Core and Laravel. For example, web applications include a document library, a note page, and a discussion forum. Web applications use compatible programming languages such as HTML, JavaScript, and CSS. Comparative analysis has been done based on the developer's experience and performance monitoring tools.

The study concludes that the Project Management System (PMS) web application built using ASP.Net Core on Windows is better when compared to PMS built with Laravel on Ubuntu and Windows operating systems. The developer's conclusion is based on the use of the MVC design pattern, learning curve, framework features, documentation, and application performance.

Acknowledgement

I would like to sincerely thank my advisor Dr. Bryant A. Julstrom for providing support and feedback throughout the course of this study. I would also like to thank Dr. Jie H. Meichsner for her support and advise. And, I would like to thank Mr. Channa J. Kumarage for providing motivation to complete this study.

A big thank you to my family and friends.

Table of Contents

Page

List of Table

Table Page

List of Figures

Figure Page

Chapter I: Introduction

Software design patterns are reusable solutions to software design problems that recur in application development (Bisson, 2017). They are templates of formalized best practices in designing real-world applications and system.

Developing software applications is complex and costly. Developers need to focus on the behavior and environment of each platform, technology, and architecture. *Software Design Patterns* help identify potential problems before implementation and reduce code complexity and code clutter.

There are various web application frameworks following many software design patterns. Among them, the developer studies the *Model-View-Controller (MVC)* pattern. The developer presents the analysis, design, and implementation of software applications based on MVC. *MVC* offers the possibility of code reuse and provides strict separation of the application layers. *MVC* gives more control to developers and helps to build lightweight web applications. The study examines the use of the *MVC* design pattern in two web development frameworks: *ASP.Net Core*

Important Definitions

Following are relevant terms defined within the scope of this comparative study:

AdminLTE**:** Open source dashboard & control panel theme that can be used as a template for the front end of the web application.

Apache HTTP server**:** A free and open-source cross-platform web server.

ASP.Net core**:** *A* web application development framework developed by Microsoft that integrates *Model-View-Controller (MVC)* architecture and *ASP.Net* platform.

ASP.Net**:** A scripting language developed by Microsoft for developing web applications and web pages.

Blade**:** A template which is used to apply standard settings to views for the development of the web pages in *Laravel* framework (Bautista, 2012).

Bootstrap**:** An open-source and an efficient front-end framework used for web application development. It includes *CSS*, *HTML*, and *JavaScript* in its framework.

C#**:** *C#* (Pronounced: C sharp) is an Object-Oriented programming language developed by Microsoft and used for the development of applications on the *ASP.NET* framework.

Cascading Style Sheets (CSS)**:** *CSS* is the standard markup language used for presenting the styles of the content in the web pages.

Client-side programming**:** Also known as front-end programming, includes everything that a user sees and interacts with on a web browser.

Command Line Interface (CLI)**:** *CLI* is used to interact with the web application by the users. Example: MS DOS and Apple DOS

Composer**:** A package dependency resolver for the *Laravel* web application framework. It is used to manage dependency in *PHP*.

Cross-Site Reference Forgery (CSRF)**:** *CSRF* is an attack forcing unwanted actions from the users without their consent.

CSS**/*****JQuery Extractor*****:** It is used to extract values from *JMeter* (a software testing tool).

Database**:** It is a set of data which is stored in a structured way. The data set can be read/updated/deleted.

Development environment: The developer develops software in the development environment.

dontnetcorepms.interfaces: The developer created the main project file named *dontnetcorepms.interfaces*. It is a class library which consists of all the interfaces needed by controllers to interact with the repositories.

dontnetcorepms.models**:** The developer created the main project file named *dontnetcorepms.models***.** It consists of all the models required for the project.

dontnetcorepms.repositories**:** The developer created the main project file named *dontnetcorepms.repositories*. The *dontnetcorepms.interfaces* includes all the interface files in the repositories of the project.

dontnetcorepms**:** The developer created the main project file named *dotnetcorepms* for PMS built with *ASP.Net core*.

Eloquent Framework**:** *Laravel* 5.5 uses *Eloquent* as an *Object Relational Mapper (ORM)* to allow conversion of data between incompatible systems (between models and databases). It is inbuilt within the *Laravel* framework.

Entity Framework**:** The *ASP.Net Core* uses *Entity Framework* as an *Object Relational Mapper (ORM)* that allows the conversion of data between incompatible systems (between models and databases).

Github**:** A hosting service used for version control of the software under development.

Graphical User Interface (GUI)**:** GUIs are used by the users to interact with the web application using graphical icons instead of commands in command line.

HTACCESS file**:** It is the configuration file for *Laravel* web application framework.

HTTP**:** *Hypertext Transfer Protocol* (*HTTP*) is a request-response protocol which is used to communicate between clients and servers.

HTTP Cookie Manager**:** It manages cookies containing data on user sessions and login information.

Hypertext Markup Language (HTML)**:** *HTML* is a standard markup language to create web pages.

Integrated Development Environment (IDE)**:** Software development application with necessary tools and development environment.

JavaScript (JS)**:** *JS* is a high-level programming language to create interactive web pages.

JMeter**:** *JMeter* is the tool to measure the application performance of the Project Management System web application.

JQuery**:** *JQuery* is a *JavaScript (JS)* library that is used to simplify the use of *JavaScript* programming language.

Laravel**:** A simple, elegant and well-documented *PHP* web application development framework (Bautista, 2012).

Microsoft Visual Studio Entity Data Model (EDMX)**:** It is a file extension *(.EDMX)* that establishes relationship between the interface file in the web application framework and the database.

Model-View-Controller (MVC)**:** *MVC* is one of the software design patterns. The View consists of the user interface and user experience. The model delivers data to the View. The

Controller takes the user's request and loads the appropriate Model and View (Helm, Johnson, Gamma, & Vlissides, 2000).

MySQL**:** An open-source database management system that stores data in multiple tables instead of using one table to store all data.

Nginx**:** An open-source web server.

Node.JS**:** It is a server environment which can be used across software platforms.

Object Relational Mapper (ORM)**:** A tool or technique that allows conversion of data between incompatible type systems using an object-oriented paradigm.

Open-source**:** Open source software can be used, modified or shared for free.

Page Latency**:** It is the page load time when establishing a connection between the web server and the browser, and vice versa.

PHP**:** Hypertext Preprocessor (*PHP*) is a language used for server-side programming for the development of web applications and web pages.

PhpMyAdmin**:** A graphical interface for the *MySQL* database server.

Production environment: After the development and testing of the software, software is deployed and executed in the *Production environment*. Software is made available to the customers in this environment.

Razor**:** A template which is used to apply standard settings to views for the development of the web pages in the *ASP.Net* framework.

Server-side programming**:** Also known as back-end programming, deals with the logic and functioning of web applications.

***Software Design Patterns*:** Software templates of formalized best practices in the world of web application development.

***SQL*:** Structured Query Language (*SQL*) is *RDBMS* that is used to access databases.

***SQL Management Studio*:** The *MSSQL Server* has *SQL Management Studio* as the graphical interface to manage the database.

***Sublime Text Editor*:** It is an *IDE* used for the development of Project Management System with the *Laravel* framework.

***The developer*:** For this study, a person creating Project Management System web application with *ASP.Net Core* and *Laravel* frameworks, and compiling reports based on the experience and literature within the scope of this study.

***Ubuntu*:** An open-source Linux operating system.

***Visual Studio*:** It is an *IDE* used for the development of Project Management System with the *ASP.Net Core* framework.

***Web development framework*:** It is the implementation of software design pattern that constitutes a set of formalized templates of software features, libraries and best practices used for the development of web applications.

***XAMPP*:** It is a free and open-source web server that can be used across software platforms.

Problem Statement

There are many software design patterns and web application frameworks in the market. It is essential for developers to know the most suitable software design pattern and the web application development framework based on that design pattern. Appropriate framework provides development speed, support features, proper usage, and less expenses. Therefore, as a developer, a good understanding of framework is vital for web development process.

Proposed Solution

The study proposes *MVC* as a suitable software design pattern for web application development. *ASP.Net Core* and *Laravel* are two popular web application frameworks based on *MVC* design pattern. Therefore, *ASP.Net Core* and *Laravel* are studied comparatively.

Both frameworks are open source, popular, and have vast communities. They are great to build Project Management System (PMS). The PMS created contains user profile, roles, document library, notes, and a discussion forum.

Description of a Web Application Development Framework

A web application development framework comprises front-end and back-end technologies. Front-end interaction with applications has changed from *CLI* to *GUI*. Most web applications use *HTML*, *CSS*, and *JS* to build a rich user experience. Also, security of web applications has become an essential requirement for web applications. Due to the availability of numerous frameworks, it is essential to use the most suitable one based on application performance, learning curve, maintenance, and user experience.

In addition, the developer needs to understand the terms and conditions of open source licenses before making use of these frameworks, to avoid any legal consequences. Quality, security, flexibility, support options, security, and open-source license are attractive characteristics.

The performance of web application depends on features such as bandwidth, the number of user requests on the network, application protocols, the application hosting environment, and the programming tools and languages used to develop the application.

Chapter II: Literature Review

Design patterns are reusable solutions to software design problems (Bisson, 2017). Design patterns reuse successful software architectures and provide design alternatives to make a system reusable. They also help document and maintain existing systems (Jailia, Kumar, Agarwal, & Sinha, 2016).

Developing generic software applications built on the dynamic web is complex and costly because developers need to understand many platforms, technologies, and architecture. *MVC* offers the possibility of code reuse and provides separation between the application layers. Platforms, technologies, and dynamic web architectures generate significant cost on a technical level. Developers are required to produce high-quality dynamic web applications within a short time (Komara, Hendradjaya, & Saptawati, 2016).

A suitable web application framework reduces much burden from the developer side. Given the popularity of the *MVC* design pattern, and *ASP.Net Core* and *Laravel* frameworks, there is a broad consensus over the use of web application frameworks and software design patterns.

ASP.Net Core Framework

ASP.Net Core is a web application development framework that integrates *MVC* architecture and *ASP.Net* scripting language. Microsoft introduced *ASP.Net* for developing web applications and web pages. The framework provides users with clean client-side code that downloads and renders quickly in the browser. It is beneficial for web applications that can be part of the cloud environment. Also, it gives stability and flexibility of rapid application development (Helm et al., 2000).

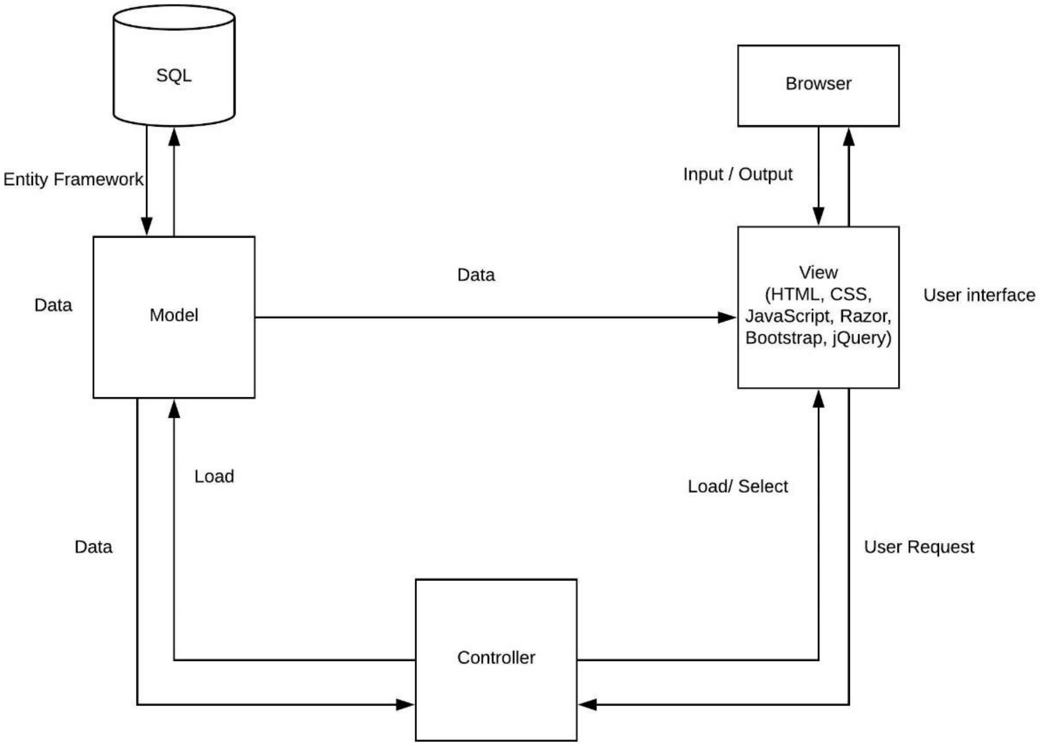


Figure 1. ASP.Net Core Architecture Diagram

As shown in Figure 1, *ASP.Net Core* follows *MVC* (Model-View-Controller) design pattern. The View is used for the user interface and user experience. The Model delivers data to the View. The Controller takes the user's request and loads the appropriate Model and View. First, the controller gets the input, and then it goes to the View. There can be many views but only one controller. The controller selects views to be rendered. Because of this clear distinction, *ASP.Net Core* helps manage large-scale projects by working on individual components (Helm et al., 2000).

The *ASP.Net Core* framework is useful when there are issues of time, security, and complexity. *ASP.Net Core* can be viewed from front-end and back-end perspectives. Front-end comprises *Razor* template, *JQuery*, and *Bootstrap* for an interactive and dynamic page. The backend is composed of controllers, *Entity Framework*, and databases.

Laravel Framework

The *Laravel* framework relies on Hypertext Preprocessor (*PHP*) as its scripting (programming) language. *Laravel* has an expressive syntax that provide solutions for development and facilitate general task in big web projects (Alfat, Triwiyatno, & Isnanto, 2015). *Laravel* is a clear, simple, elegant, and well-documented framework that focuses on equipping and enabling developers. It helps developers learn, start, and develop quickly (Stauffer, 2016). The *Laravel* framework is useful when there are issues of time, security, and complexity.

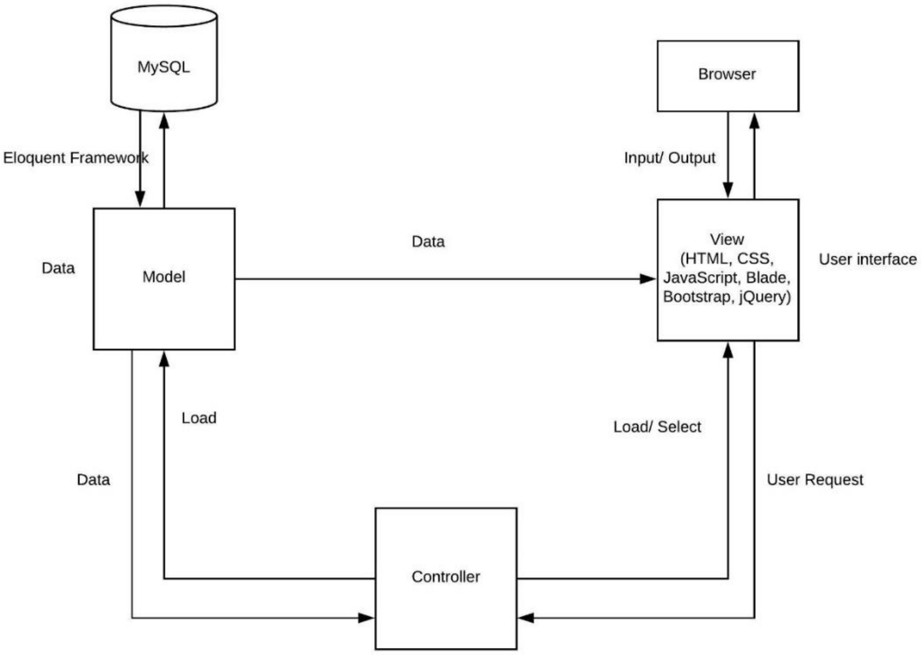


Figure 2. Laravel Architecture Diagram

Laravel follows the same architecture as the *ASP.Net Core* framework, as shown in Figures 1 and 2. Therefore, the interactions among Model, View, and Controller are the same as in the *ASP.Net Core* framework. The difference between *ASP.Net Core* and *Laravel* is the technologies used in achieving that *MVC* design pattern.

ASP.Net Core can be viewed from front-end and back-end perspectives. The front-end comprises a *Blade* template, *JQuery*, and *Bootstrap* for an interactive and dynamic page. The backend is composed of controllers, *Eloquent Framework*, and databases.

Project Management System (PMS)

PMS is a system where users are added in roles, such as: Administrator, User, etc. The user with Administrator role can add other users to the PMS for access. However, the Users with privileges other than Administrator are not able to add other users. Once logged in to the PMS, users can add, update or delete items from the notes, document library and forum sections. This system allows users to collaborate on projects or tasks assigned to the users.

System Requirements

Table 1 shows the software components needed to develop and deploy Project Management System web application in the *ASP.Net Core* framework and in the *Laravel* framework. The list of components is also used for comparing results. The columns list the components used to build the web application.

Table 1

Requirements of PMS

Components	*ASP.Net Core* (Dev)	*Laravel* 5.5 (Dev)	*Laravel* 5.5 (Deploy)
Environment	*Windows 10 (Physical)*	*Windows 10 (Physical)*	*Ubuntu* 17.10 (Virtual)
Object Relational Mapper (ORM)	*Entity Framework*	*Eloquent*	*Eloquent*
Template Engine	*Razor*	*Blade*	*Blade*
License	MIT License	MIT License	MIT License
Support and Security	Developed and maintained by Microsoft and the *ASP.Net* community on *Github*	Developed and maintained by Taylor Otwell and the *Laravel* community on *Github*	Developed and maintained by Taylor Otwell and the *Laravel* community on *Github*
User Interface / User Experience Design	*AdminLTE* – An open source admin dashboard	*AdminLTE* – An open source admin dashboard	*AdminLTE* – An open source admin dashboard
Programming Language	*C#*	*PHP*	*PHP*
Integrated Development Environment (IDE)	*Visual Studio 2017 Community*	*Sublime Text 3*	*Sublime Text 3*
Database Server	*Microsoft SQL Server 2014*	*MySQL*	*MySQL*
Community	Large Community	Growing Community	Growing Community

Microsoft *Windows 10* is required to develop web application based on *ASP.Net Core* 2.0. The *RDBMS* for the web application will be Microsoft *SQL* Server 2014. The web server is inbuilt for *ASP.Net Core* 2.0. *Visual Studio Community 2017* will be the *IDE* to build this web application.

The *Ubuntu* 17.10 operating system is required to develop web applications based on *Laravel* which are written in the *PHP* programming. *MySQL* is the database system. The web servers are *Nginx* (for deploy) and Apache (for dev), and Sublime Text 3 is the *IDE* used for the development of the web application.

Chapter III: Design

The design section comprises the description of the *MVC* design pattern and architecture of a PMS built with both *ASP.Net Core* and *Laravel*.

Architecture of PMS

The high-level flow of information in the PMS is shown in Figure 3 and 4.

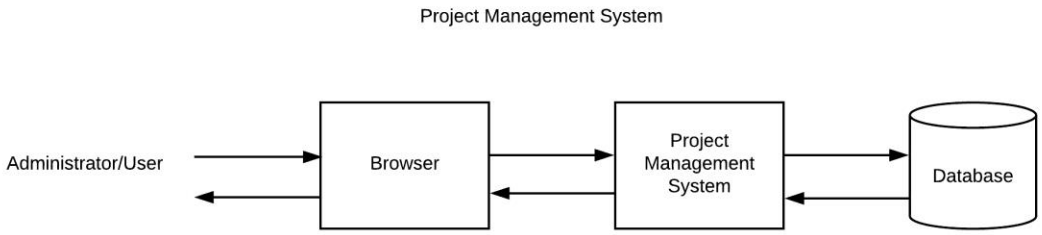


Figure 3. Flow of PMS

Figure 3 shows the implementation of the PMS with both *ASP.Net Core* and *Laravel* frameworks. Administrator and Users log into the PMS with their usernames and passwords. They have access according to their roles assigned by the Administrator. For example: Administrator and User. Only Administrator has full control of the PMS and can provide or revoke access of all other users.

Figure 4 shows the detailed architecture of the PMS built on *ASP.Net Core* and *Laravel* frameworks. The web application has insert, update, and delete functionalities. Performance evaluation of the application includes page load time, page latency, bytes sent to the server, and bytes received from the server. The evaluation is performed using application performance tools such as *JMeter*, *debugbar* and the *IDE*s.

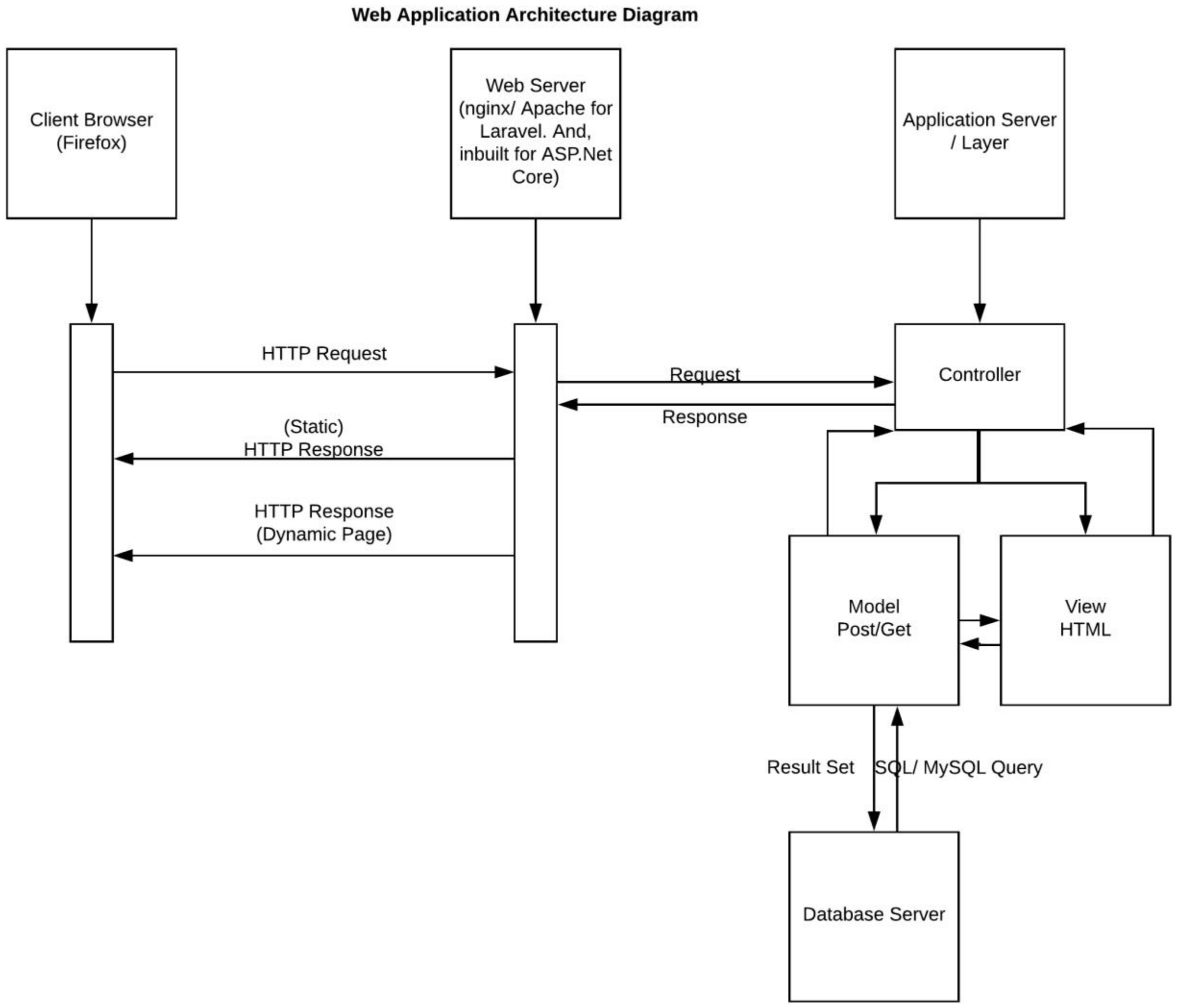

Web Application Architecture Diagram
Client Browser (Firefox)
Web Server (nginx/ Apache for Laravel. And, inbuilt for ASP.Net Core)
Application Server / Layer
HTTP Request
(Static) HTTP Response
HTTP Response (Dynamic Page)
Request
Response
Controller
Model Post/Get
View HTML
Result Set
SQL/ MySQL Query
Database Server

Figure 4. Architecture Diagram of PMS

As shown in Figure 4, the client browser sends an *HTTP* (*GET*/*POST*) request to the web server. The web server accepts the *HTTP* request and forwards the request to the controller. The controller determines whether it needs to interact with the model or not. If interaction with the model or the database is not needed, it returns a view, composed of *HTML* markup code as *HTTP*, which is a static page. If the controller determines that interaction with the model be needed, it interacts with the model, and then the model interacts with the database server using the *SQL* queries generated by the controller. The data set is generated by the database server, which is interpreted by the controller. Then, the controller responds with a web page.

Chapter IV: Implementation

This section explains the execution of procedures and requirements to develop the PMS with *ASP.Net Core* on *Windows*, and PMS with *Laravel* on *Windows* and *Ubuntu*.

Implementation of PMS with ASP.Net Core framework

Before starting the development of a PMS in *ASP.Net Core* 2.0, the developer had to set up the development environment as well as plan the development of the application. The setup for the software development involved installation of *Visual Studio 2017 Community,* which included required software packages. Then the developer installed Microsoft *SQL* Server 2014 and *ASP.Net Core* 2.0 in the *Windows 10* operating system.

After the setup, coding of the application started. This involved adding new controllers, models, views, and resources to the *ASP.Net Core* framework. A new *ASP.Net Core* web application project was created using the *Visual Studio 2017 Community* edition *IDE*. The project was created using the inbuilt authentication feature called *ASP.Net Single identity* at first, but due to its lack of flexibility in the development process, the developer had to remove the feature and build a custom authentication module. The custom authentication module required a login controller where user logins were validated, and if valid, login sessions were created in the web browser.

The user interface framework named *AdminLTE*, was used for all the PMS web applications. After the user interface was integrated into the application, the developer made changes in the login page and the dashboard, removed unwanted files and integrated *JS* and *CSS* plugins.

A coding standard was set up for all the available modules. The standard was to create four different projects under the same solution in *Visual Studio*. The four projects were- *dontnetcorepms* (main project), *dontnetcorepms.interfaces*, *dontnetcorepms.models*, and *dontnetcorepms.repositories*. They were separated based on their functionalities. The *dontnetcorepms* project included the main web application contents and integrated the other three projects. The *dontnetcorepms.interfaces* is a class library which consists of all the interfaces needed by controllers to interact with repositories. As the name suggests, the *dontnetcorepms.interfaces* include all the interfaces which are implemented in the repositories project. The *dontnetcorepms.repositories* is a class library which consists of implementations of the interfaces. The *dontnetcorepms.models* consists of all the models required for the project. The decomposition of the entire application based on their functionalities helps to debug and troubleshoot the application.

Implementation of PMS with Laravel Framework in Windows

The developer set up the development environment and planned the development of the application. *Laravel* 5.5 was downloaded, installed and configured on the development machine using *Composer*. The setup for the web application development involved installation of *XAMPP 7.0.27* which included *PHP 7.0.27*, *Apache HTTP server* 2.4 and *MySQL* database server. The *Composer* and *Node.JS* were also installed in the development machine.

The installation involved configuring the application by adding a *.env* file which contains the basic configuration options for the application. Since *Laravel* is installed with *Composer*, a package dependency resolver for *Laravel*, it is easy to add the few features needed for rapid application development. The *Composer* was used to add an authentication service to the application which allowed the developer to add basic authentication and started to build the application.

Migration service is inbuilt in *Laravel*, so it was enabled in the application to make it easy to migrate the application from the development machine (*Windows*) to the production machine (*Ubuntu*). An *HTACCESS file* along with routing rules were added during the development.

After setup, coding of the application began with adding new controllers, models, views, and resources to the *Laravel* framework. A third-party debugger called *Debugbar* was added to make it easy for the developer to debug the application.

An open-source user interface template called *AdminLTE* was added using *npm*, a default package manager for the *Node.JS*. The developer encountered various problems working with the newly integrated *AdminLTE*. So, the developer had to remove the *npm* dependency on the *AdminLTE* user interface framework. A stable version of *AdminLTE* was added by the developer without the *npm* dependency. The developer made changes in the *Composer.JSon* file before issuing the *Composer* update command to resolve the dependencies.

The developer encountered several other issues while working on the project, such as a redirection problem while logging out of the application, error in the sidebar of the interface, and a problem in the migration of the permission module. Therefore, the developer had to roll back significant changes to troubleshoot the problems and finally got the application to be stable and run smoothly.

Implementation of PMS with Laravel Framework in Ubuntu

The production environment for the PMS in *Ubuntu* was set up with the installation of the *nginx* web server and *MySQL* database server. After setting up the production environment in *Ubuntu*, *Laravel* 5.5 was downloaded, installed and configured on the production machine using the *Composer*. Since *Laravel* is installed with *Composer,* the features needed were added for the application development.

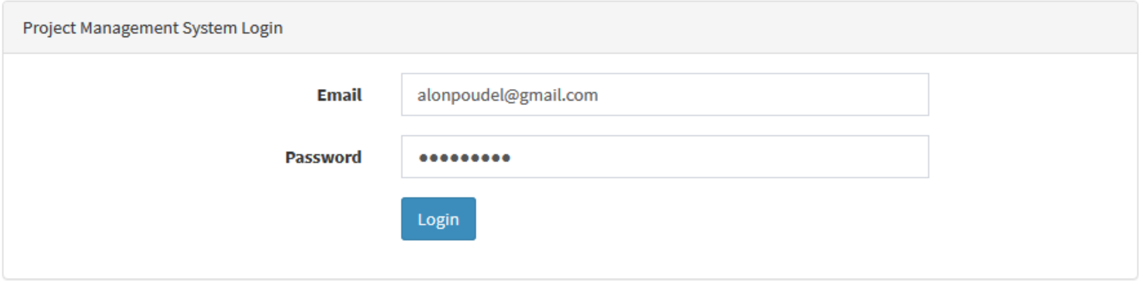
Installation involved configuring the application by adding a *.env* file which contained the basic configuration options for the application. Migration service was inbuilt in *Laravel*, so it was enabled in the application for easier migration of the application from the development machine to the production machine. An *HTACCESS file* along with the routing rules was added during the development of the PMS application.

User Interface of PMS

This section explains the user interface of the PMS. It consists of the login page, dashboard page, sign up page, roles page, document library page and discussion page.

Login Page

Figure 5 is the login page for the PMS in *Laravel* on *Windows* and *Ubuntu*, and *ASP.Net Core* on *Windows*. It is the default page that is displayed after launching the application. Users provide their registered email and password to access the PMS.



Project Management System Login

Email: alonpoudel@gmail.com

Password: ●●●●●●●●●

Figure 5. Login Page of PMS

Dashboard

Figure 6 is the dashboard page for the PMS. After logging in, the user can access the dashboard. The left navigation bar provides links to the dashboard itself, *User Management* (*Roles* and *Users*), *Members* (*Notes* and *Documents*), *Forum*, *Change Password*, and *Logout*.

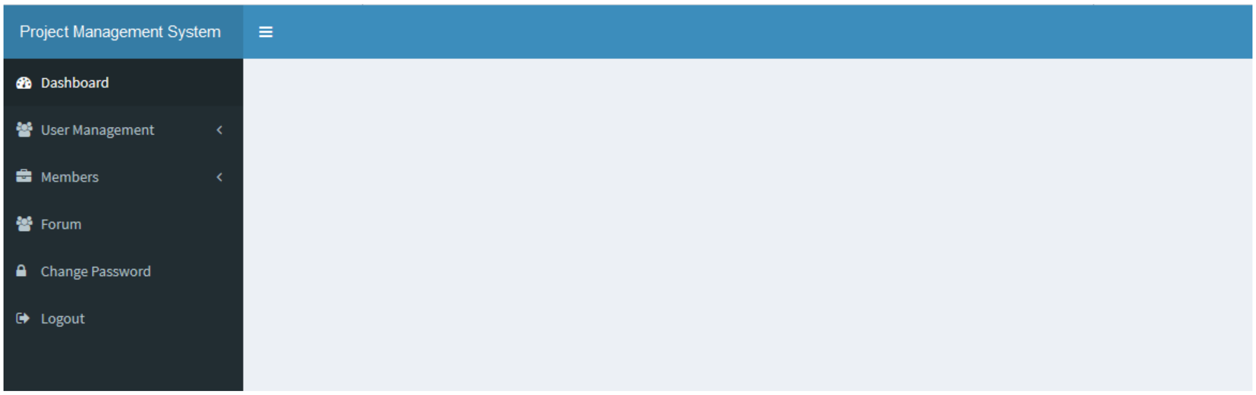
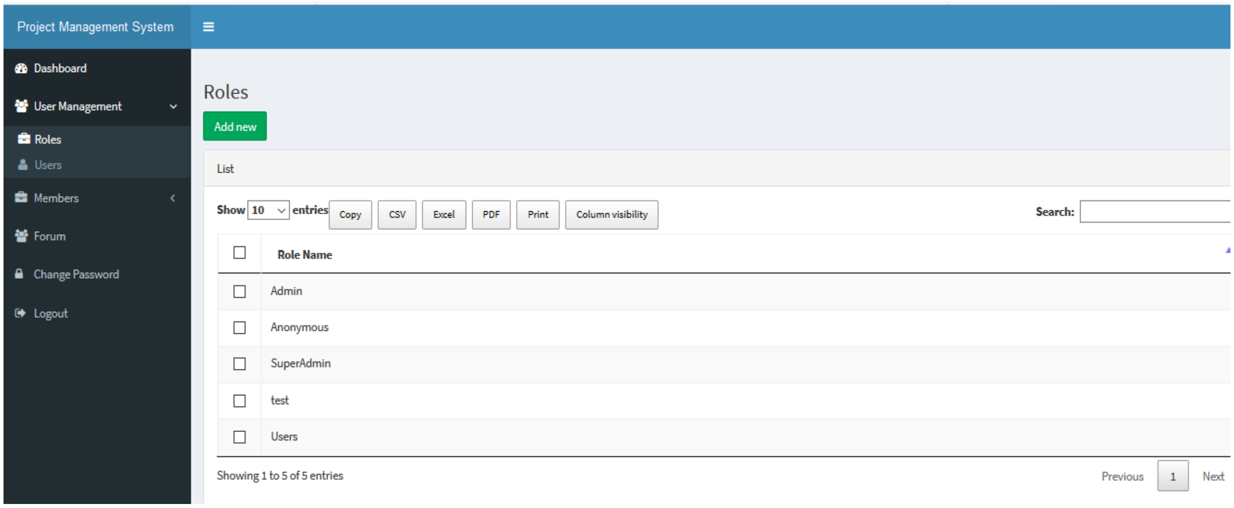


Figure 6. Dashboard Page of PMS

Roles

The Roles page (Figure 7) lets the *Administrato*r add roles of the users that have access to the PMS. The roles can be *Administrator* and *User*.



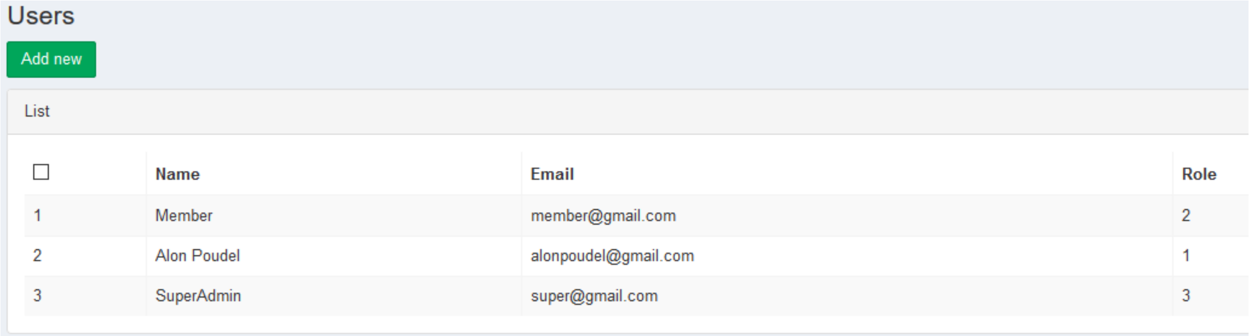
□	Role Name
□	Admin
□	Anonymous
□	SuperAdmin
□	test
□	Users

Showing 1 to 5 of 5 entries

Figure 7. Role Management Page of PMS

Users

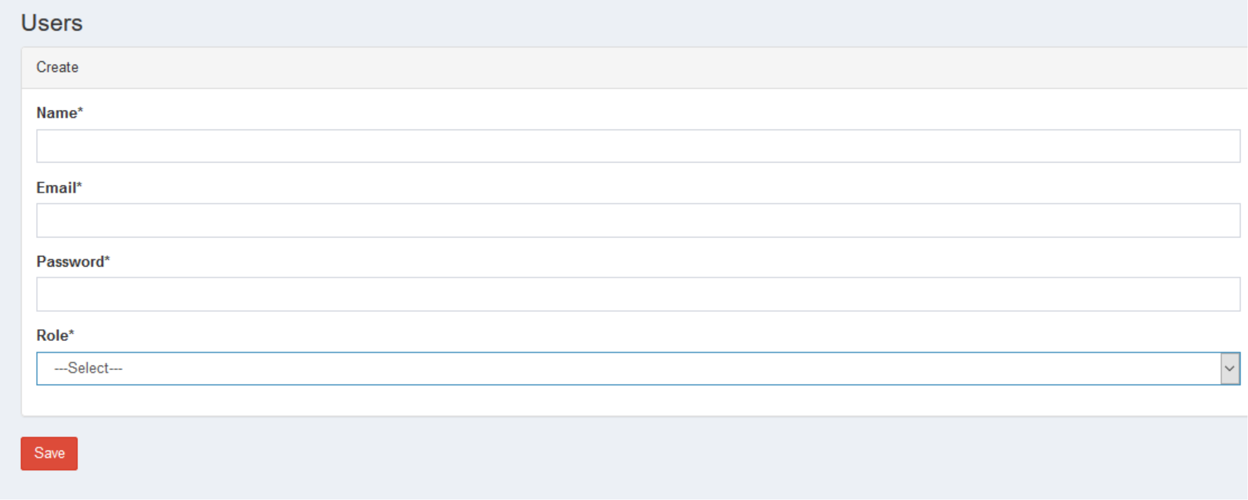
The *Users* page (Figure 8) lists the users who were added to the PMS. The admin assigns users according to the roles with associated permissions. Only users from the list can get access to the system. The numbers from the *Role* column at the end is the type of access the user has. For example, the first row with "*Member*" is as the username has the "*User*" privilege.



□	Name	Email	Role
1	Member	member@gmail.com	2
2	Alon Poudel	alonpoudel@gmail.com	1
3	SuperAdmin	super@gmail.com	3

Figure 8. Registered Users in PMS

The administrator can add new users and assign them roles with *Users* form (Figure 9). The new user can then access the PMS with the assigned privilege and the password.



Users

Create

Name*

Email*

Password*

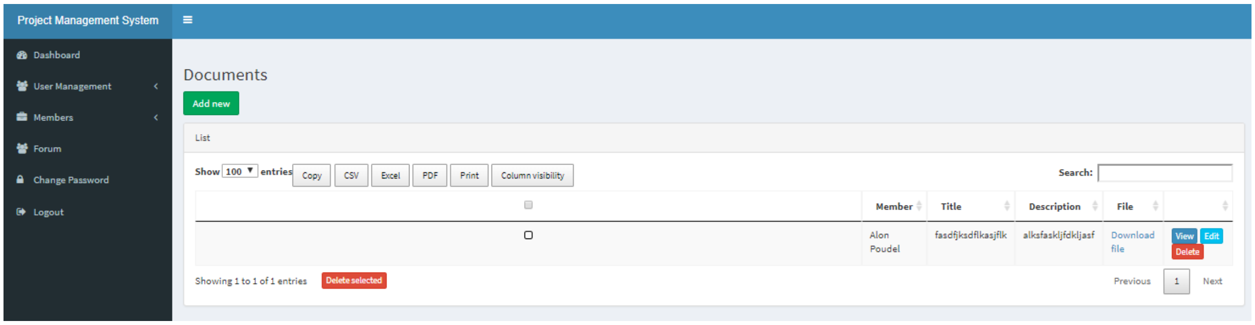
Role*

---Select---

Figure 9. Create User Form in PMS

Document Library Page

The *Documents* page (Figure 10) gives the list of documents added to the PMS. It provides the member name, title, description and the file information.



Documents

List

Show 100 ▼ entries

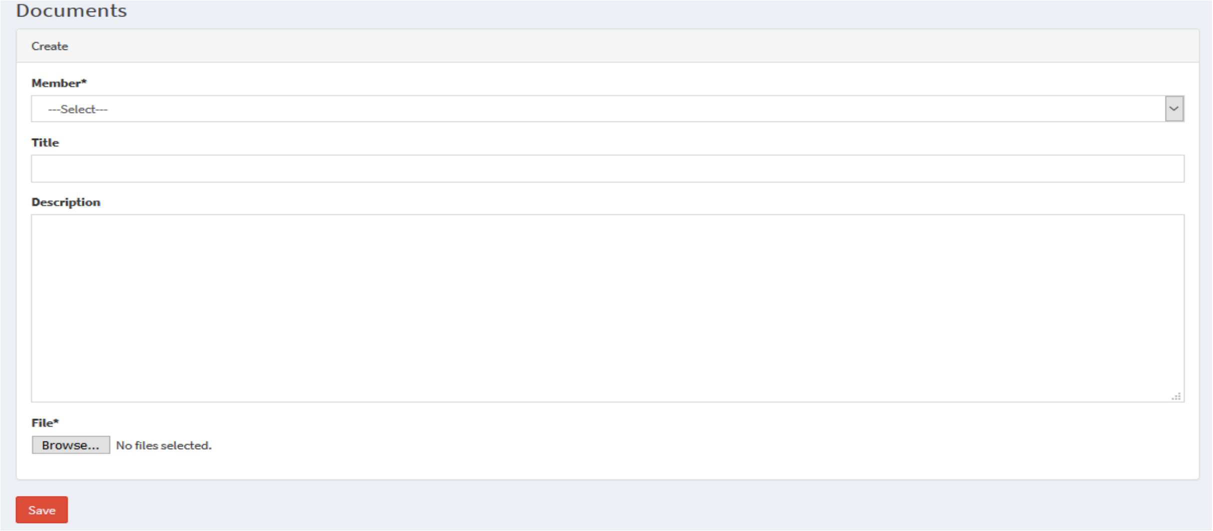
Search:

□	Member	Title	Description	File	
□	Alon Poudel	fasdfjksdflkasjflk	alksfaskljfdkljasf	Download file	View Edit Delete

Showing 1 to 1 of 1 entries

Figure 10. Document Page of PMS

The new *Documents* page (Figure 11) lets any users add a new document in the PMS. The user can select a name, title, description and file for upload.



Documents

Create

Member*

---Select---

Title

Description

File*

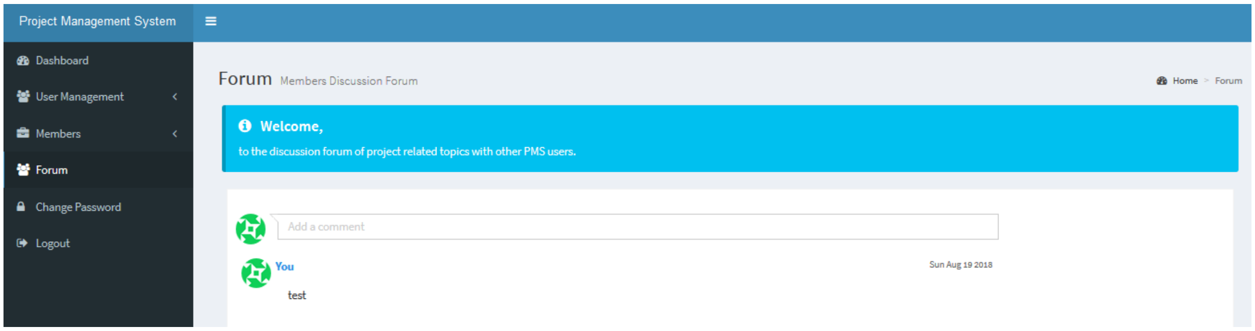
Browse... No files selected.

Save

Figure 11. New Document Page of PMS

Discussion Forum Page

The *Forum* page (Figure 12) lets users add comments and reply to the posted comments. It is great for communicating with other users on topics of interest. Users can add comments at the empty bar with the hint "*Add a comment*". To reply or edit, users can click on the "*Reply"* or "*Edit*" button and then press "*send.*"



Forum Members Discussion Forum

Welcome,

to the discussion forum of project related topics with other PMS users.

Add a comment

You Sun Aug 19 2018

test

Figure 12. Discussion Forum Page of PMS

Chapter V: Results and Discussions

This section describes the findings of the comparative study based on the literature review, system requirements, system design, implementation, testing and developer's experience.

Results

A sample size of 400 was taken for the study to show comparative performance of PMS. There are four pages (*login* page, *dashboard* page, *documents* library page, and *role* page) considered for the test, with each page being tested 100 times.

Figure 13 shows page latency of the four pages, in each of the three PMS developed in *Laravel* and *ASP.Net Core,* on *Windows* and *Ubuntu*. Latency is a delay when connecting between web server and browser, and vice versa. In Figure 13, the red graph (Range: 311-720) is the latency in *Laravel* on *Windows*, the black graph (Range: 25-201) is the latency in *Laravel* on *Ubuntu*, and the blue graph (Range: 2-28, except an outlier of 416 during the first page load time) is the latency in *ASP.Net Core* on *Windows*. *ASP.Net Core* on *Windows* has least page latency when compared to *Laravel* on *Windows* and *Ubuntu*.

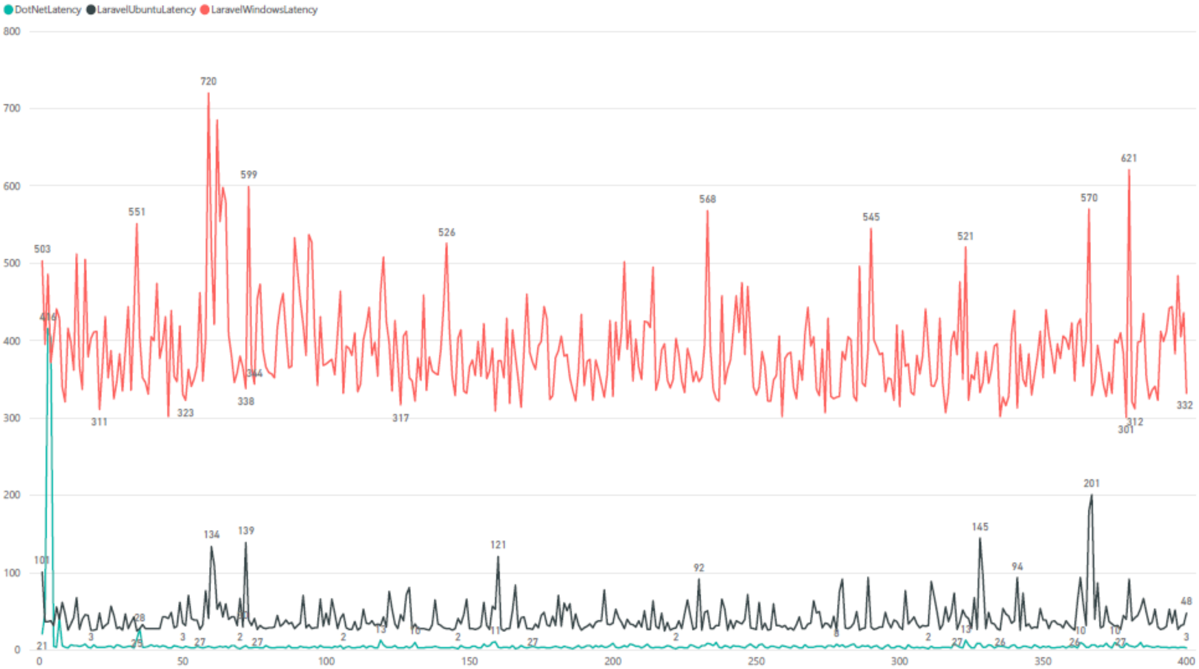
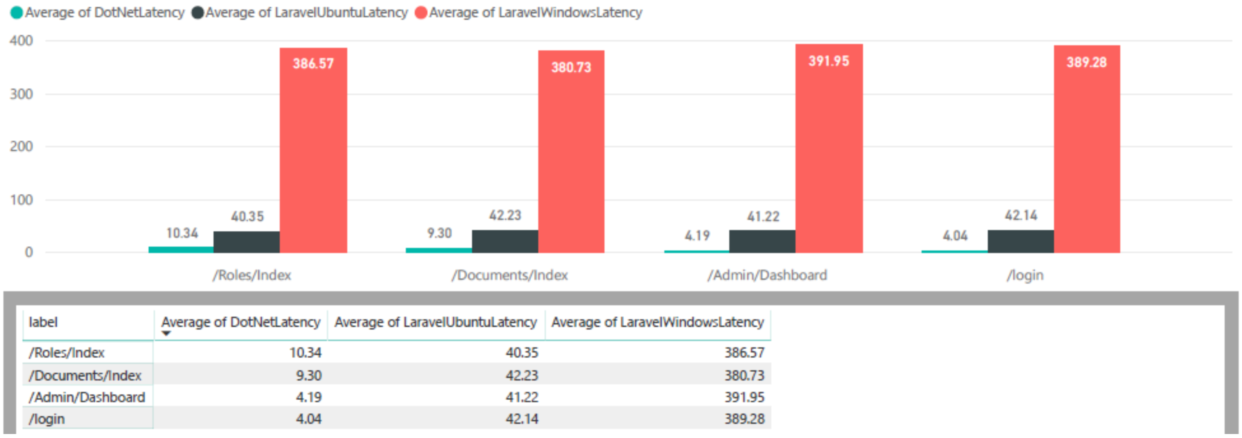


Figure 13. Page Latency of PMS

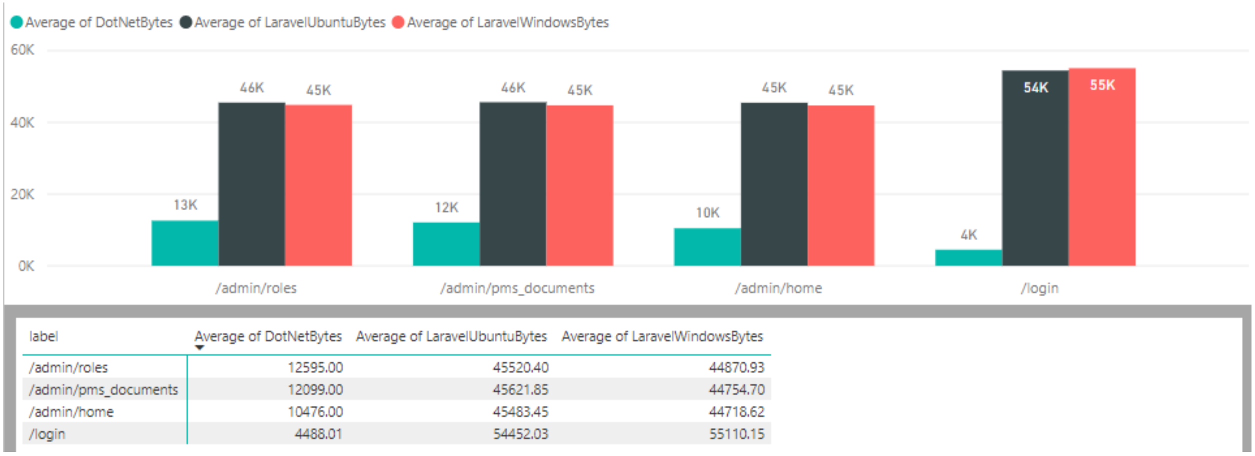
Figure 14 represents average page latency of PMS. The green bar (*/Roles/Index*: 10.34, */Documents/Index*: 9.3, */Admin/Dashboard*: 4.19, */login*: 4.04) is the average page latency of the PMS on *ASP.Net Core* on *Windows*, the black bar (*/Roles/Index*: 40.35, */Documents/Index*: 42.23, */Admin/Dashboard*: 41.22, */login*: 42.14) is of *Laravel* on *Ubuntu*, and the red bar (*/Roles/Index*: 386.57, */Documents/Index*: 380.73, */Admin/Dashboard*: 391.95, */login*: 389.28) is for *Laravel* on *Windows*. The average latency of the *ASP.Net Core* is the least in comparison.



label	Average of DotNetLatency	Average of LaravelUbuntuLatency	Average of LaravelWindowsLatency
/Roles/Index	10.34	40.35	386.57
/Documents/Index	9.30	42.23	380.73
/Admin/Dashboard	4.19	41.22	391.95
/login	4.04	42.14	389.28

Figure 14. Average Latency of PMS

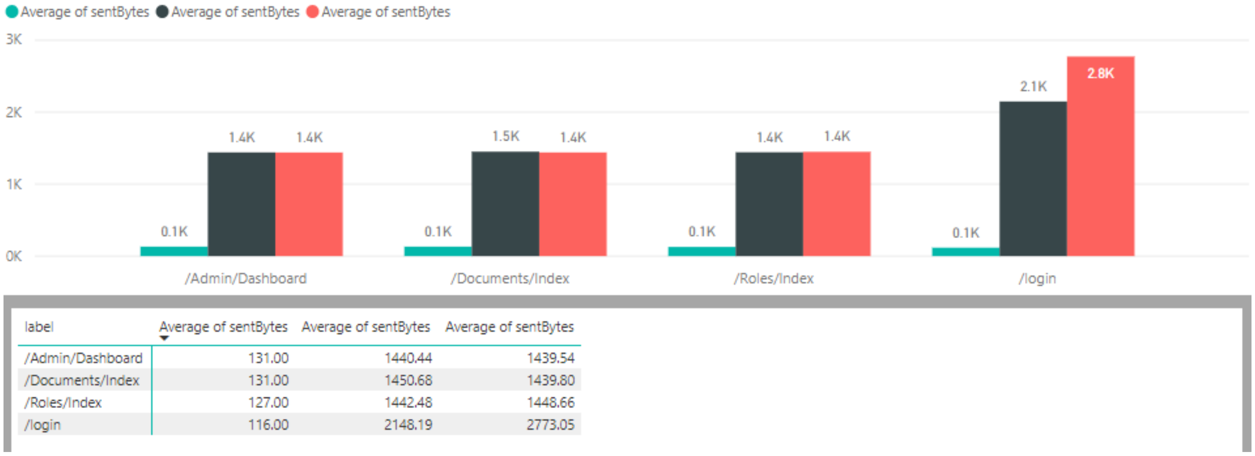
Bytes represent quantity of data in a sample response returned from the server. Figure 15 represents average bytes received from the server for each page. The green bar (*/admin/roles*: 12595, */admin/pms_documents*: 12099, */admin/home*: 10476, */login*: 4488) is the average bytes received from the server of the PMS on *ASP.Net Core* on *Windows*, the black bar (*/admin/roles*: 45520, */admin/pms_documents*: 45621, */admin/home*: 45483, */login*: 54452) is of *Laravel* on *Ubuntu*, and the red bar (*/admin/roles*: 44870, */admin/pms_documents*: 44754, */admin/home*: 44718, */login*: 55110) is of *Laravel* on *Windows*. *ASP.Net Core* has the least average bytes received from the server.



label	Average of DotNetBytes	Average of LaravelUbuntuBytes	Average of LaravelWindowsBytes
/admin/roles	12595.00	45520.40	44870.93
/admin/pms_documents	12099.00	45621.85	44754.70
/admin/home	10476.00	45483.45	44718.62
/login	4488.01	54452.03	55110.15

Figure 15. Average Bytes Received from the Server of PMS

Sent bytes represents quantity of data in a sample sent the server. Figure 16 represents the average bytes sent to the server for each page. The green bar (*/Roles/Index*: 131, */Documents/Index*: 131, */Admin/Dashboard*: 127, /login: 116) is the average bytes sent to the server of the PMS on *ASP*.*Net Core* on *Windows*, the black bar (*/Roles/Index*: 1440, */Documents/Index*: 1450, */Admin/Dashboard*: 1442, /login: 2148) is of *Laravel* on *Ubuntu*, and the red bar (*/Roles/Index*: 1439, */Documents/Index*: 1439, */Admin/Dashboard*: 1448, */login*: 2773) is of *Laravel* on *Windows*. Again, *ASP*.*Net Core* has the least average bytes sent to the server.



label	Average of sentBytes	Average of sentBytes	Average of sentBytes
/Admin/Dashboard	131.00	1440.44	1439.54
/Documents/Index	131.00	1450.68	1439.80
/Roles/Index	127.00	1442.48	1448.66
/login	116.00	2148.19	2773.05

Figure 16. Average Bytes Sent to the Server of PMS

Discussions

The discussion section is based on the developer’s experience during the development of PMS built with *ASP.Net Core* on *Windows*, *Laravel* on *Windows*, and *Laravel* on *Ubuntu*.

User Interface

With the development in the field of Information Technology (IT), interaction with applications has changed immensely. *CLI* was the most effective way to control computers in the past. Now, *GUI* has made it easier to control applications. Scripting and markup languages like *HTML*, *CSS* and *JS* has been used excessively to build a robust user interface and experience.

User Interface (UI) refers to the way users interact with devices using a series of input controls and navigational components like buttons, text fields, icons, and checkboxes. *User experience design (UX)* refers to internal experience users have when interacting with various aspects of web applications. *UI* and UX elements play a vital role to provide easy access and proper understanding of applications. Users use web applications from browsers, cell phones, search engines, and other devices. Therefore, the developer selected *Admin LTE* for PMS web applications to give a great *UI* and *UX* experience to the PMS users.

Admin LTE is an open source admin dashboard & control panel theme. *Razor* and *Blade* syntaxes are used to integrate *AdminLTE*. *Razor* is a view or template engine of *ASP.Net Core*. The developer found *Razor* view engine as a user-friendly and easy to learn view engine, which integrates *C#* with *HTML*. The code looks cleaner using *Razor* view engine when compared to views without *Razor*. Similarly, *Blade* view engine is also a simple, easy to learn, and powerful view template engine of *Laravel* web development framework. The developer found *Blade* to be less user-friendly and less organized in comparison to *Razor*. *Razor* is integrated into *Visual Studio* which auto-indents codes for a clean look and provides auto-completion of codes. Whereas, *Sublime Text* does not provide auto indention and completion feature like the *Visual Studio*.

Security

Data travels through various points between the source and destination. It makes data vulnerable to interception, alteration, and control by malicious users. These vulnerabilities can create a severe threat to data integrity and safety. Therefore, web application frameworks need to apply various security measures. There should also be a limitation on software features of web applications to reduce vulnerabilities. Going beyond the framework’s capability brings in problems like hacking, network congestion, and attacks.

Microsoft and *ASP.Net* community on *Github* addresses support and security vulnerabilities of *ASP.Net core*. If *ASP.Net Core* has any security vulnerabilities or any software related issues, it is more likely to get fixed immediately. Unlike *ASP.Net core*, if any security vulnerabilities are discovered in *Laravel*, the issues are addressed by its developer Taylor Otwell and the *Laravel Github* community. Large organizations are more likely to choose *ASP.Net Core* over *Laravel* as their web application development framework because Microsoft is a trusted name and can provide enterprise-level support.

Application Performance

Web application performance indicators can help developers make smart choices. Web applications are more robust and minimize response time. Performance of web applications depend on aspects such as application bandwidth, number of user requests on the network, different application protocols, application hosting environment, programming tools, and programming languages. To increase application performance, pages must render efficiently with less content and less communication between applications and browsers. It should also focus on easily scalable web application framework. Web applications and portals need to be fast and flexible. Long page load time influences users to move on to the next website instead.

PMS in *ASP.Net Core* on *Windows* is better when compared to Laravel on average page latency, average bytes received from the server, and average bytes sent to the server. Then the PMS in *Laravel* on *Ubuntu* is slightly better than the PMS in *Laravel* on *Windows*. Therefore, the developer recommends *ASP.Net Core* on *Windows* as a preferred framework in terms of web application performance.

Technology

Web application development comprise of two categories: client-side programming and server-side programming. Client-side programming, also known as front-end programming, includes everything that a user sees on a web browser. Whereas server-side programming, also known as back-end programming, deals with logic and databases of web applications. There are plenty of tools and technology to choose from for front-end and back-end development.

Therefore, developers must take performance, programming and scripting languages (*HTTP*, *JS* and *CSS*), user interface, scalability, software design pattern, web frameworks, application protocols, servers, database, and storage into account before opting for the right technology. Furthermore, it is only logical to avoid creating something that is already there and improved upon rigorously. Choosing a framework that a development team is comfortable with, is a significant advantage.

Accordingly, the developer focused on *Laravel*, *ASP.Net core*, *SQL*, *MySQL*, *Object Relational Mapper (ORM)*, *Entity Framework*, *Eloquent Framework*, *PhpMyAdmin*, *XAMPP*, *nginx,* and *Admin LTE*.

The *Object Relational Mapper (ORM)* is a tool or technique that allows conversion of data between incompatible systems using an object-oriented paradigm. The *ASP.Net Core* uses *Entity Framework* as an *ORM*. The *Entity Framework* (Core) is not available in *ASP.Net Core* project (if a new project is selected), but it was installed through the *NuGe*t package manager. *Laravel* 5.5 uses *Eloquent* as an *ORM* and can be found in-built within the framework. The *ASP.Net* framework (except *.Net Core*) supports *Entity Framework* integrated with *EDMX* (“*.EDMX*” is a file extension) database model. *EDMX* allows importing database and its components directly from the database server. The developer wrote codes for the database context, database set, transient service, and models. It can efficient if the *EDMX* database model was available for *ASP.Net core*.

In the context of *Laravel*, *Eloquent* handles most of the database connection and database components easily. The developer wrote codes for the model to interact the with the database server.

PMS developed using *ASP.Net Core* utilizes *C#* as the programming language. The web application developed using *Laravel* 5.5 utilizes *PHP* as the programming language. Both programming languages are general-purpose programming languages. However, the developer found *PHP* easier to learn than *C#*. Nevertheless, *C#* is a preferred language to develop enterprise applications than *PHP*.

ASP.Net Core based PMS uses Microsoft *SQL* Server 2014 as *DBMS* server, and *Laravel* based web application uses *MySQL* as *DBMS* server. *MSSQL Server* has *SQL Management Studio* as the *GUI* to manage database. Whereas, *MySQL* database server *has PhpMyAdmin* as it's *GUI*. The developer preferred *MSSQL* over *MySQL* based on their user interface, and integration with PMS.

License

Quality, security, flexibility, customizability, support options, interoperability, and security of open-source software are attractive features of open-source software. However, developers need to know about the terms and conditions of respective open source licenses before using it to avoid legal consequences. It is also important to research about licenses of the framework's add-ons, plug-ins, and extensions. *ASP.Net Core* and *Laravel* frameworks are couple of examples of such open-source software. Both *ASP.Net Core* and *Laravel* have MIT License.

Testing

Testing is performed to increase code quality, reusability, efficiency, and bug fixes. The primary purpose of testing is to ensure that each unit or function or area of the software performs as expected. A well-tested web application that behaves and performs as expected enhances overall user satisfaction. Moreover, it increases the chance of new features working correctly without unexpected output.

While testing and debugging, PMS built with *ASP.Net Core* and deployed on *Windows* performed better than PMS in *Laravel* on *Windows* and in *Laravel* on *Windows*. This is because *Visual Studio* provides debugging and intelligence feature, whereas *Sublime Text* does not provide either. So, the developer recommends *ASP.Net Core* on *Windows* based on application testing.

Learning Curve

The developer took less time and effort to complete PMS web application built with *ASP.Net Core* when compared to the PMS web application built with *Laravel* on *Windows* and *Ubuntu*. Choosing the steepest learning curve is essential to building a web application. *Visual Studio 2017 Community* is more advanced, helpful and user-friendly tool in developing *ASP.Net Core* applications. However, *Visual Studio* takes more time to install and takes more disk space on the development machine in comparison to *Sublime Text 3*, which is lightweight and fast.

Development Framework Communities

Active community support is the lifeline of any design pattern. Diverse and highly motivated members are the signs of a great community. Choosing a framework with a friendly community ensures the web application’s longevity. Proper documentation of the web application framework is essential for the web application’s growth and stability. The web application framework backed by the active community will ensure regular updates, application robustness, and high performance.

Both *ASP.Net Core* and *Laravel* have a good community presence. However, the developer found that *ASP.Net Core* has a greater community on *Github* than *Laravel*.

Chapter VI: Conclusion

The developer concludes that the PMS built with *ASP.Net Core* on *Windows*, is a better choice when compared to the PMS built with *Laravel* framework on *Ubuntu* and *Windows*. The study was based on the learning curve, features, implementation of *MVC*, and application performance. The developer spent less time to develop PMS on *ASP.Net Core* in comparison to the *Laravel* framework. Individual latency, average latency and average bytes measurements were the performance benchmarks set by PMS built with *ASP.Net Core*. PMS developed on *ASP.Net Core* performed better than PMS developed on the *Laravel* framework on Window and *Ubuntu*. Furthermore, the developer felt more comfortable setting up the environment and building the application in the *ASP.Net Core* on *Windows*. While *Laravel* provided better *ORM* feature and easier programming language (*PHP*), the developer spent a lot of time to set up *Laravel* environment and to build the application.

For future work, the developer recommends deploying *PMS* with *ASP.Net Core* on a *Windows 10* virtual machine to get a better perspective on the application performance. It is very likely that the page load time and byte size would change depending on the physical and virtual host environment of the application. Hence, it will provide a comprehensive perspective of *PMS* on *ASP.Net Core* and *Laravel.*

Overall, the developer learned about *MVC, ASP.Net Core*, and *Laravel*. These are important concepts and software knowledge that will help the developer to further contribute in the field of computer science, as an academic and a professional web developer.

References

Alfat, L., Triwiyatno, A., & Isnanto, R. R. (2015). *Sentinel web: Implementation of Laravel framework in web-based temperature and humidity monitoring syste*m. 2015 Second International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), 46-51.

Bautista, N. (2012*). Building web applications from scratch with laravel*. Retrieved from HTTPs://code.tutsplus.com/tutorials/building-web-applications-from-scratch-with-Laravel--net-25517

Bisson, S. (2017). *Net framework or .Net core? When to use which*. Retrieved from HTTPs://www.infoworld.com/article/3180478/development-tools/net-framework-or-net-core-when-to-use-which.HTML

Helm, R., Johnson, R. E., Gamma, E., & Vlissides, J. (2000). *Design patterns elements of reusable object-oriented software*. Charlesbourg, Québec: Braille Jymico Inc.

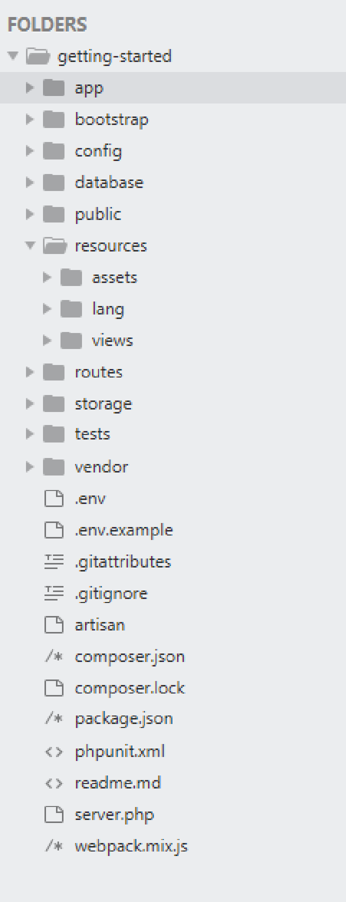
Jailia, M., Kumar, A., Agarwal, M., & Sinha, I. (2016). *Behavior of MVC (Model View Controller) based web application developed in PHP and .NET framework*. 2016 International Conference on ICT in Business Industry and Government (ICTBIG), 1-5.

Komara, H., Hendradjaya, B., & Saptawati, G. P. (2016). *Dynamic generic web pattern formulti-platformm*. 2016 International Conference on Data and Software Engineering (ICoDSE), 1-5.

Stauffer, M. (2016). *Laravel: Up and running: A framework for building modern PHP Apps* (6th ed.). Sebastopol, CA: O'Reilly Media.

Appendix A: PMS File Directory in Laravel

Figure 17 shows folders and files of the PMS in the *Sublime Text Editor* for *Laravel*. Most of the content in the figure is auto-generated. However, it is the developer's responsibility to modify, add and delete files according to the requirements of the web application under development.



```
FOLDERS
getting-started
  app
  bootstrap
  config
  database
  public
  resources
    assets
    lang
    views
  routes
  storage
  tests
  vendor
  .env
  .env.example
  .gitattributes
  .gitignore
  artisan
  composer.json
  composer.lock
  package.json
  phpunit.xml
  readme.md
  server.php
  webpack.mix.js
```

Figure 17. File Structure of the PMS in the Sublime Text Editor

Following is the list of files added and modified by the developer to the file directory of the PMS built with *Laravel* framework:

- ***PMSDocumentController.php*****:** Users add documents into the PMS with the help of this file. The documents are shared with other users to view and download. The document can be a pdf or text file, and it can be edited and deleted by the users.
- ***PMSForumController.php*****:** It provides a common platform to all the users to discuss on a common topic.
- ***PMSNoteController.php*****:** It adds and shares notes among all the members in the PMS.
- ***RolesController.php*****:** It manages roles in the PMS. The role can be an administrator or a regular user. Administrator has full privilege to add, delete, edit and read, unlike the regular user who has limited privileges throughout the PMS.
- ***UsersController.php*****:** It manages users in the PMS. It can be accessed only with administrator privilege. It can be used to create a new user, delete the existing user, and update the information of the existing user or list out the available users in the PMS.
- ***UploadFileTrait.php*****:** It is a helper file in *Laravel* to help users upload files. It uploads users' file based on file type. For example: If a file is an image, it resizes the file and uploads it in the server. It is also manages file permission.
- ***Controller.php*****:** It is the *base controller* class which provides functionalities to newly created controllers.
- ***StorePMSDocumentsRequest.php***: It validates data obtained from the form submitted by the user. It validates data based on the given parameters. It executes database operation if the validation is successful but denies it if they are not validated.

For example: If a form requires numeric data, but the user supplies special character, then the application will issue a warning that the input data is not valid. Similarly, *StorePMSNotesRequest.php, StoreRolesRequest.php, StoreUsersRequest.php, UpdatePMSDocumentsRequest.php, UpdatePMSNotesRequest.php, UpdateRolesRequest.php, and UpdateUsersRequest.php* also validate data obtained from the form submitted by users.

- ***PMSDocument.php***: This file inherits its functionalities from the Model class. *PMSDocument* consists of field names such as name, description, file, user_id which are used to interact with the corresponding field names in the database using *Eloquent ORM* in *Laravel*. These field names are bound in the forms, which in turn carry data from users and interacts with the database. Similarly, *PMSNote.php, Role.php, and User.php* inherit from the Model class.
- ***App.php*:** This file provides basic configuration of the application. The properties of the application such as the application name, application environment, debugging settings, application URL, application providers, and aliases reside here.
- ***Database.php*:** It consists of database connection settings for the application. Such as name and type of the database to connect, and user credentials of the database to connect.
- ***PMSDocumentFactory.php***: It is a file that comes with *Laravel* which allows creating fake data for models. It is beneficial to test the application and seed the database with the fake data to test the application in action before it is made available

for real-time users to use. Similarly, *PMSNoteFactory.php, RoleFactory.php,* and *UserFactory.php* also allow creating fake data for models.

- ***Migration (folder)*****:** It is a feature in *Laravel* that allows creation of tables in the database. For example: During deployment of PMS, it is not necessary to create tables manually in the database, the migration feature in *Laravel* creates those required tables into the database.
- ***Seed (folder)*****:** It allows to insert default data into the tables in the database.
- ***Public (folder)*****:** The public folder consists of web resources such as *CSS,* images, and *JS* files of the application.
- ***Resources/Views*****:** It contains the user interface or presentation part of the application. The controller calls view method and renders layout in these files.
- ***.env*****:** It is a configuration file for *Laravel* which consists of important variable names and their corresponding values.

Appendix B: Sequence Diagram of PMS in Laravel

Figure 18 shows a user sending a request to display a form via an interface. PMS accepts the request and responds with a login form. The user fills up the login details and submits the form. The application validates the user details, and if the validation is successful, PMS redirects the use to the dashboard page. If the validation is unsuccessful, the application returns the login page with validation errors.

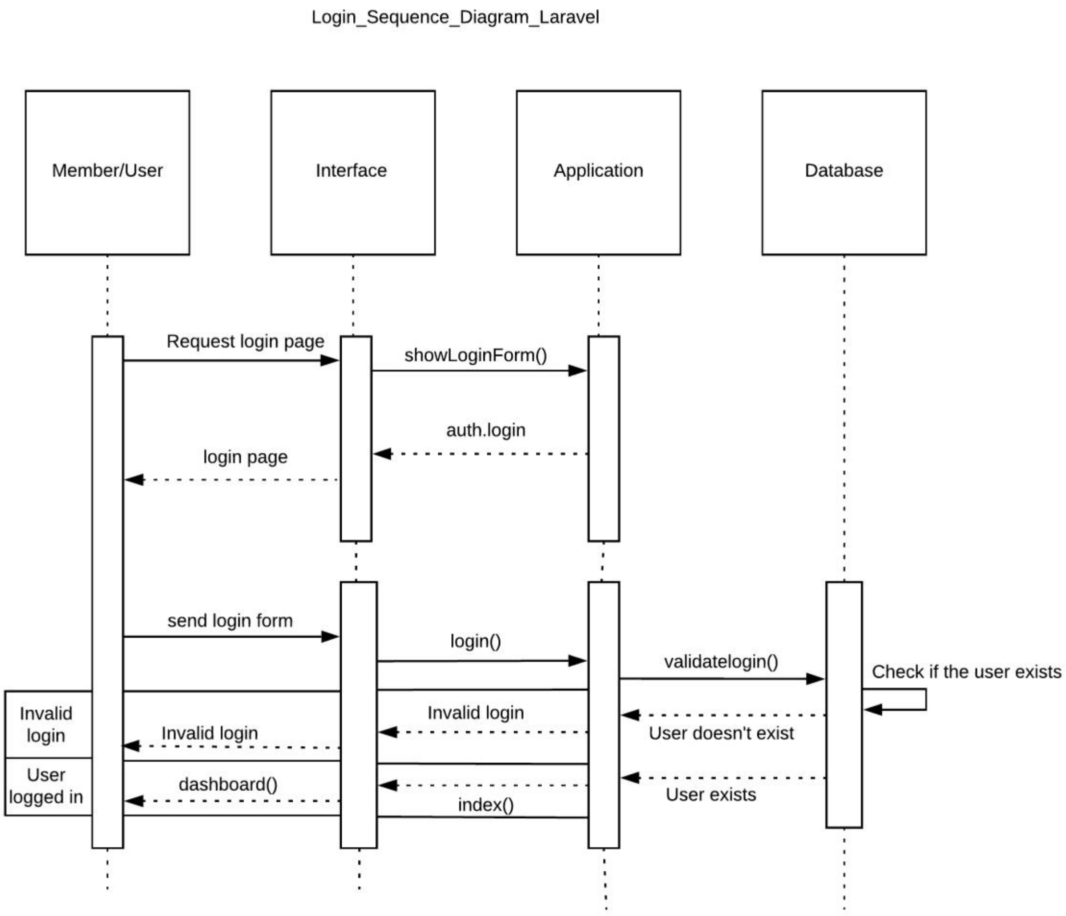


Figure 18. User Login Sequence Diagram of PMS in *Laravel* on *Windows* and *Ubuntu*.

In Figure 19, when a user clicks on *'Add Document'* button, web browser (interface) sends a *HTTP GET* request to a web server (application). Web server accepts *HTTP GET* request, and a *create()* function is triggered in the document controller. Then it returns a view page with a new document form. The user fills up empty fields in the new document form and submits it to create a new document. Then the web browser sends an *HTTP POST* request to the web server (Application), the web server accepts the request, and *store()* function validates required fields received from the posted data. If the validation of required fields is successful, the request is forwarded to *create()* function where a database query is generated based on create request to the database server. Data is inserted into the database and notification of successful creation of a new document is sent to the web browser as a response. If the validation is unsuccessful, a function returns a view page with a message stating that required fields are not filled.

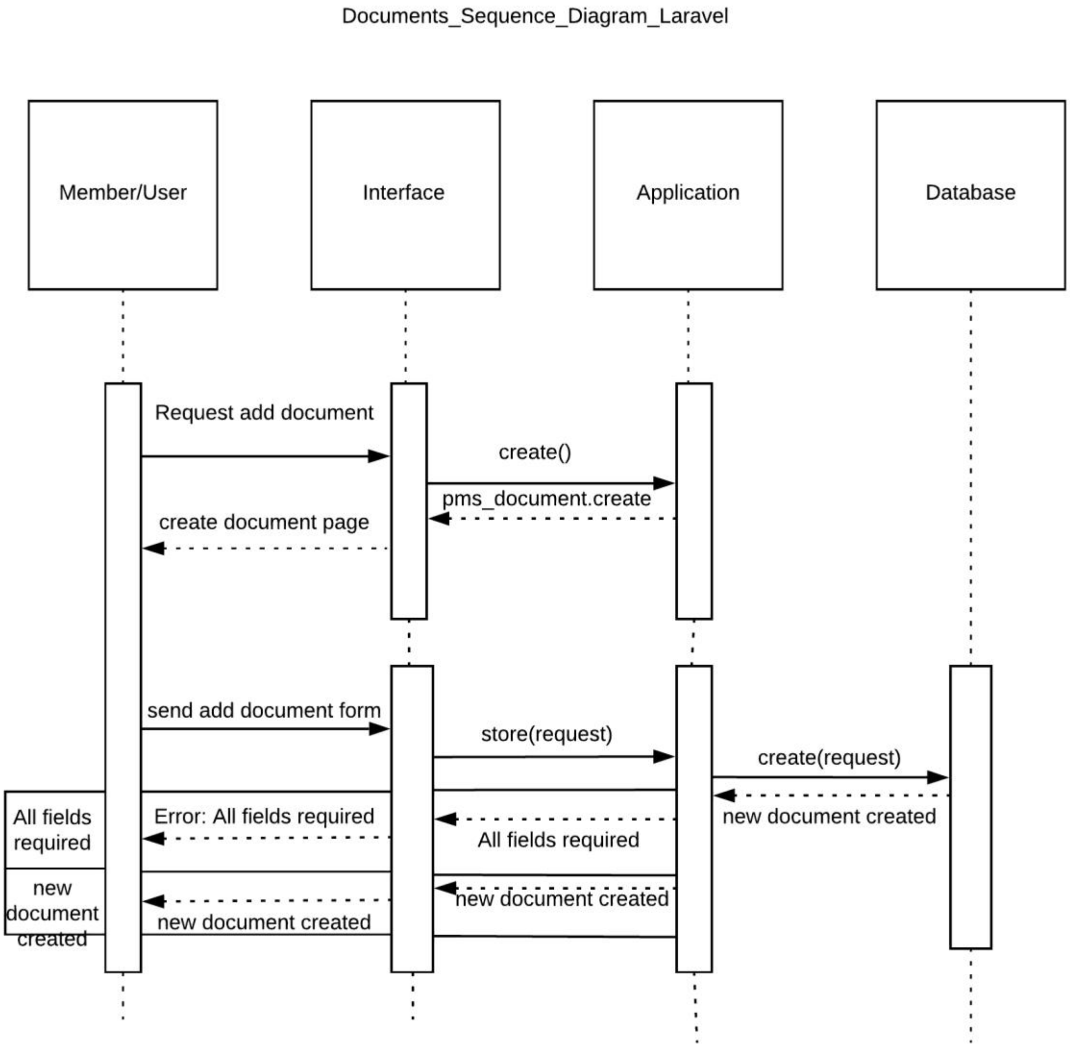


Figure 19. Add/Update Documents Sequence Diagram of PMS in *Laravel* on *Windows* and *Ubuntu*.

A user sends a request to PMS to show a forum page via an interface, and the forum page is displayed. The user adds a new comment and submits the form, which is then sent to the application. PMS sends the request to the database where the new comment gets inserted. Forum page is displayed with a list of available comments. The user can reply to the comments posted, which is sent to PMS and finally inserted into the database. New replies list is displayed after the update.

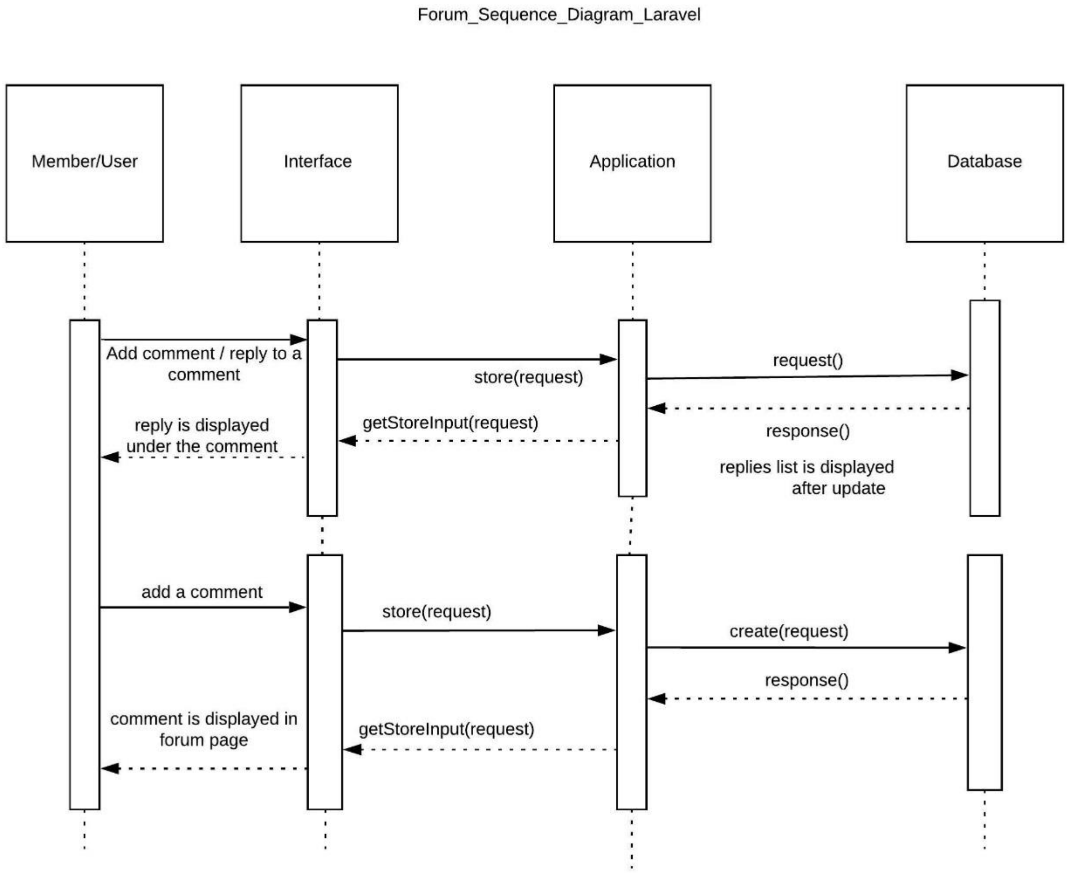

Forum_Sequence_Diagram_Laravel
Member/User
Interface
Application
Database
Add comment / reply to a comment
store(request)
request()
response()
getStoreInput(request)
reply is displayed under the comment
replies list is displayed after update
add a comment
store(request)
create(request)
response()
comment is displayed in forum page
getStoreInput(request)

Figure 20. Discussion Forum Sequence Diagram of the PMS in *Laravel* on *Windows* and *Ubuntu*.

A user sends a request to PMS to show a *Notes* page via an interface, and a *Notes* page is displayed. The user adds a new note and submits it, which is then sent to PMS, and PMS send the request to the database where the new note gets inserted along with the member’s name. The *Notes* page is displayed with a list of available notes. Also, the user can update the notes posted, which is sent to the application and finally inserted into the database.

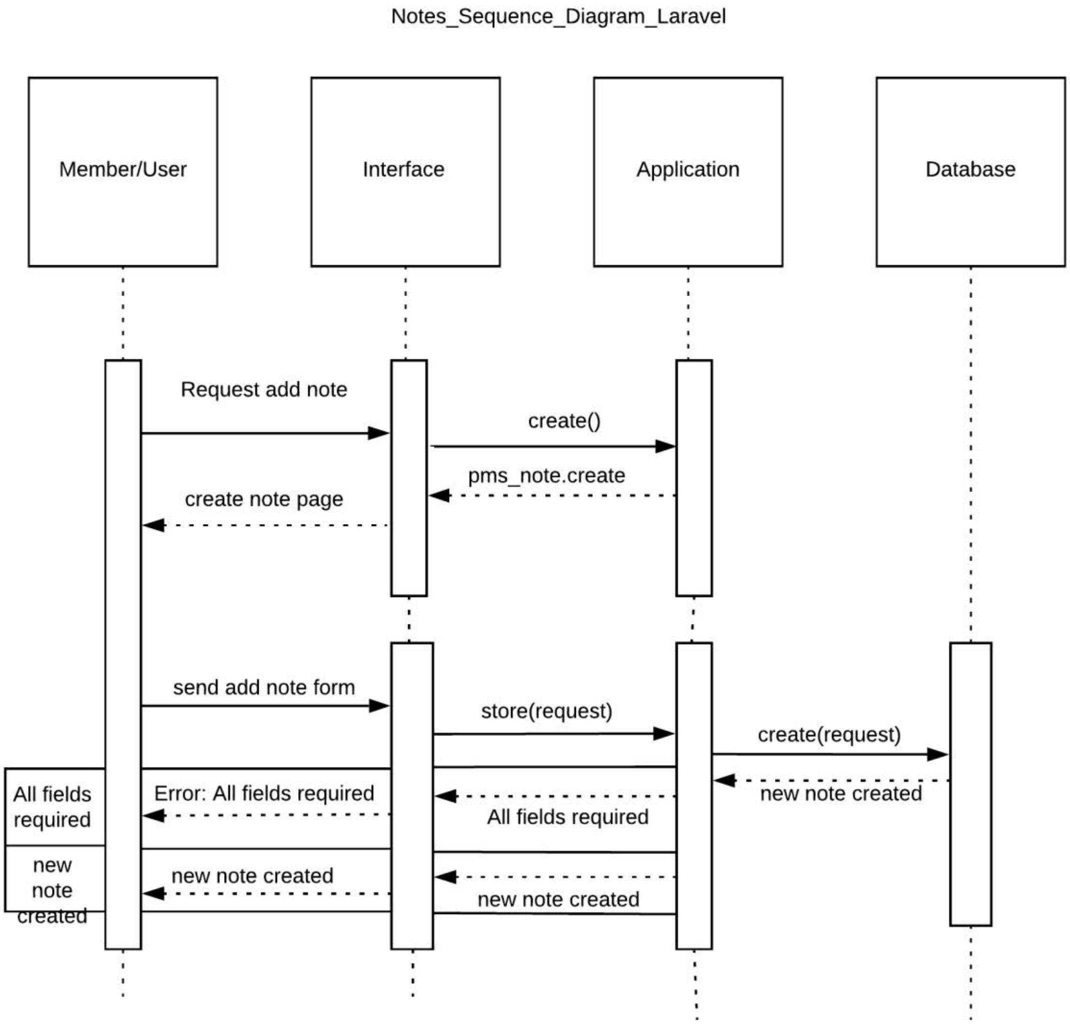
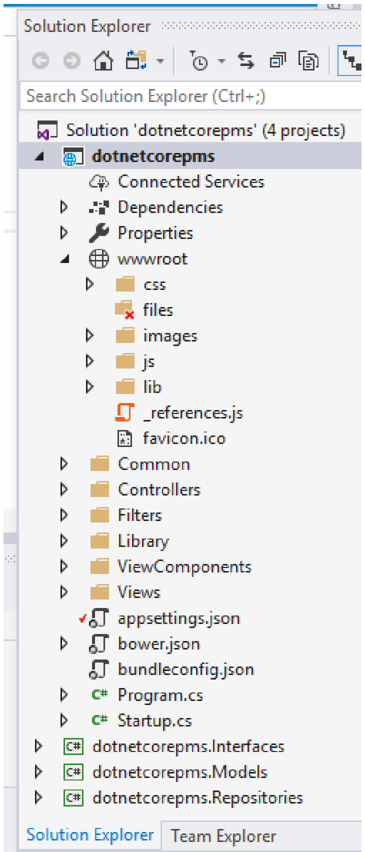


Figure 21. Add/Update Notes Sequence Diagram of PMS in *Laravel* on *Windows* and *Ubuntu*.

Appendix C: PMS File Structure in ASP.NET Core Framework

Figure 22 shows folders and files of PMS listed in the *Visual Studio* for *ASP.Net core*. Most of the content is auto-generated. However, it is the developer's responsibility to modify, add and delete files according to the requirements of the web application.



```
Solution 'dotnetcorepms' (4 projects)
◢ dotnetcorepms
    Connected Services
  ▷ Dependencies
  ▷ Properties
  ◢ wwwroot
    ▷ css
      files
    ▷ images
    ▷ js
    ▷ lib
      _references.js
      favicon.ico
  ▷ Common
  ▷ Controllers
  ▷ Filters
  ▷ Library
  ▷ ViewComponents
  ▷ Views
   ✓appsettings.json
  ▷ bower.json
    bundleconfig.json
  ▷ Program.cs
  ▷ Startup.cs
▷ dotnetcorepms.Interfaces
▷ dotnetcorepms.Models
▷ dotnetcorepms.Repositories
```

Figure 22. File Structure of PMS in *ASP.Net Core* in *Visual Studio*

Following is a list of files added/updated by the developer to the file directory:

- ***AdminController.cs***: It displays dashboard of the PMS after a user is successfully logged in.
- ***BaseController.cs***: It provides basic and common functionalities to all the other controllers in the web application.

- ***Common/ModuleHelper.cs*****:** It lists main menu items and their sub-menu items (if any) in the navigation sidebar.
- ***DocumentsController.cs*****:** It allows users to upload their documents into the server and view uploaded documents. A user can upload a word documents or text files via this feature.
- ***EncryptionLibrary.cs*****:** It encrypts and decrypts passwords for authentication purposes
- ***ErrorController.cs*****:** It redirects PMS to an error page if an error occurs.
- ***ForumController.cs*****:** It provides a common platform for users in PMS to communicate with each other. Users provide their comments on topics and discusses with each other.
- ***NotesController.cs*****:** It allows users to write notes and share notes among users in the PMS.
- ***RolesController.cs*****:** It manages roles in PMS. The roles such as Administrator, and Users are created, deleted, viewed and updated by this feature.
- ***UsersController.cs*****:** It is responsible for managing users in the application. The *UsersController* can be used to create, delete, view and update the users.

Appendix D: Sequential Diagrams of PMS in ASP.Net Core

In Figure 23, a user sends a request to show a form via an interface, PMS accepts the request and responds with the login form. The user fills up login details and submits it. PMS validates the details. If validation is successful, PMS redirects to dashboard page. If validation is unsuccessful, it returns a login page with validation errors.

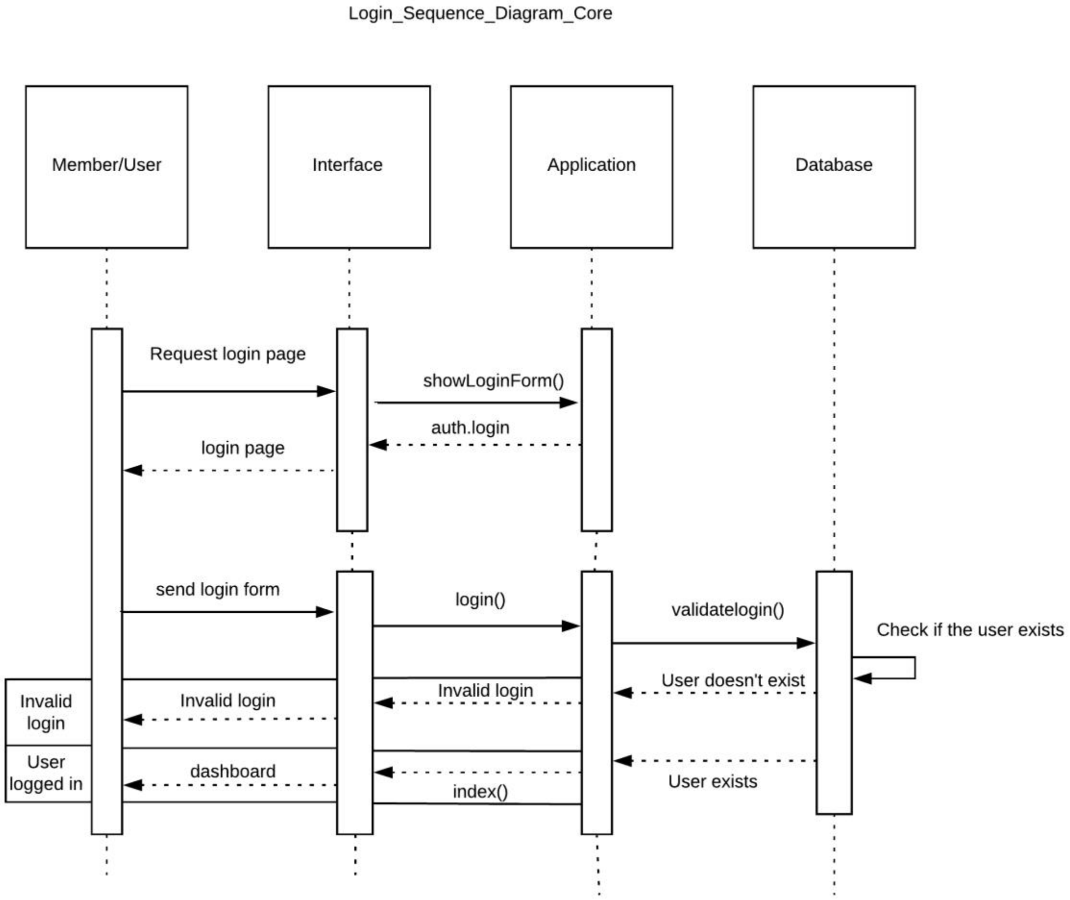


Figure 23. User Login Sequence Diagram of PMS in *ASP.Net Core* on *Windows*

As shown in Figure 24, to add a new document, a web browser (interface) sends a *HTTP GET* request to a web server (application). Document controller gets triggered once the web server accepts *HTTP GET* request. Then it returns a view page which consists of a new document form. It sends a response to the web browser, which displays the new document form. The user fills up the fields in the new document form and submits it to create a new document. The web browser sends a *HTTP POST* request to the web server (Application). The web server accepts the request, and *store()* function validates required fields received. If validation of required fields is successful, the request is forwarded to *create()* function where a database query is generated in the database server based on the create request. Data is inserted into the database, and notification of successful creation of a new document is sent to the web browser as a response. If validation is unsuccessful, a function returns a view page with a message that the required fields are not filled.

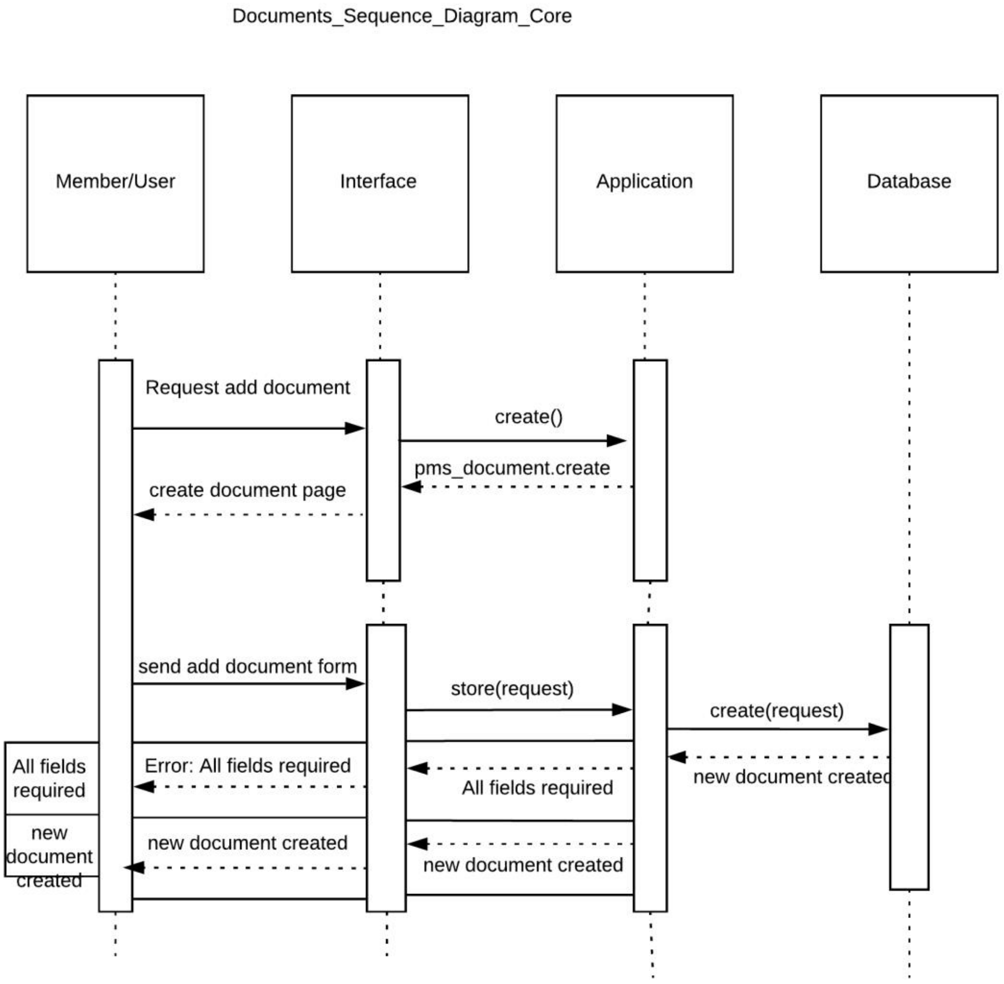


Figure 24. Add/Update Documents Sequence Diagram of PMS in *ASP.Net Core* on *Windows*

In Figure 25, a user sends a request to PMS to show a forum page via an interface, and the forum page is displayed. The user adds a new comment on comment form and submits the form, which is then sent to PMS. PMS sends the request to a database where the new comment gets inserted. Forum page is displayed with a list of available comments. The user can reply to comments posted, which is sent to PMS and finally inserted into the database. A list of new replies is displayed after the update.

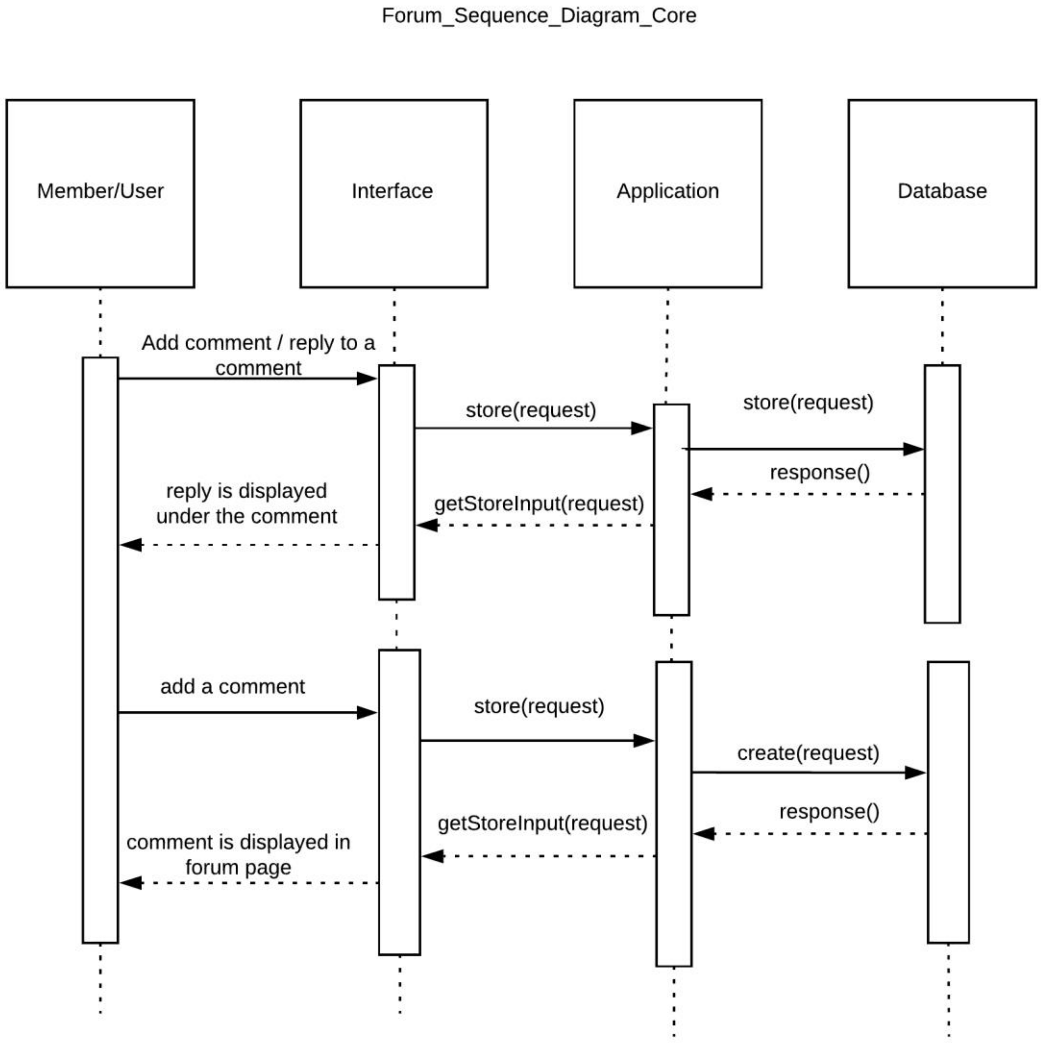


Figure 25. Discussion Forum Sequence Diagram of the PMS in *ASP*.*Net Core* on *Windows*

In Figure 26, a user sends a request to PMS to show a *Notes* page via an interface, and a *Notes* page is displayed. The user adds a new note and submits it, which is then sent to PMS. PMS sends the request to the database where the new note gets inserted along with the member's name. A *Notes* page is displayed with a list of available notes. The user can update the notes posted, which is sent to PMS and finally inserted into the database.

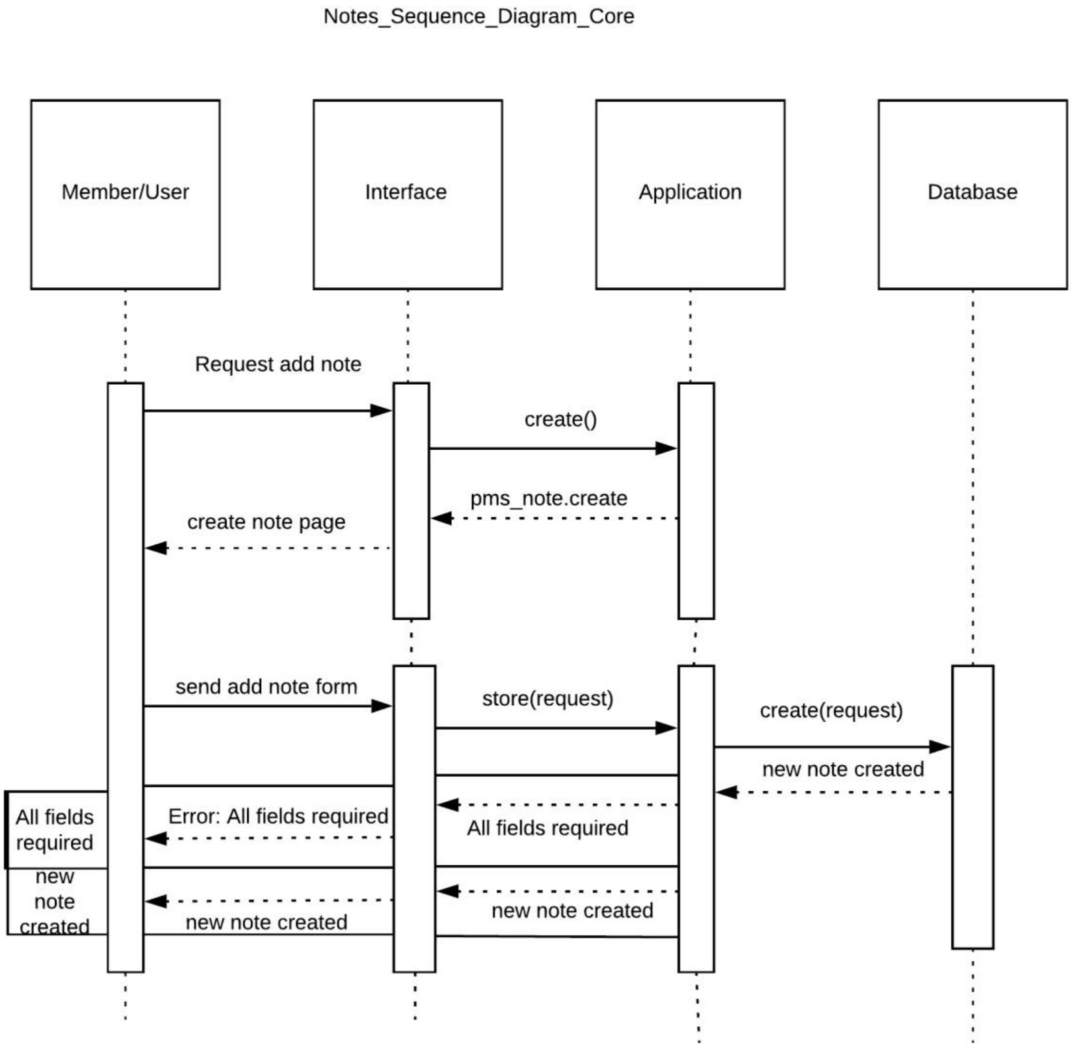


Figure 26. Add/Update *Notes* Sequence Diagram of PMS in *ASP.Net Core* on *Windows*

Appendix E: Class Diagrams of PMS

Figure 27 shows properties and methods of all classes are public. Users class contains properties such as name, emailid, username, and password. Users class is used to add, update, delete, validate, and retrieve user information. It is accessed by *Forum*, *Roles, Notes* and *Document* classes to execute inherited methods: add/ update/ delete/ validate.

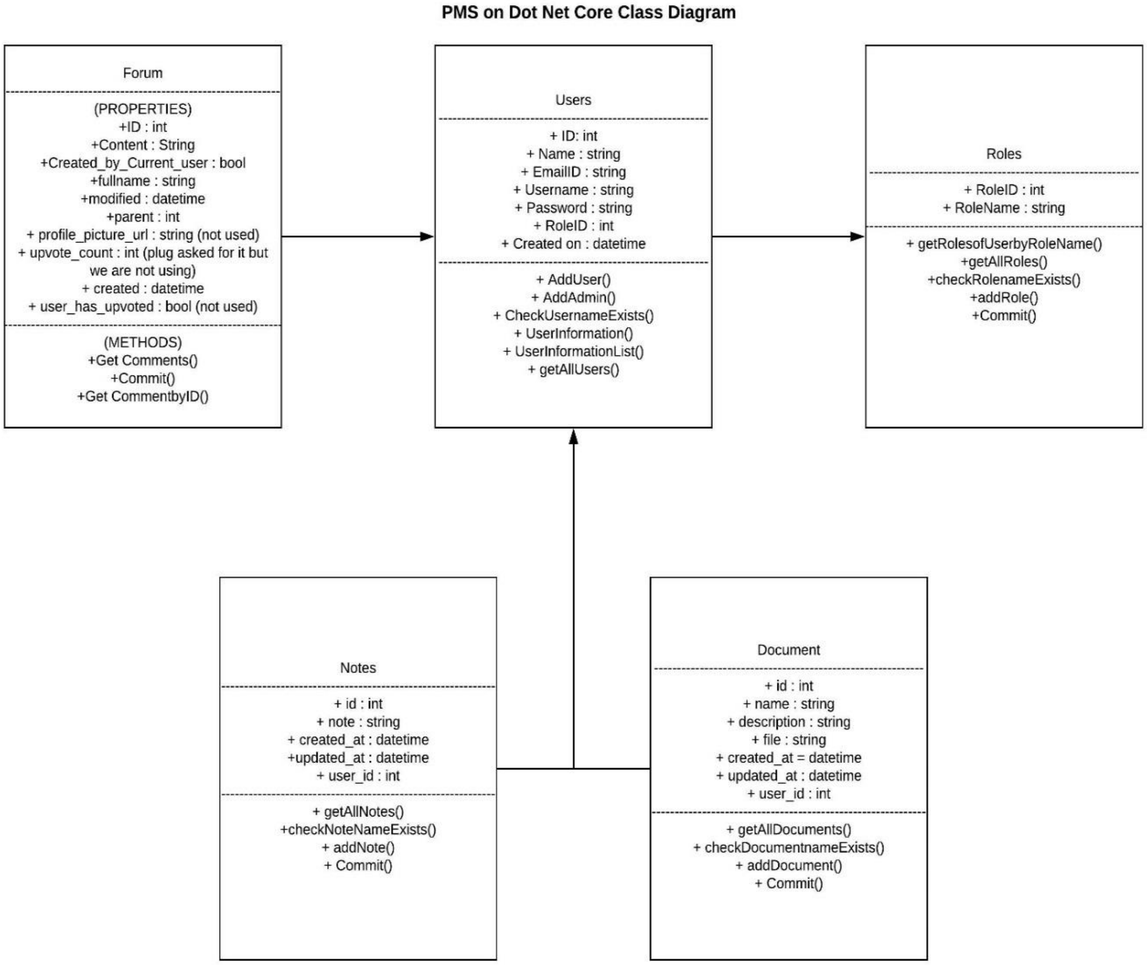


Figure 27. Class Diagram of PMS in *ASP.Net Core* on *Windows*

Figure 28 shows properties and methods of all classes are public. Similar to Figure 27, Users class contains properties such as name, emailid, username, and password. Users class is responsible to add, update, delete, validate, and retrieve user information. Users class is accessed by *Forum*, *Roles*, *Notes* and *Document* classes to execute add/ update/ delete/ validate methods. *Comment_reports* and *Comment_Votes* are typically used for comments with a user id and comment votes. This inbuilt classes are not used in PMS.

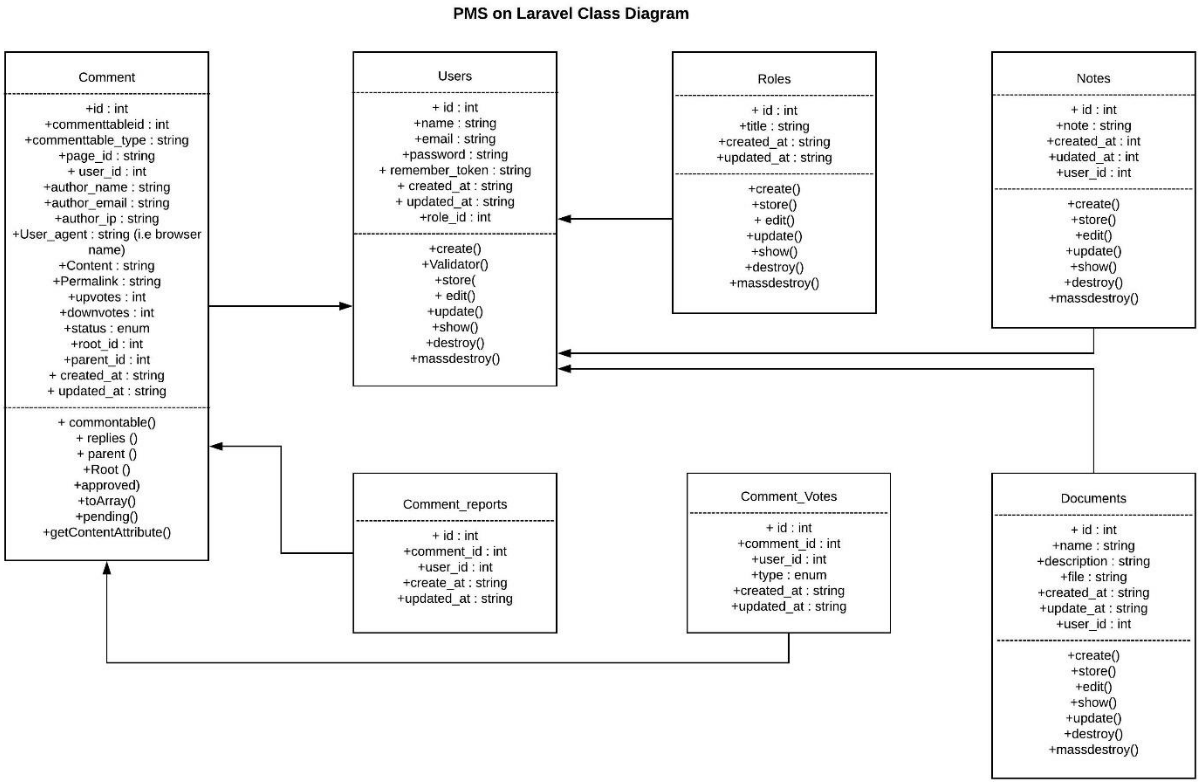
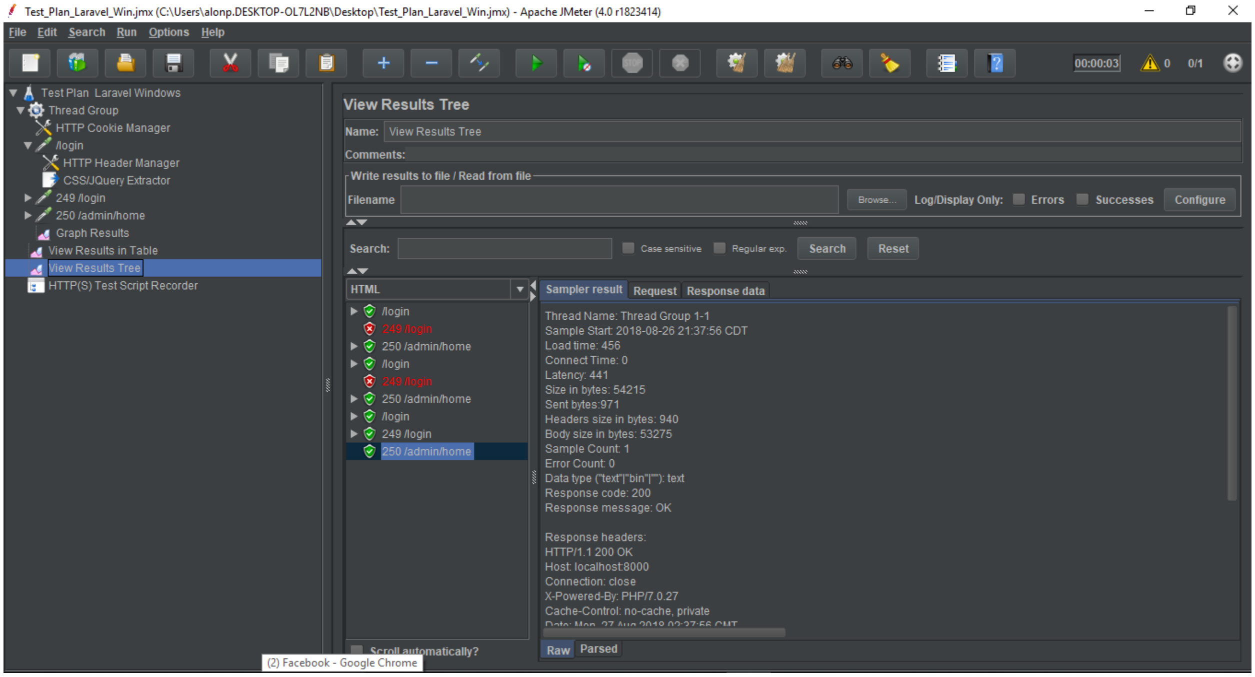


Figure 28. Class Diagram of PMS in Laravel on Windows and Ubuntu

Appendix F: Testing of PMS with JMeter

Apache *JMeter* is used to measure application performance of PMS. Data of page load time was obtained through *JMeter*. Pages considered during testing are */login, /admin/home*, */admin/pms_documents*, and */admin/roles*. These can be seen under sub-heading, *"Thread Group"*. Results can be viewed in sub-heading "*View Results in Table*" and "*View Results Tree*". The *HTTP(S) Test Script Recorder* creates a proxy server that acts as an intermediary between the browser and the application, retrieves the pages when the authentication page is accessed via the browser. Using *CSS/JQuery Extractor*, the retrieved authentication page is modified to mimic actual authentication to access the pages. *HTTP Cookie Manager* is also added since authentication page requires a cookie to store authentication details as a session.



- Test Plan Laravel Windows
  - Thread Group
    - HTTP Cookie Manager
    - /login
      - HTTP Header Manager
      - CSS/JQuery Extractor
    - 249 /login
    - 250 /admin/home
    - Graph Results
  - View Results in Table
  - View Results Tree
  - HTTP(S) Test Script Recorder

Name: View Results Tree

- /login
- 249 /login
- 250 /admin/home
- /login
- 249 /login
- 250 /admin/home
- /login
- 249 /login
- 250 /admin/home

```
Thread Name: Thread Group 1-1
Sample Start: 2018-08-26 21:37:56 CDT
Load time: 456
Connect Time: 0
Latency: 441
Size in bytes: 54215
Sent bytes:971
Headers size in bytes: 940
Body size in bytes: 53275
Sample Count: 1
Error Count: 0
Data type ("text"|"bin"|""): text
Response code: 200
Response message: OK

Response headers:
HTTP/1.1 200 OK
Host: localhost:8000
Connection: close
X-Powered-By: PHP/7.0.27
Cache-Control: no-cache, private
Date: Mon, 27 Aug 2018 02:37:56 GMT
```

Figure 29. Testing of PMS with *JMeter*

Appendix F: Codes of PMS in ASP.Net Core

Program.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace dotnetcorepms
{
    public class Program
    {
        public static void Main(string[] args)
        {
            //CreateWebHostBuilder(args).Build().Run();
            var host = new WebHostBuilder()
                .UseKestrel()
                .UseContentRoot(Directory.GetCurrentDirectory())
                .UseIISIntegration()
                .UseStartup<Startup>()
                .UseApplicationInsights()
                .Build();

            host.Run();
        }

        //public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        //    WebHost.CreateDefaultBuilder(args)
        //        .UseStartup<Startup>();
    }
}
```

Startup.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using dotnetcorepms.Filters;
using dotnetcorepms.Interfaces;
using dotnetcorepms.Repositories;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Http.Features;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;

namespace dotnetcorepms
{
    public class Startup
    {
        public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true);

            //if (env.IsDevelopment())
            //{
            //    // For more details on using the user secret store see 
http://go.microsoft.com/fwlink/?LinkID=532709
            //    builder.AddUserSecrets<Startup>();
            //}

            builder.AddEnvironmentVariables();
            Configuration = builder.Build();
        }

        public IConfigurationRoot Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the 
container.
        // For more information on how to configure your application, visit 
https://go.microsoft.com/fwlink/?LinkID=398940
        public void ConfigureServices(IServiceCollection services)
        {
            // Add framework services.
            services.AddMvc(options =>
            {
                options.Filters.Add(new CustomExceptionFilterAttribute());
                options.ReturnHttpNotAcceptable = true;
                // options.OutputFormatters = xml
            })
            .AddJsonOptions(options =>
            {
                //For Maintaining Json Format
                options.SerializerSettings.ContractResolver = new 
Newtonsoft.Json.Serialization.DefaultContractResolver();
            });

            // For FileUpload
            services.Configure<FormOptions>(x =>
            {
                x.ValueLengthLimit = int.MaxValue;
                x.MultipartBodyLengthLimit = int.MaxValue; // In case of multipart
                x.ValueLengthLimit = int.MaxValue; //not recommended value
                x.MemoryBufferThreshold = Int32.MaxValue;
            });

            // For Setting Session Timeout
            services.AddSession(options => {
                options.IdleTimeout = TimeSpan.FromMinutes(30);
            });

            //Getting Connection String From Database
            var connection = Configuration.GetConnectionString("DatabaseConnection");
            // UseRowNumberForPaging for Using Skip and Take in .Net Core
            services.AddDbContext<DatabaseContext>(options => options.UseSqlServer(connection,
b => b.UseRowNumberForPaging()));

            services.AddTransient<IUsers, UsersRepo>();
            services.AddTransient<IRoles, RolesRepo>();
            services.AddTransient<ILogin, LoginRepo>();
            services.AddTransient<IForum, ForumRepo>();
            services.AddTransient<IDocuments, DocumentsRepo>();
            services.AddTransient<INotes, NotesRepo>();
            services.AddTransient<ICommon, CommonRepo>();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory
loggerFactory)
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            if (env.IsDevelopment())
            {
                //app.UseDeveloperExceptionPage();
                //app.UseBrowserLink();
                app.UseExceptionHandler("/Error");
            }
            else
            {
                app.UseExceptionHandler("/Error");
            }

            // Using Static Files
            app.UseStaticFiles();
            // Enabling Session
            app.UseSession();
            // Routing
            app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=Login}/{action=Login}/{id?}");
            });
        }
    }
}
```

bower.json

```
{
  "name": "asp.net",
  "private": true,
  "dependencies": {
    "bootstrap": "3.3.6",
    "jquery-validation": "1.15.0",
    "jquery-validation-unobtrusive": "3.2.6",
    "datatables": "1.10.11",
    "fastclick": "1.0.6",
    "jquery": "2.2.3",
    "font-awesome": "4.6.1",
    "PACE": "pace#^1.0.2",
    "Ionicons": "ionicons#^2.0.1"
  },
  "resolutions": {
    "jquery": ">= 2.1.0",
    "jquery-validation": "1.15.0",
    "mocha": "2.2.3"
  }
}
```

Common/ ModuleHelper.cs

```
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;

namespace dotnetcorepms.Common
{
    /// <summary>
    /// This is where you customize the navigation sidebar
    /// </summary>
    public static class ModuleHelper
    {
        public enum Module
        {
            Dashboard,
            UserManagement,
            Members,
            Forum,
            ChangePassword,
            Logout,
            Roles,
            Users,
            Notes,
            Documents
        }

        public static SidebarMenu AddHeader(string name)
        {
            return new SidebarMenu
            {
                Type = SidebarMenuType.Header,
                Name = name,
            };
        }

        public static SidebarMenu AddTree(string name, string iconClassName = "fa fa-link")
        {
            return new SidebarMenu
            {
                Type = SidebarMenuType.Tree,
                IsActive = false,
                Name = name,
                IconClassName = iconClassName,
                URLPath = "#",
            };
        }
        public static SidebarMenu AddModule(Module module, Tuple<int, int, int> counter =
null)
        {
            if (counter == null)
                counter = Tuple.Create(0, 0, 0);

            switch (module)
            {
                case Module.Dashboard:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "Dashboard",
                        IconClassName = "fa fa-dashboard",
                        URLPath = "/Admin/Dashboard",
                        LinkCounter = counter,
                    };
                case Module.UserManagement:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "User management",
                        IconClassName = "fa fa-users",
                        URLPath = "#",
                        LinkCounter = counter,
                    };
                case Module.Roles:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "Roles",
                        IconClassName = "fa fa-briefcase",
                        URLPath = "/Roles/Index",
                        LinkCounter = counter,
                    };
                case Module.Users:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "Users",
                        IconClassName = "fa fa-user",
                        URLPath = "/Users/Index",
                        LinkCounter = counter,
                    };
                case Module.Members:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "Members",
                        IconClassName = "fa fa-briefcase",
                        URLPath = "#",
                        LinkCounter = counter,
                    };
                case Module.Notes:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "Notes",
                        IconClassName = "fa fa-building-o",
                        URLPath = "/Notes/Index",
                        LinkCounter = counter,
                    };
                case Module.Documents:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "Documents",
                        IconClassName = "fa fa-file",
                        URLPath = "/Documents/Index",
                        LinkCounter = counter,
                    };
                case Module.Forum:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "Forum",
                        IconClassName = "fa fa-group",
                        URLPath = "/Forum/Index",
                        LinkCounter = counter,
                    };
                case Module.ChangePassword:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "Change Password",
                        IconClassName = "fa fa-lock",
                        URLPath = "/Login/ChangePassword",
                        LinkCounter = counter,
                    };
                case Module.Logout:
                    return new SidebarMenu
                    {
                        Type = SidebarMenuType.Link,
                        Name = "Logout",
                        IconClassName = "fa fa-sign-out",
                        URLPath = "/Login/LogOut",
                        LinkCounter = counter,
                    };
                default:
                    break;
            }

            return null;
        }
    }
}
```

Controllers/AdminController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using dotnetcorepms.Filters;
using dotnetcorepms.Models;
using Microsoft.AspNetCore.Http;
```

```
namespace dotnetcorepms.Controllers
{
    //[ValidateAdminSession]
    public class AdminController : BaseController
    {
        public IActionResult Dashboard()
        {
            return View(new UsersViewModelLst { sessionUsername =
HttpContext.Session.GetString("FullName"), });
        }
    }
}
```

Controllers/ BaseController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using dotnetcorepms.Models;
using Microsoft.AspNetCore.Mvc;

namespace dotnetcorepms.Controllers
{
    public class BaseController : Controller
    {
        public enum TOSTRMSGTYPE
        {
            SUCCESS,
            ERROR,
            WARNING,
            OTHER
        };

        public enum ROLE
        {
            SUPER_USER = 1
        }

        public PairModel GetToStrJsonData(TOSTRMSGTYPE MsgType, string msg)
        {
            switch (MsgType)
            {
                case TOSTRMSGTYPE.SUCCESS:
                    return new PairModel { Key = "Success", Value = msg };
                case TOSTRMSGTYPE.WARNING:
                    return new PairModel { Key = "Warning", Value = msg };
                case TOSTRMSGTYPE.ERROR:
                    return new PairModel { Key = "Error", Value = msg };
                default:
                    return new PairModel { Key = "Info", Value = msg };
            }
        }
    }
}
```

Controllers/ DocumentsController.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using dotnetcorepms.Filters;
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;

namespace dotnetcorepms.Controllers
{
    //[ValidateUserSession]
    public class DocumentsController : BaseController
    {
        private readonly IHostingEnvironment _env;
        IDocuments _IDocuments;
        ICommon _ICommon;

        public DocumentsController(IDocuments IDocuments, ICommon ICommon, IHostingEnvironment 
hostingEnvironment)
        {
            _IDocuments = IDocuments;
            _ICommon = ICommon;
            _env = hostingEnvironment;
        }

        [HttpGet]
        public IActionResult Index()
        {
            var repo = _IDocuments.getAllDocuments();
            return View(new DocumentsViewModelLst
            {
                dbModelLst = repo,
                ddlUser = _ICommon.GetPairModel("Users")
            });
        }

        [HttpGet]
        public IActionResult Create()
        {
            return View(new DoucumentsViewModel {
                ddlUsers = _ICommon.GetPairModelWithDefault("Users"),
            });
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(DoucumentsViewModel documents)
        {
            try
            {
                var docNameExists = 
_IDocuments.checkDocumentnameExists(documents.dbModel.name);
                if (docNameExists)
                {
                    TempData["MessageRegistration"] = "Document Name already exists.";
                }
                else
                {
                    var uploadFileName = string.Empty;

                    if (HttpContext.Request.Form.Files != null)
                    {
                        var fileName = string.Empty;
                        string PathDB = string.Empty;

                        var files = HttpContext.Request.Form.Files;

                        if (files == null)
                        {
                            ModelState.AddModelError("", "Upload a document");
                            return View(documents.dbModel);
                        }

                        if (!ModelState.IsValid)
                        {
                            return View(documents.dbModel);
                        }

                        var uploads = Path.Combine(_env.WebRootPath, "files");
                        foreach (var file in files)
                        {
                            if (file.Length > 0)
                            {
                                fileName = 
ContentDispositionHeaderValue.Parse(file.ContentDisposition).FileName.Trim('"');

                                if (fileName.EndsWith(".txt") || fileName.EndsWith(".pdf") || 
fileName.EndsWith(".docx") || fileName.EndsWith(".doc"))
                                {
                                    var myUniqueFileName = Convert.ToString(Guid.NewGuid());
                                    var FileExtension = Path.GetExtension(fileName);
                                    uploadFileName = myUniqueFileName + FileExtension;
                                    fileName = Path.Combine(_env.WebRootPath, "files") + 
$@"\{uploadFileName}";
                                    PathDB = uploadFileName;
                                    using (FileStream fs = System.IO.File.Create(fileName))
                                    {
                                        file.CopyTo(fs);
                                        fs.Flush();
                                    }
                                }
                                else
                                {
                                    ModelState.AddModelError("", "Invalid document format. 
Please upload a file with extension .pdf, .txt, .docx or .doc");
                                    return View(documents.dbModel);
                                }
                            }
                        }
                        documents.dbModel.file = PathDB;
                        documents.dbModel.created_at = documents.dbModel.updated_at =
DateTime.Now;
                        if (_IDocuments.Commit(documents.dbModel, 0) > 0)
                        {
                            TempData["MessageRegistration"] = "Document Name successfully
added.";
                            //return View(documents.dbModel);
                            return RedirectToAction("Index");
                        }
                        else
                        {
                            return View(documents.dbModel);
                        }
                    }
                }
                return View(documents.dbModel);
            }
            catch (Exception)
            {

                throw;
            }
        }

        [HttpGet]
        public IActionResult Delete(int id)
        {
            var result = _IDocuments.getDocument(id);
            return View(new DoucumentsViewModel
            {
                dbModel = result,
                ddlUsers = _ICommon.GetPairModelWithDefault("Users"),
            });
        }

        [HttpGet]
        public IActionResult View(int id)
        {
            var result = _IDocuments.getDocument(id);
            return View(new DoucumentsViewModel
            {
                dbModel = result,
                ddlUsers = _ICommon.GetPairModelWithDefault("Users"),
            });
        }


        [HttpGet]
        public IActionResult Edit(int id)
        {
            var result = _IDocuments.getDocument(id);
            return View(new DoucumentsViewModel
            {
                dbModel = result,
                ddlUsers = _ICommon.GetPairModelWithDefault("Users"),
            });
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Edit(DoucumentsViewModel entity)
        {
            try
            {

                if (_IDocuments.Commit(entity.dbModel, 1) > 0)
                {
                    TempData["MessageRegistration"] = "Document updated successfully!";
                    //return View(entity.dbModel);
                    return RedirectToAction("Index");
                }
                else
                {
                    return RedirectToAction("Index");
                }
            }
            catch (System.Exception)
            {
                throw;
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Delete(DoucumentsViewModel entity)
        {
            try
            {

                if (_IDocuments.Commit(entity.dbModel, 2) > 0)
                {
                    TempData["MessageRegistration"] = "Document deleted successfully!";
                    //return View(entity.dbModel);
                    return RedirectToAction("Index");
                }
                else
                {
                    return RedirectToAction("Index");
                }
            }
            catch (System.Exception)
            {
                throw;
            }
        }
    }
}
```

Controllers/ ErrorController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```
// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860

namespace dotnetcorepms.Controllers
{
    public class ErrorController : BaseController
    {
        // GET: /<controller>/
        public IActionResult Error()
        {
            HttpContext.Session.Clear();
            return View("Error");
        }
    }
}
```

Controllers/ ForumController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using dotnetcorepms.Repositories;
using Microsoft.AspNetCore.Mvc;

namespace dotnetcorepms.Controllers
{
    public class ForumController : BaseController
    {
        IForum _IForum;

        public ForumController(IForum IForum)
        {
            _IForum = IForum;
        }

        public IActionResult Index()
        {
            return View();
        }

        [HttpGet]
        public JsonResult GetComments_Json()
        {
            var model = _IForum.GetComments();
            return Json(model);
        }

        [HttpPost]
        public ActionResult PostComments_Json(Forums entity)
        {
            return CRUD(entity, 0);
        }

        [HttpPost]
        public ActionResult PutComment_Json(int id, Forums entity)
        {
            return CRUD(entity, 1);
        }

        [HttpPost]
        public ActionResult DeleteComment_Json(int id, Forums entity)
        {
            return CRUD(entity, 2);
        }

        [NonAction]
        private JsonResult CRUD(Forums entity, int actionType)
        {
            int identity = 0;
            if (actionType == 2)
            {
                _IForum.Commit(entity, actionType);
                return Json(new { success = true, responseText = "Your message has been
deleted!" });
            }
            else
            {
                try
                {
                    identity = _IForum.Commit(entity, actionType == 3 ? 0 : actionType);
                    var model = _IForum.GetCommentsByID(identity);
                    return Json(model);
                }
                catch (Exception)
                {
                    return Json(new { success = false, responseText = "Your message couldn't
be sent!" });
                }
            }
        }
    }
}
```

Controllers/ LoginController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using dotnetcorepms.Library;

// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860

namespace dotnetcorepms.Controllers
{
    public class LoginController : BaseController
    {
        ILogin _ILogin;
        IUsers _IUsers;
        public LoginController(ILogin ILogin, IUsers IUsers)
        {
            _ILogin = ILogin;
            _IUsers = IUsers;
        }

        [HttpGet]
        public ActionResult Login()
        {
            return PartialView();
        }

        [HttpPost]
        //[ValidateAntiForgeryToken]
        public ActionResult Login(LoginViewModel loginViewModel)
        {
            try
            {
                if (!string.IsNullOrEmpty(loginViewModel.Email) &&
!string.IsNullOrEmpty(loginViewModel.Password))
                {
                    var email = loginViewModel.Email;
                    var password = EncryptionLibrary.EncryptText(loginViewModel.Password);

                    var result = _ILogin.ValidateUser(email, password);

                    if (result != null)
                    {
                        if (result.ID == 0 || result.ID < 0)
                        {
                            ViewBag.errormessage = "Entered Invalid Username and Password";
                        }
                        else
                        {
                            var RoleID = result.RoleID;
                            remove_Anonymous_Cookies(); //Remove Anonymous_Cookies
                            HttpContext.Session.SetString("UserID",
Convert.ToString(result.ID));
                            HttpContext.Session.SetString("RoleID",
Convert.ToString(result.RoleID));
                            HttpContext.Session.SetString("Email",
Convert.ToString(result.EmailID));
                            HttpContext.Session.SetString("FullName",
Convert.ToString(result.Name));
                            if (RoleID == 1)
                            {
                                return RedirectToAction("Dashboard", "Admin");
                            }
                            else if (RoleID == 2)
                            {
                                return RedirectToAction("Dashboard", "Admin");
                            }
                            else if (RoleID == 3)
                            {
                                return RedirectToAction("Dashboard", "SuperAdmin");
                            }
                        }
                    }
                    else
                    {
                        ViewBag.errormessage = "Invalid Email or Password";
                        return PartialView();
                    }
                }
                return PartialView();
            }
            catch (Exception)
            {
                throw;
            }
        }

        [HttpGet]
        public ActionResult Logout()
        {
            try
            {
                CookieOptions option = new CookieOptions();

                if (Request.Cookies["EventChannel"] != null)
                {
                    option.Expires = DateTime.Now.AddDays(-1);
                    Response.Cookies.Append("EventChannel", "", option);
                }

                HttpContext.Session.Clear();
                return RedirectToAction("Login", "Login");
            }
            catch (Exception)
            {
                throw;
            }

        }

        [NonAction]
        public void remove_Anonymous_Cookies()
        {
            if (Request.Cookies["EventChannel"] != null)
            {
                CookieOptions option = new CookieOptions();
                option.Expires = DateTime.Now.AddDays(-1);
                Response.Cookies.Append("EventChannel", "", option);
            }
        }

        [HttpGet]
        public IActionResult ChangePassword()
        {
            return View(new ChangePasswordModel());
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult ChangePassword(ChangePasswordModel ChangePasswordModel)
```

```
        {
            if (!ModelState.IsValid)
            {
                return View(ChangePasswordModel);
            }

            var password = EncryptionLibrary.EncryptText(ChangePasswordModel.Password);
            var registrationModel =
_IUsers.Userinformation(Convert.ToInt32(HttpContext.Session.GetString("UserID")));

            if (registrationModel.Password == password)
            {
                var registration = new Users();
                registration.Password =
EncryptionLibrary.EncryptText(ChangePasswordModel.NewPassword);
                registration.ID = Convert.ToInt32(HttpContext.Session.GetString("UserID"));
                var result = _ILogin.UpdatePassword(registration);

                if (result)
                {
                    TempData["MessageUpdate"] = "Password Updated Successfully";
                    ModelState.Clear();
                    return View(new ChangePasswordModel());
                }
                else
                {
                    return View(ChangePasswordModel);
                }
            }
            else
            {
                TempData["MessageUpdate"] = "Invalid Password";
                return View(ChangePasswordModel);
            }
        }
    }
}
```

Controllers/ NormalController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;

// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860

namespace dotnetcorepms.Controllers
{
    public class SuperAdminController : BaseController
    {
        // GET: /<controller>/
        public IActionResult Dashboard()
        {
            return View();
        }
    }
}
```

Controllers/ NotesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using dotnetcorepms.Filters;
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using Microsoft.AspNetCore.Mvc;

namespace dotnetcorepms.Controllers
{
    //[ValidateUserSession]
    public class NotesController : BaseController
    {
        INotes _INotes;
        ICommon _ICommon;

        public NotesController(INotes INotes, ICommon ICommon)
        {
            _INotes = INotes;
            _ICommon = ICommon;
        }

        [HttpGet]
        public IActionResult Index()
        {
            var repo = _INotes.getAllNotes();
            return View(new NotesViewModelLst
            {
                dbModelLst = repo,
                ddlUsers = _ICommon.GetPairModel("Users")
            });
        }


        [HttpGet]
        public IActionResult Create()
        {
            return View(new NotesViewModel {
               ddlUsers = _ICommon.GetPairModelWithDefault("Users"),
            });
        }


        [HttpGet]
        public IActionResult Edit(int id)
        {
            var result = _INotes.getNote(id);
            return View(new NotesViewModel
            {
                dbModel = result,
                ddlUsers = _ICommon.GetPairModelWithDefault("Users"),
            });
        }
```

```
[HttpGet]
public IActionResult View(int id)
{
    var result = _INotes.getNote(id);
    return View(new NotesViewModel
    {
        dbModel = result,
        ddlUsers = _ICommon.GetPairModelWithDefault("Users"),
    });
}

[HttpGet]
public IActionResult Delete(int id)
{
    var result = _INotes.getNote(id);
    return View(new NotesViewModel
    {
        dbModel = result,
        ddlUsers = _ICommon.GetPairModelWithDefault("Users"),
    });
}

[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create(NotesViewModel notes)
{
    try
    {
        var noteNameExists = _INotes.checkNotenameExists(notes.dbModel.note);
        if (noteNameExists)
        {
            TempData["MessageRegistration"] = "Note Name already exists.";
        }
        else
        {
            notes.dbModel.created_at = notes.dbModel.updated_at = DateTime.Now;
            if (_INotes.Commit(notes.dbModel, 0) > 0)
            {
                TempData["MessageRegistration"] = "Note Name successfully added.";
                return RedirectToAction("Index");
            }
            else
            {
                return View(notes.dbModel);
            }
        }
        return View(notes.dbModel);
    }
    catch (Exception)
    {
        throw;
    }
}

[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Edit(NotesViewModel entity)
```

```
        {
            try
            {

                if (_INotes.Commit(entity.dbModel, 1) > 0)
                {
                    TempData["MessageRegistration"] = "Note updated successfully!";
                    //return View(entity.dbModel);
                    return RedirectToAction("Index");
                }
                else
                {
                    return RedirectToAction("Index");
                }
            }
            catch (System.Exception)
            {
                throw;
            }
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Delete(NotesViewModel entity)
        {
            try
            {

                if (_INotes.Commit(entity.dbModel, 2) > 0)
                {
                    TempData["MessageRegistration"] = "Note deleted successfully!";
                    //return View(entity.dbModel);
                    return RedirectToAction("Index");
                }
                else
                {
                    return RedirectToAction("Index");
                }
            }
            catch (System.Exception)
            {
                throw;
            }
        }
    }
}
```

Controllers/ RolesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using dotnetcorepms.Filters;
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using Microsoft.AspNetCore.Mvc;
```

```
namespace dotnetcorepms.Controllers
{
    //[ValidateUserSession]
    public class RolesController : BaseController
    {
        IRoles _IRoles;
        ICommon _ICommon;

        public RolesController(IRoles IRoles, ICommon ICommon)
        {
            _IRoles = IRoles;
            _ICommon = ICommon;
        }

        [HttpGet]
        public IActionResult Index()
        {
            var repo = _IRoles.getAllRoles();
            return View(new RolesViewModelLst
            {
                dbModelLst = repo
            });
        }

        [HttpGet]
        public IActionResult Create()
        {
            return View();
        }

        [HttpGet]
        public IActionResult Delete(int id)
        {
            var result = _IRoles.getRole(id);
            return View(new RolesViewModel
            {
                dbModel = result,
            });
        }

        [HttpGet]
        public IActionResult Edit(int id)
        {
            var result = _IRoles.getRole(id);
            return View(new RolesViewModel
            {
                dbModel = result,
            });
        }

        [HttpGet]
        public IActionResult View(int id)
        {
            var result = _IRoles.getRole(id);
            return View(new RolesViewModel
            {
                dbModel = result,
            });
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(Roles roles)
        {
            try
            {
                var roleNameExists = _IRoles.checkRolenameExists(roles.Rolename);
                if (roleNameExists)
                {
                    TempData["MessageRegistration"] = "Role Name already exists.";
                }
                else
                {
                    roles.Created_At = DateTime.Now;
                    roles.Updated_At = DateTime.Now;
                    if (_IRoles.addRole(roles) > 0)
                    {
                        TempData["MessageRegistration"] = "Role Name successfully added.";
                        return View(roles);
                    }
                    else
                    {
                        return View(roles);
                    }
                }
                return View(roles);
            }
            catch (Exception)
            {

                throw;
            }
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Edit(RolesViewModel entity)
        {
            try
            {

                if (_IRoles.Commit(entity.dbModel, 1) > 0)
                {
                    TempData["MessageRegistration"] = "Role updated successfully!";
                    //return View(entity.dbModel);
                    return RedirectToAction("Index");
                }
                else
                {
                    return RedirectToAction("Index");
                }
            }
            catch (System.Exception)
            {
```

```
                throw;
            }
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Delete(RolesViewModel entity)
        {
            try
            {

                if (_IRoles.Commit(entity.dbModel, 2) > 0)
                {
                    TempData["MessageRegistration"] = "Role deleted successfully!";
                    //return View(entity.dbModel);
                    return RedirectToAction("Index");
                }
                else
                {
                    return RedirectToAction("Index");
                }
            }
            catch (System.Exception)
            {
                throw;
            }
        }
    }
}
```

Controllers/ UserProfileController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using dotnetcorepms.Interfaces;
using Microsoft.AspNetCore.Http;
using dotnetcorepms.Filters;

// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860

namespace dotnetcorepms.Controllers
{
    [ValidateUserSession]
    public class UserProfileController : BaseController
    {
        IUsers _IRepository;
        public UserProfileController(IUsers IRepository)
        {
            _IRepository = IRepository;
        }

        [HttpGet]
        public IActionResult Profile()
        {
            try
            {
                var profile =
_IRepository.Userinformation(Convert.ToInt32(HttpContext.Session.GetString("UserID")));
                return View(profile);
            }
            catch (Exception)
            {
                throw;
            }
        }

    }
}
```

Controllers/ UsersController.cs

```
using dotnetcorepms.Filters;
using dotnetcorepms.Interfaces;
using dotnetcorepms.Library;
using dotnetcorepms.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;

namespace dotnetcorepms.Controllers
{
    [ValidateAdminSession]
    public class UsersController : BaseController
    {

        IUsers _IRepository;
        IRoles _IRoles;
        ICommon _ICommon;

        public UsersController(IUsers IRepository, IRoles IRoles, ICommon ICommon)
        {
            _IRepository = IRepository;
            _IRoles = IRoles;
            _ICommon = ICommon;
        }

        [HttpGet]
        public IActionResult Index()
        {
            var repo = _IRepository.getAllUsers();
            return View(new UsersViewModelLst
            {
                dbModelLst = repo,
                ddlRoleLst = _ICommon.GetPairModel("RoleName"),
            });
        }

        [HttpGet]
        public IActionResult Edit(int id)
        {
            var result = _IRepository.Userinformation(id);
            return View(new UsersViewModel
            {
                dbModel = result,
                ddlRoleLst = _ICommon.GetPairModelWithDefault("RoleName"),
            });
        }

        [HttpGet]
        public IActionResult Create()
        {
            return View(new UsersViewModel {
                ddlRoleLst = _ICommon.GetPairModelWithDefault("RoleName"),
            });
        }

        [HttpGet]
        public IActionResult Delete(int id)
        {
            var result = _IRepository.Userinformation(id);
            return View(new UsersViewModel
            {
                dbModel = result,
                ddlRoleLst = _ICommon.GetPairModelWithDefault("RoleName"),
            });
        }

        [HttpGet]
        public IActionResult View(int id)
        {
            var result = _IRepository.Userinformation(id);
            return View(new UsersViewModel
            {
                dbModel = result,
                ddlRoleLst = _ICommon.GetPairModelWithDefault("RoleName"),
            });
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(UsersViewModel entity)
        {
            try
            {
                var isUsernameExists =
_IRepository.CheckUserNameExists(entity.dbModel.Username);

                if (isUsernameExists)
                {
                    ModelState.AddModelError("", errorMessage: "Username already exists.
Please enter a unique username.");
                }
                else
                {
                    entity.dbModel.CreatedOn = DateTime.Now;
                    entity.dbModel.RoleID = entity.dbModel.RoleID;
                    entity.dbModel.Password =
EncryptionLibrary.EncryptText(entity.dbModel.Password);
                    entity.dbModel.ConfirmPassword =
EncryptionLibrary.EncryptText(entity.dbModel.Password);
```

```
            if (_IRepository.AddUser(entity.dbModel) > 0)
            {
                TempData["MessageRegistration"] = "User created successfully!";
                // return View(entity.dbModel);
                return RedirectToAction("Index");
            }
            else
            {
                return RedirectToAction("Index");
            }
        }

        return RedirectToAction("Index");
    }
    catch (System.Exception)
    {
        throw;
    }
}

[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Edit(UsersViewModel entity)
{
    try
    {

        if (_IRepository.Commit(entity.dbModel, 1) > 0)
        {
            TempData["MessageRegistration"] = "User updated successfully!";
            //return View(entity.dbModel);
            return RedirectToAction("Index");
        }
        else
        {
            return RedirectToAction("Index");
        }
    }
    catch (System.Exception)
    {
        throw;
    }
}

[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Delete(UsersViewModel entity)
{
    try
    {

        if (_IRepository.Commit(entity.dbModel, 2) > 0)
        {
            TempData["MessageRegistration"] = "User deleted successfully!";
            //return View(entity.dbModel);
            return RedirectToAction("Index");
        }
        else
```

```
                {
                    return RedirectToAction("Index");
                }
            }
            catch (System.Exception)
            {
                throw;
            }
        }

        public JsonResult CheckUserNameExists(string Username)
        {
            try
            {
                var isUsernameExists = _IRepository.CheckUserNameExists(Username);
                if (isUsernameExists)
                {
                    return Json(data: true);
                }
                else
                {
                    return Json(data: false);
                }
            }
            catch (System.Exception)
            {
                throw;
            }
        }
    }
}
```

Model/ ChangePasswordModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dotnetcorepms.Models
{
    [NotMapped]
    public class ChangePasswordModel
    {
        [MinLength(7, ErrorMessage = "Minimum Password must be 7 in charaters")]
        [Required(ErrorMessage = "Password Required")]
        public string Password { get; set; }

        [Compare("NewPassword", ErrorMessage = "Enter Valid Password")]
        public string ConfirmPassword { get; set; }

        [MinLength(7, ErrorMessage = "Minimum Password must be 7 in charaters")]
        [Required(ErrorMessage = "Password Required")]
        public string NewPassword { get; set; }
    }
}
```

Model/ DocumentsModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;

namespace dotnetcorepms.Models
{
    public partial class Documents
    {
        [Key]
        public int id { get; set; }
        public string name { get; set; }
        public string description { get; set; }
        public string file { get; set; }
        public DateTime? created_at { get; set; }
        public DateTime? updated_at { get; set; }
        public int? user_id { get; set; }
    }

    public class DoucumentsViewModel
    {
        public Documents dbModel { get; set; }
        public IEnumerable<PairModel> ddlUsers { get; set; }
    }

    public class DocumentsViewModelLst
    {
        public IQueryable<Documents> dbModelLst { get; set; }
        public IEnumerable<PairModel> ddlUser { get; set; }
    }
}
```

Model/ ForumModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Text;

namespace dotnetcorepms.Models
{
    public partial class Forums
    {
        [Key]
        public int ID { get; set; }
        public string content { get; set; }
        public bool? created_by_current_user { get; set; }
        public string fullname { get; set; }
        public DateTime? modified { get; set; }
        public int? parent { get; set; }
        public string profile_picture_url { get; set; }
        public int? upvote_count { get; set; }
        public bool? user_has_upvoted { get; set; }
        public DateTime? created { get; set; }
    }
}
```

Model/ GlobalModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Text;

namespace dotnetcorepms.Models
{
    public class MessageModel
    {
        public string title { get; set; }
        public string area { get; set; }
        public string controller { get; set; }
        public string action { get; set; }
    }

    public class PairModel
    {
        public object Key { get; set; }
        public object Value { get; set; }
    }

    public class CommonTwoModel
    {
        [Required]
        [Display(Name = "Created On")]
        public DateTime Created_At { get; set; }

        [Required]
        [Display(Name = "Updated On")]
        public DateTime Updated_At { get; set; }
    }
}
```

Model/ LoginViewModel.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace dotnetcorepms.Models
{
    [NotMapped]
    public class LoginViewModel
    {

        [Required(ErrorMessage = "Email Required")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Password Required")]
        public string Password { get; set; }

    }
}
```

Model/ Message.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace dotnetcorepms.Models
{
    public class Message
    {
        public int Id { get; set; }
        public string UserId { get; set; }
        public string DisplayName { get; set; }
        public string FontAwesomeIcon { get; set; }
        public string AvatarURL { get; set; }
        public string URLPath { get; set; }
        public string ShortDesc { get; set; }
        public string TimeSpan { get; set; }
        public int Percentage { get; set; }
        public string Type { get; set; }
    }
}
```

Model/ NotesModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;

namespace dotnetcorepms.Models
{
    public partial class Notes
    {
        [Key]
        public int id { get; set; }
        public string note { get; set; }
        public DateTime? created_at { get; set; }
        public DateTime? updated_at { get; set; }
        public int? user_id { get; set; }
    }

    public class NotesViewModel
    {
        public Notes dbModel { get; set; }
        public IEnumerable<PairModel> ddlUsers { get; set; }
    }

    public class NotesViewModelLst
    {
        public IQueryable<Notes> dbModelLst { get; set; }
        public IEnumerable<PairModel> ddlUsers { get; set; }
    }
}
```

Model/ RolesModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dotnetcorepms.Models
{
    public partial class Roles : CommonTwoModel
    {
        [Key]
        public int RoleID { get; set; }
        public string Rolename { get; set; }
    }

    public class RolesViewModel
    {
        public Roles dbModel { get; set; }
    }

    public class RolesViewModelLst
    {
        public IQueryable<Roles> dbModelLst { get; set; }
    }
}
```

Model/ SidebarMenu.cs

```
using System;
using System.Collections.Generic;

namespace dotnetcorepms.Models
{
    public class SidebarMenu
    {
        public SidebarMenuType Type { get; set; }
        public bool IsActive { get; set; } = false;
        public string Name { get; set; }
        public string IconClassName { get; set; }
        public string URLPath { get; set; }
        public List<SidebarMenu> TreeChild { get; set; }
        public Tuple<int, int, int> LinkCounter  { get; set; }
    }

    public enum SidebarMenuType
    {
        Header,
        Link,
        Tree
    }
}
```

Model/ UsersModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;

namespace dotnetcorepms.Models
{
    public partial class Users
    {
        [Key]
        public int ID { get; set; }

        [Required(ErrorMessage = "Name Required")]
        public string Name { get; set; }

        [Required(ErrorMessage = "Email Required")]
        [RegularExpression(@"^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$", ErrorMessage = "Please
enter a valid e-mail adress")]
        public string EmailID { get; set; }

        [MinLength(6, ErrorMessage = "Minimum Username must be 6 in charaters")]
        [Required(ErrorMessage = "Username Required")]
        public string Username { get; set; }

        [MinLength(7, ErrorMessage = "Minimum Password must be 7 in charaters")]
        [Required(ErrorMessage = "Password Required")]
        public string Password { get; set; }

        [Compare("Password", ErrorMessage = "Enter Valid Password")]
        public string ConfirmPassword { get; set; }

        public int? RoleID { get; set; }

        public DateTime? CreatedOn { get; set; }
    }

    public class UsersModel
    {
        [Key]
        public int ID { get; set; }
        public string Name { get; set; }
        public string EmailID { get; set; }
        public int? RoleID { get; set; }
        public string CreatedOn { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
    }

    public class UsersViewModel
    {
        public Users dbModel { get; set; }
        public IEnumerable<PairModel> ddlRoleLst { get; set; }
    }

    public class UsersViewModelLst
    {
        public IQueryable<UsersModel> dbModelLst { get; set; }
        public List<PairModel> ddlRoleLst { get; set; }
        public string sessionUsername { get; set; }
    }
}
```

Views/ Admin/ Dashboard.cshtml

```
@model dotnetcorepms.Models.UsersViewModelLst
@{
    Layout = "~/Views/Shared/_AdminLayout.cshtml";
}

<section class="content-header">
    <h1>
        Dashboard <small>Welcome, <b>@Model.sessionUsername</b> !</small>
    </h1>
    <ol class="breadcrumb">
        <li><a href="#"><i class="fa fa-dashboard"></i> Home</a></li>
        <li class="active">Dashboard</li>
    </ol>
</section>
```

- CreateAdminUser
    - Create.cshtml

```
@model dotnetcorepms.Models.Users
@{
    Layout = "~/Views/Shared/_SuperAdminLayout.cshtml";
}
<div class="container">
    <br />
    <div class="panel panel-default">
        <div class="panel-heading">Register</div>
        <div class="panel-body">

            @if (TempData["MessageRegistration"] != null)
            {
                <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
            }

            <form method="post" asp-controller="CreateAdminUser" asp-action="Create">
                @Html.AntiForgeryToken()
                @Html.ValidationSummary(true)
                <div class="row">
                    <div class="col-lg-4">
                        <label class="control-label" asp-for="Name">Name</label>
                        <input asp-for="Name" type="text" class="form-control" />
                        <span asp-validation-for="Name" class="text-danger"></span>
                    </div>
                </div>

                <div class="row">
                    <div class="col-lg-4">
                        <label class="control-label" asp-for="EmailID">EmailID</label>
                        <input asp-for="EmailID" type="text" class="form-control" />
                        <span asp-validation-for="EmailID" class="text-danger"></span>
                    </div>

                </div>


                <div class="row">
                    <div class="col-lg-4">
                        <label class="control-label" asp-for="Username">Username</label>
                        <input asp-for="Username" onblur="CheckUsernameExists();" type="text"
class="form-control" />
                        <span asp-validation-for="Username" class="text-danger"></span>
                    </div>
                    <div class="col-lg-4">
                        <label class="control-label" asp-for="Password">Password</label>
                        <input asp-for="Password" type="password" class="form-control" />
                        <span asp-validation-for="Password" class="text-danger"></span>
                    </div>
                    <div class="col-lg-4">
                        <label class="control-label" asp-
for="ConfirmPassword">ConfirmPassword</label>
                        <input asp-for="ConfirmPassword" type="password" class="form-control"
/>
                        <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
                    </div>
                </div>

                <div class="row">
                    <div class="col-lg-4">
                        <br />
                        <input id="Submit1" type="submit" class="btn btn-primary"
value="Register" />
                        <a class="btn btn-success" href="/CreateAdminUser/Create">Clear</a>
                    </div>


                </div>
            </form>


        </div>
    </div>

</div>
```

Views/ Customer/ ChangePassword.cshtml

```
@model dotnetcorepms.Models.ChangePasswordModel
@{
    Layout = "~/Views/Shared/_LayoutCustomer.cshtml";
}

<div class="container">
    <br />
    <div class="panel panel-default">
        <div class="panel-heading">Change Password</div>
        <div class="panel-body">
            @Html.AntiForgeryToken()
            @if (TempData["MessageUpdate"] != null)
            {
                <p class="alert alert-success"
id="successMessage">@TempData["MessageUpdate"]</p>
            }
            @Html.ValidationSummary(true)
            <form method="post" asp-controller="Customer" asp-action="ChangePassword">
                <div class="row">
                    <div class="col-lg-4">
```

```
                        <label class="control-label" asp-for="Password">Old Password</label>
                        <input asp-for="Password" type="password" class="form-control" />
                        <span asp-validation-for="Password" class="text-danger"></span>
                    </div>
                    <div class="col-lg-4">
                        <label class="control-label" asp-for="NewPassword">New
Password</label>
                        <input asp-for="NewPassword" type="password" class="form-control" />
                        <span asp-validation-for="NewPassword" class="text-danger"></span>
                    </div>
                    <div class="col-lg-4">
                    </div>
                </div>
                <div class="row">
                    <div class="col-lg-4">
                        <br />
                        <input id="Submit" type="submit" class="btn btn-primary" value="Change
Password" />
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>
```

Views/ Customer/ Dashboard.cshtml

```
@{
    Layout = "~/Views/Shared/_LayoutCustomer.cshtml";
}
```

Views/ Documents/ Create.cshtml

```
@model dotnetcorepms.Models.DoucumentsViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Documents</h3>
    <form method="post" enctype="multipart/form-data" asp-controller="Documents" asp-
action="Create">
        @Html.AntiForgeryToken()
        @Html.ValidationSummary(true)

        <div class="panel panel-default">
            <div class="panel-heading">
                Create
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.user_id">Member*</label>
                        <select asp-for="dbModel.user_id" class="form-control"
```

```
                                asp-items="@(new SelectList(Model.ddlUsers,"Key",
"Value",0))"></select>
                        <span asp-validation-for="dbModel.user_id" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.name">Title</label>
                        <input asp-for="dbModel.name" type="text" class="form-control" />
                        <span asp-validation-for="dbModel.name" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-
for="dbModel.description">Description</label>
                        <textarea asp-for="dbModel.description" rows="10" cols="40"
class="form-control"></textarea>
                    </div>
                </div>
                <div class="form-group">
                    <div class="controls">
                        <label class="control-label" asp-for="dbModel.file">File*</label>
                        <input required type="file" title="" name="dbModel.file" multiple />
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Save">
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Documents/ Delete.cshtml

```
@model dotnetcorepms.Models.DoucumentsViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Documents</h3>
    <form method="post" asp-controller="Documents" asp-action="Delete">
        @Html.HiddenFor(m => m.dbModel.id)
        @Html.HiddenFor(m => m.dbModel.created_at)
        @Html.AntiForgeryToken()
        <!-- CSRF protection to prevent phising / fake pages -->
        @Html.ValidationSummary(true)
        <div class="panel panel-default">
            <div class="panel-heading">
                Are you sure you want to Delete?
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
```

```
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.user_id">Member</label>
                        @Html.DropDownListFor(model => model.dbModel.user_id, new
SelectList(Model.ddlUsers, "Key", "Value"), new { @class = "form-control" })
                        <span asp-validation-for="dbModel.user_id" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.name">Title</label>
                        @Html.TextBoxFor(model => model.dbModel.name, new { @class = "form-
control" })

                        <span asp-validation-for="dbModel.name" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-
for="dbModel.description">Email*</label>
                        @Html.TextAreaFor(model => model.dbModel.description, new { @class =
"form-control", rows = 10, cols = 60 })
                        <span asp-validation-for="dbModel.description" class="text-
danger"></span>
                    </div>
                </div>
                <div class="form-group">
                    <div class="controls">
                        <label class="control-label" asp-for="dbModel.file">File*</label> @if
(Model.dbModel.file != string.Empty && Model.dbModel.file != null)
                        {<a href="~/files/@Model.dbModel.file" target="_blank">Download
file</a>}
                        <input type="file" title="" name="dbModel.file" multiple />
                    </div>
                </div>

            </div>
            </div>
            <input class="btn btn-danger" type="submit" value="Delete">
            @Html.ActionLink("Cancel", "Index", "Documents", null, new { @class = "btn btn-
danger" })
</form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Documents/ Edit.cshtml

```
@model dotnetcorepms.Models.DoucumentsViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Documents</h3>
    <form method="post" asp-controller="Documents" asp-action="Edit">
        @Html.HiddenFor(m => m.dbModel.id)
        @Html.HiddenFor(m => m.dbModel.created_at)
        @Html.AntiForgeryToken()
        <!-- CSRF protection to prevent phising / fake pages -->
        @Html.ValidationSummary(true)
        <div class="panel panel-default">
            <div class="panel-heading">
                Edit
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.user_id">Member</label>
                        @Html.DropDownListFor(model => model.dbModel.user_id, new
SelectList(Model.ddlUsers, "Key", "Value"), new { @class = "form-control" })
                        <span asp-validation-for="dbModel.user_id" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.name">Title</label>
                        @Html.TextBoxFor(model => model.dbModel.name, new { @class = "form-
control" })
                        <span asp-validation-for="dbModel.name" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-
for="dbModel.description">Email*</label>
                        @Html.TextAreaFor(model => model.dbModel.description, new { @class =
"form-control", rows = 10, cols = 60 })
                        <span asp-validation-for="dbModel.description" class="text-
danger"></span>
                    </div>
                </div>
                <div class="form-group">
                    <div class="controls">
                        <label class="control-label" asp-for="dbModel.file">File*</label> @if
(Model.dbModel.file != string.Empty && Model.dbModel.file != null) { <a
href="~/files/@Model.dbModel.file" target="_blank">Download file</a>}
                        <input required type="file" title="" name="dbModel.file" multiple />
                    </div>
                </div>

            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Update">
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Documents/ Index.cshtml

```
@model dotnetcorepms.Models.DocumentsViewModelLst
@{
    ViewData["Title"] = "Index";
}

<h3 class="page-title">Documents</h3>
<p>
    <a href="@Url.Action("Create","Documents")" class="btn btn-success">Add new</a>
</p>
<div class="panel panel-default">
    <div class="panel-heading">List</div>
    <div class="panel-body table-responsive">
        <table id="data-table" class="table table-striped table-bordered select">
            <thead>
                <tr>
                    <th><input name="select_all" value="1" type="checkbox"></th>
                    <th>Member</th>
                    <th>Title</th>
                    <th>Description</th>
                    <th>File</th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                @foreach (var item in Model.dbModelLst)
                {
                <tr id="@item.id">
                    <td id="@item.id">@item.id</td>
                    <td>@(Model.ddlUser.Where(x => int.Parse(x.Key.ToString()) ==
item.user_id).Select(x => x.Value).FirstOrDefault())</td>
                    <td>@item.name</td>
                    <td>@item.description</td>
                    <td>
                        @if (item.file != string.Empty && item.file != null)
                        {<a href="~/files/@item.file" target="_blank">Download file</a>}
                        else
                        { <b>No File</b>}
                </td>
                <td>
                    @Html.ActionLink("View", "View", "Documents", new { id = item.id }, new {
@class = "btn btn-xs btn-primary" })
                    @Html.ActionLink("Edit", "Edit", "Documents", new { id = item.id }, new {
@class = "btn btn-xs btn-info" })
                    @Html.ActionLink("Delete", "Delete", "Documents", new { id = item.id },
new { @class = "btn btn-xs btn-danger" })
                </td>
                </tr>
                }
            </tbody>
        </table>
    </div>
</div>
```

Views/ Documents/ View.cshtml

```
@model dotnetcorepms.Models.DoucumentsViewModel
@{
    ViewData["Title"] = "View";
}
```

```
<section class="content">

    <!-- <div class="row">
        <div class="col-md-12"> -->


    <h3 class="page-title">Documents</h3>

    <div class="panel panel-default">
        <div class="panel-heading">
            View
        </div>

        <div class="panel-body table-responsive">
            <div class="row">
                <div class="col-md-6">
                    <table class="table table-bordered table-striped">
                        <tbody>
                            <tr>
                                <th>Member</th>
                                <td field-key="role">@(Model.ddlUsers.Where(x =>
x.Key.ToString() == Model.dbModel.user_id.ToString()).Select(x =>
x.Value.ToString()).FirstOrDefault())</td>
                            </tr>
                            <tr>
                                <th>Title</th>
                                <td field-key="email">@(Model.dbModel.name)</td>
                            </tr>
                            <tr>
                                <th>Description</th>
                                <td field-key="email">@(Model.dbModel.description)</td>
                            </tr>
                            <tr>
                                <th>File</th>
                                <td>
                                    @if (Model.dbModel.file != string.Empty &&
Model.dbModel.file != null)
                                    {
                                        <a href="~/files/@(Model.dbModel.file)"
target="_blank">Download file</a>}
                                    else
                                    {
                                        <b>No File</b>}
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>

            <p>&nbsp;</p>
            @Html.ActionLink("Back to list", "Index", "Documents", null, new { @class = "btn
btn-default" })
        </div>
    </div>

    <!-- </div>
    </div> -->
```

```
</section>
```

Views/ Error / Error.cshtml

```
<style>
    .center {
        text-align: center;
        margin-left: auto;
        margin-right: auto;
        margin-bottom: auto;
        margin-top: auto;
    }
</style>
<div class="container">
    <div class="row">
        <div class="span12">
            <div class="hero-unit center">
                <h1>Something Went Wrong<small><font face="Tahoma"
color="red">Error</font></small></h1>
                <br />
                <p>The page you requested could not be found, either contact your webmaster or
try again. Use your browsers <b>Back</b> button to navigate to the page you have prevously
come from</p>
                <p><b>Or you could just press this neat little button:</b></p>
                <a href="/Admin/Dashboard" class="btn btn-large btn-info"><i class="icon-home
icon-white"></i> Take Me Home</a>
            </div>
            <br />
            <br />

            @if (ViewData["ErrorMessage"] != null)
            {
                <p class="alert alert-success"
id="successMessage">@ViewData["ErrorMessage"]</p>
            }
            <br />
            <p></p>
            <!-- By ConnerT HTML & CSS Enthusiast -->
        </div>
    </div>
</div>
```

Views/ Forum/ Index.cshtml

```
<section class="content-header">
    <h1>
        Forum
        <small>Members Discussion Forum</small>
    </h1>
    <ol class="breadcrumb">
        <li><a href="#"><i class="fa fa-dashboard"></i> Home</a></li>
        <li class="active">Forum</li>
    </ol>
</section>

<div class="pad margin no-print">
    <div class="callout callout-info" style="margin-bottom: 0!important;">
        <h4>
```

```
            <i class="fa fa-info-circle"></i> &nbsp;
            Welcome,
        </h4>
        to the discussion forum of project related topics with other PMS users.
    </div>
</div>

<!-- Main content -->
<section class="invoice">
                    <div class="row">
                        <div class="span4">
                            <script type="text/javascript">
                                    $(function () {
                                        $('#comments-container').comments({
                                            profilePictureURL:
'https://www.evatar.io/d2e8860457a4e16483876f0c837669ab',
                                            roundProfilePictures: true,
                                            textareaRows: 1,
                                            enableAttachments: false,
                                            enableUpvoting: false,
                                            enableNavigation: false,
                                            enableEditing: true,
                                            // youtText : $('#username').val(),
                                            getComments: function (success, error) {
                                                $.ajax({
                                                    type: 'get',
                                                    url: '/Forum/GetComments_Json',
                                                    success: function (commentsArray) {
                                                        success(commentsArray)
                                                    },
                                                    error: error
                                                });
                                            },
                                            postComment: function (commentJSON, success,
error) {
                                                $.ajax({
                                                    type: 'post',
                                                    url: '/Forum/PostComments_Json',
                                                    dataType: 'json',
                                                    contentType: "application/json;
charset=utf-8",
                                                    data: JSON.stringify(commentJSON),
                                                    success: setTimeout(function () {
                                                        success(commentJSON);
                                                        // console.log("LOGTRACE",
commentJSON);
                                                    }, 500),
                                                    error: error
                                                });
                                            },
                                            //putComment: function (data, success, error) {
                                            //    setTimeout(function () {
                                            //        success(data);
                                            //    }, 500);
                                            //},
                                            putComment: function (commentJSON, success, error)
{
                                                $.ajax({
                                                    type: 'post',
                                                    url: '/Forum/PutComment_Json/' +
commentJSON.id,
                                                    dataType: 'json',
                                                    contentType: "application/json;
charset=utf-8",
                                                    data: JSON.stringify(commentJSON),
                                                    success: setTimeout(function () {
                                                        success(commentJSON);
                                                    }, 500),
                                                    error: error
                                                });
                                            },
                                            //deleteComment: function (data, success, error) {
                                            //    setTimeout(function () {
                                            //        success();
                                            //    }, 500);
                                            //},
                                            deleteComment: function (commentJSON, success,
error) {
                                                $.ajax({
                                                    type: 'post',
                                                    url: '/Forum/DeleteComment_Json/' +
commentJSON.id,
                                                    success: success,
                                                    error: error
                                                });
                                            },
                                            upvoteComment: function (data, success, error) {
                                                setTimeout(function () {
                                                    success(data);
                                                }, 500);
                                            },
                                            uploadAttachments: function (dataArray, success,
error) {
                                                setTimeout(function () {
                                                    success(dataArray);
                                                }, 500);
                                            },
                                        });
                                    });
                                                            </script>
                            <div id="comments-container"></div>
                        </div>
                </div>
</section>
<!-- /.content -->
<div class="clearfix"></div>
```

Views/ Login/ ChangePassword.cshtml

```
@model dotnetcorepms.Models.ChangePasswordModel
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Change Password</h3>
    <form method="post" asp-controller="Login" asp-action="ChangePassword">
        <!-- If no success message in flash session show change password form  -->
        <div class="panel panel-default">
            <div class="panel-heading">
                Edit
            </div>
            <div class="panel-body">
                @Html.AntiForgeryToken()
                @if (TempData["MessageUpdate"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageUpdate"]</p>
                }
                @Html.ValidationSummary(true)
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="Password">Current
Password</label>
                        <input asp-for="Password" type="password" class="form-control" />
                        <span asp-validation-for="Password" class="text-danger"></span>
                    </div>
                </div>

                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="NewPassword">New
Password</label>
                        <input asp-for="NewPassword" type="password" class="form-control" />
                        <span asp-validation-for="NewPassword" class="text-danger"></span>
                    </div>
                </div>

                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="ConfirmPassword">New password
confirmation</label>
                        <input asp-for="ConfirmPassword" type="password" class="form-control"
/>
                        <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Save">
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Login/ Login.cshtml

```
@model dotnetcorepms.Models.LoginViewModel
<!DOCTYPE html>
<!--
This is a starter template page. Use this page to start your new project from
```

```
scratch. This page gets rid of all links and provides the needed markup only.
-->
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Project Management System</title>
    <!-- Tell the browser to be responsive to screen width -->
    <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no"
name="viewport">
    <environment names="Development">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
        <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.css" />
        <link rel="stylesheet" href="~/lib/Ionicons/css/ionicons.css" />
        <link rel="stylesheet" href="~/css/AdminLTE.css" />
        <link rel="stylesheet" href="~/css/skins/skin-blue.css" />
        <link rel="stylesheet" href="~/css/site.css" />
    </environment>
    <environment names="Staging,Production">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.min.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/lib/Ionicons/css/ionicons.min.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/css/AdminLTE.min.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/css/skins/skin-blue.min.css" asp-append-version="true"
/>
        <link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />
    </environment>
    <environment names="Development">
        <script src="~/lib/jquery/dist/jquery.js" asp-append-version="true"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.js" asp-append-version="true"></script>
        <script src="~/js/app.js" asp-append-version="true"></script>
        <script src="~/lib/PACE/pace.js" asp-append-version="true"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
    </environment>
    <environment names="Staging,Production">
        <script src="~/lib/jquery/dist/jquery.min.js" asp-append-version="true"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.min.js" asp-append-
version="true"></script>
        <script src="~/js/app.min.js" asp-append-version="true"></script>
        <script src="~/lib/PACE/pace.min.js" asp-append-version="true"></script>
        <script src="~/js/site.min.js" asp-append-version="true"></script>
    </environment>
</head>
<body>
    <div style="margin-top: 10%;"></div>
    <div class="container-fluid">
        <div class="row">
            <div class="col-md-8 col-md-offset-2">
                <div class="panel panel-default">
                    <div class="panel-heading">Project Management System Login</div>
                    <div class="panel-body">
                        @if (ViewBag.errormessage != null)
                        {
                            <p class="alert alert-danger"
id="successMessage">@ViewBag.errormessage</p>
```

```
                        }
                        <form asp-controller="Login" asp-action="Login" method="post" 
class="form-horizontal">

                            <div class="form-group">
                                <label class="col-md-4 control-label" asp-
for="Email">Email</label>
                                <div class="col-md-6">
                                    <input autocomplete="off" asp-for="Email" type="text" 
class="form-control" />
                                </div>
                                <span asp-validation-for="Email" class="text-danger"></span>
                            </div>
                            <div class="form-group">
                                <label class="col-md-4 control-label" asp-
for="Password">Password</label>
                                <div class="col-md-6">
                                    <input autocomplete="off" asp-for="Password" 
type="password" class="form-control" />
                                </div>
                                    <span asp-validation-for="Password" class="text-
danger"></span>
                                </div>
                            <div class="form-group">
                                <div class="col-md-6 col-md-offset-4">
                                    <input id="Submit1" class="btn btn-primary" type="submit" 
value="Login">
                                </div>
                                </div>
</form>
                    </div>
                </div>
                </div>
            </div>
        </div>
</body>
</html>
```

Views/ Normal/ Dashboard.cshtml

```
@{
    Layout = "~/Views/Shared/_SuperAdminLayout.cshtml";
}
```

Views/ Notes/ Create.cshtml

```
@model dotnetcorepms.Models.NotesViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Notes</h3>
    <form method="post" asp-controller="Notes" asp-action="Create">
        @Html.AntiForgeryToken()
        @Html.ValidationSummary(true)

        <div class="panel panel-default">
            <div class="panel-heading">
                Create
            </div>
            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                        <div class="col-xs-12 form-group">
                            <label class="control-label" asp-
for="dbModel.user_id">Member*</label>
                            <select asp-for="dbModel.user_id" class="form-control"
                                    asp-items="@(new SelectList(Model.ddlUsers,"Key",
"Value",0))"></select>
                            <span asp-validation-for="dbModel.user_id" class="text-
danger"></span>
                        </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.note">Notes</label>
                        <textarea asp-for="dbModel.note" rows="10" cols="40" class="form-
control"></textarea>
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Save">
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Notes/ Delete.cshtml

```
@model dotnetcorepms.Models.NotesViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Notes</h3>
    <form method="post" asp-controller="Notes" asp-action="Delete">
        @Html.HiddenFor(m => m.dbModel.id)
        @Html.HiddenFor(m => m.dbModel.created_at)
        @Html.AntiForgeryToken()
        <!-- CSRF protection to prevent phising / fake pages -->
        @Html.ValidationSummary(true)
        <!-- checks if the  -->

        <div class="panel panel-default">
            <div class="panel-heading">
                Are you sure you want to Delete?
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.user_id">Member*</label>
                        @Html.DropDownListFor(model => model.dbModel.user_id, new
SelectList(Model.ddlUsers, "Key", "Value"), new { @class = "form-control" })
                        <span asp-validation-for="dbModel.user_id" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.note">Note</label>
                        @Html.TextBoxFor(model => model.dbModel.note, new { @class = "form-
control" })
                        <span asp-validation-for="dbModel.note" class="text-danger"></span>
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Delete">
        @Html.ActionLink("Cancel", "Index", "Notes", null, new { @class = "btn btn-danger" })
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Notes/ Edit.cshtml

```
@model dotnetcorepms.Models.NotesViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Notes</h3>
    <form method="post" asp-controller="Notes" asp-action="Edit">
        @Html.HiddenFor(m => m.dbModel.id)
        @Html.HiddenFor(m => m.dbModel.created_at)
        @Html.AntiForgeryToken()
        <!-- CSRF protection to prevent phising / fake pages -->
        @Html.ValidationSummary(true)
        <div class="panel panel-default">
            <div class="panel-heading">
                Edit
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.user_id">Role*</label>
                        @Html.DropDownListFor(model => model.dbModel.user_id, new
SelectList(Model.ddlUsers, "Key", "Value"), new { @class = "form-control" })
                        <span asp-validation-for="dbModel.user_id" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.note">Notes</label>
                        @Html.TextAreaFor(model => model.dbModel.note, new { @class = "form-
control" , rows=10, cols=40})
                        <span asp-validation-for="dbModel.note" class="text-danger"></span>
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Update">
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Notes/ Index.cshtml

```
@model dotnetcorepms.Models.NotesViewModelLst
@{
    ViewData["Title"] = "Index";
}

<h3 class="page-title">Notes</h3>
<p>
    <a href="@Url.Action("Create","Notes")" class="btn btn-success">Add new</a>
</p>
<div class="panel panel-default">
    <div class="panel-heading">List</div>
    <div class="panel-body table-responsive">
        <table id="data-table" class="table table-striped table-bordered select">
            <thead>
                <tr>
                    <th><input name="select_all" value="1" type="checkbox"></th>
                    <th>Member</th>
                    <th>Notes</th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                @foreach (var item in Model.dbModelLst)
                {
                <tr id="@item.id">
                    <td id="@item.id">@item.id</td>
                    <td>@(Model.ddlUsers.Where(x => int.Parse(x.Key.ToString()) ==
item.user_id).Select(x => x.Value).FirstOrDefault())</td>
                    <td>@item.note</td>
                    <td>
                        @Html.ActionLink("View", "View", "Notes", new { id = item.id }, new {
@class = "btn btn-xs btn-primary" })
                        @Html.ActionLink("Edit", "Edit", "Notes", new { id = item.id }, new {
@class = "btn btn-xs btn-info" })
                        @Html.ActionLink("Delete", "Delete", "Notes", new { id = item.id },
new { @class = "btn btn-xs btn-danger" })
```

```
                    </td>
                </tr>
                }
            </tbody>
        </table>
    </div>
</div>
```

Views/ Notes/ View.cshtml

```
@model dotnetcorepms.Models.NotesViewModel
@{
    ViewData["Title"] = "View";
}
<section class="content">
    <h3 class="page-title">Notes</h3>

    <div class="panel panel-default">
        <div class="panel-heading">
            View
        </div>

        <div class="panel-body table-responsive">
            <div class="row">
                <div class="col-md-6">
                    <table class="table table-bordered table-striped">
                        <tbody>
                            <tr>
                                <th>Member</th>
                                <td field-key="role">@(Model.ddlUsers.Where(x =>
x.Key.ToString() == Model.dbModel.user_id.ToString()).Select(x =>
x.Value.ToString()).FirstOrDefault()</td>
                            </tr>
                            <tr>
                                <th>Notes</th>
                                <td field-key="email">@(Model.dbModel.note)</td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>

            <p>&nbsp;</p>
            @Html.ActionLink("Back to list", "Index", "Notes", null, new { @class = "btn btn-
default" })
        </div>
    </div>

    <!-- </div>
    </div> -->
</section>
```

Views/ Roles/ Create.cshtml

```
@model dotnetcorepms.Models.Roles

<!-- Main content -->
<section class="content">
```

```
    <h3 class="page-title">Roles</h3>
    <form method="post" asp-controller="Roles" asp-action="Create">
        @Html.AntiForgeryToken()
        @Html.ValidationSummary(true)
        <div class="panel panel-default">
            <div class="panel-heading">
                Create
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="Rolename">Name*</label>
                        <input asp-for="Rolename" type="text" class="form-control" />
                        <span asp-validation-for="Rolename" class="text-danger"></span>
                    </div>
                </div>
            </div>
        </div>
        <input class="btn btn-danger" type="submit" value="Save">
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Roles/ Delete.cshtml

```
@model dotnetcorepms.Models.RolesViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Roles</h3>
    <form method="post" asp-controller="Roles" asp-action="Delete">
        @Html.HiddenFor(m => m.dbModel.RoleID)
        @Html.HiddenFor(m => m.dbModel.Created_At)
        @Html.AntiForgeryToken()
        <!-- CSRF protection to prevent phising / fake pages -->
        @Html.ValidationSummary(true)
        <!-- checks if the  -->

        <div class="panel panel-default">
            <div class="panel-heading">
                Are you sure you want to Delete?
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
```

```
                        <label class="control-label" asp-for="dbModel.Rolename">Name*</label>
                        @Html.TextBoxFor(model => model.dbModel.Rolename, new { @class =
"form-control" })
                        <span asp-validation-for="dbModel.Rolename" class="text-
danger"></span>
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Delete">
        @Html.ActionLink("Cancel", "Index", "Roles", null, new { @class = "btn btn-danger" })
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Roles/ Edit.cshtml

```
@model dotnetcorepms.Models.RolesViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Roles</h3>
    <form method="post" asp-controller="Roles" asp-action="Edit">
        @Html.HiddenFor(m => m.dbModel.RoleID)
        @Html.HiddenFor(m => m.dbModel.Created_At)
        @Html.AntiForgeryToken()
        <!-- CSRF protection to prevent phising / fake pages -->
        @Html.ValidationSummary(true)
        <div class="panel panel-default">
            <div class="panel-heading">
                Edit
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.Rolename">Name*</label>
                        @Html.TextBoxFor(model => model.dbModel.Rolename, new { @class =
"form-control" })
                        <span asp-validation-for="dbModel.Rolename" class="text-
danger"></span>
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Save">
    </form>

    <!-- </div>
    </div> -->
```

```
</section>
```

Views/ Roles/ Index.cshtml

```
@model dotnetcorepms.Models.RolesViewModelLst
@{
    ViewData["Title"] = "Index";
}

<h3 class="page-title">Roles</h3>
<p>
    <a href="@Url.Action("Create","Roles")" class="btn btn-success">Add new</a>
</p>
<div class="panel panel-default">
    <div class="panel-heading">List</div>
    <div class="panel-body table-responsive">
        <table id="data-table" class="table table-striped table-bordered select">
            <thead>
                <tr>
                    <th><input name="select_all" value="1" type="checkbox"></th>
                    <th>Role Name</th>
                    <th>&nbsp;</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var item in Model.dbModelLst)
                {
                <tr id="@item.RoleID">
                    <td id="@item.RoleID">@item.RoleID</td>
                    <td>@item.Rolename</td>
                    <td>@Html.ActionLink("View", "View", "Roles", new { id = item.RoleID },
new { @class = "btn btn-xs btn-primary" })
                    @Html.ActionLink("Edit", "Edit", "Roles", new { id = item.RoleID }, new {
@class = "btn btn-xs btn-info" })
                    @Html.ActionLink("Delete", "Delete", "Roles", new { id = item.RoleID },
new { @class = "btn btn-xs btn-danger" })</td>
                </tr>
                }
            </tbody>
            </table>
    </div>
</div>
```

Views/ Roles/ View.cshtml

```
@model dotnetcorepms.Models.RolesViewModel
@{
    ViewData["Title"] = "View";
}
<section class="content">

    <!-- <div class="row">
        <div class="col-md-12"> -->


    <h3 class="page-title">Roles</h3>

    <div class="panel panel-default">
```

```
        <div class="panel-heading">
            View
        </div>

        <div class="panel-body table-responsive">
            <div class="row">
                <div class="col-md-6">
                    <table class="table table-bordered table-striped">
                        <tbody>
                            <tr>
                                <th>Title</th>
                                <td field-key="name">@(Model.dbModel.Rolename)</td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>

            <p>&nbsp;</p>
            @Html.ActionLink("Back to list", "Index", "Roles", null, new { @class = "btn btn-
default" })
        </div>
    </div>

    <!-- </div>
    </div> -->
</section>
```

Views/ Shared/ Components/ Breadcrumb/ Default.cshtml

```
@model IEnumerable<Message>
<ol class="breadcrumb">
    <li><a href="/"><i class="fa fa-home"></i> Home</a></li>

    @foreach (var message in Model)
    {
        if (Model.Last() == message)
        {
            <li class="active">@message.DisplayName</li>
        }
        else
        {
            <li><a href="@message.URLPath"> @message.DisplayName</a></li>
        }
    }
</ol>
```

Views/ Shared/ Components/ ControlSidebar/ Default.cshtml

```
<aside class="control-sidebar control-sidebar-dark">
    <!-- Create the tabs -->
    <ul class="nav nav-tabs nav-justified control-sidebar-tabs">
        <li class="active"><a href="#control-sidebar-home-tab" data-toggle="tab"><i class="fa
fa-home"></i></a></li>
        <li><a href="#control-sidebar-settings-tab" data-toggle="tab"><i class="fa fa-
gears"></i></a></li>
    </ul>
    <!-- Tab panes -->
    <div class="tab-content">
        <!-- Home tab content -->
        <div class="tab-pane active" id="control-sidebar-home-tab">
            @Html.Raw(ViewBag.PageHelpContainer)
        </div>
        <!-- /.tab-pane -->
        <!-- Stats tab content -->
        <div class="tab-pane" id="control-sidebar-stats-tab">Stats Tab Content</div>
        <!-- /.tab-pane -->
        <!-- Settings tab content -->
        <div class="tab-pane" id="control-sidebar-settings-tab">
            <form method="post">
                <h3 class="control-sidebar-heading">General Settings</h3>
                <div class="form-group">
                    <label class="control-sidebar-subheading">
                        Report panel usage
                        <input type="checkbox" class="pull-right" checked>
                    </label>
                    <p>
                        Some information about this general settings option
                    </p>
                </div>
                <!-- /.form-group -->
            </form>
        </div>
        <!-- /.tab-pane -->
    </div>
</aside>
```

Views/ Shared/ Components/ Footer/ Default.cshtml

```
<footer class="main-footer">
    <!-- To the right -->
    <div class="pull-right hidden-xs">
        <b>Project Management System</b> (PMS)
    </div>
    <!-- Default to the left -->
    <strong>Copyright &copy; @DateTime.Now.Year <a href="#">Alon Poudel</a>.</strong> All
rights reserved.
</footer>

<script type="text/javascript">
    $(document).ready(function () {

        $('#data-table').DataTable({
            //'ajax': 'https://api.myjson.com/bins/1us28',
            dom: "lBfrtip",
            buttons: [
                { extend: "copy", className: "btn-sm btn-info", exportOptions: { modifier: {
selected: true } } },
                { extend: "csv", className: "btn-sm btn-info", exportOptions: { modifier: {
selected: true } } },
                { extend: "excel", className: "btn-sm btn-info", exportOptions: { modifier: {
selected: true } } },
                { extend: "pdf", className: "btn-sm btn-info", exportOptions: { modifier: {
selected: true } } },
                { extend: "print", className: "btn-sm btn-info", exportOptions: { modifier: {
selected: true }, columns: [4, 5, 6, 7, 8, 9, 10, 11] }, title: "PMS", message: "other
descriptions go here" },
                { extend: "colvis", className: "btn-sm btn-info", exportOptions: { modifier: {
selected: true } } },
            ],
            'select': { 'style': 'multi' },
            'columnDefs': [{
                'targets': 0,
                'searchable': false,
                'orderable': false,
                'width': '1%',
                'className': 'dt-body-center',
                'render': function (data, type, full, meta) {
                    return '<input type="checkbox">';
                },
                'checkboxes': { 'selectRow': true },
            }],
            'order': [1, 'asc'],
        });
    });
</script>
```

Views/ Shared/ Components/ Header/ Default.cshtml

```
<header class="main-header">
    <!-- Logo -->
    <a href="/" class="logo" style="font-size:15px;">
        <!-- mini logo for sidebar mini 50x50 pixels -->
        <span class="logo-mini">PMS</span>
        <!-- logo for regular state and mobile devices -->
        <span class="logo-lg">Project Management System</span>
    </a>
    <!-- Header Navbar -->
    <nav class="navbar navbar-static-top" role="navigation">
        <!-- Sidebar toggle button-->
        <a href="#" class="sidebar-toggle" data-toggle="offcanvas" role="button">
            <span class="sr-only">Toggle navigation</span>
        </a>
    </nav>
</header>
```

Views/ Shared/ Components/ MenuMessage/ Defualt.cshtml

```
@model IEnumerable<Message>
<li class="dropdown messages-menu">
    <!-- Menu toggle button -->
    <a href="#" class="dropdown-toggle" data-toggle="dropdown">
        <i class="fa fa-envelope-o"></i>
        <span class="label label-success">@Model.Count()</span>
    </a>
    <ul class="dropdown-menu">
        <li class="header">You have @Model.Count() messages</li>
        <li>
            <!-- inner menu: contains the messages -->
            <ul class="menu">
            @foreach (var message in Model)
            {
                <li>
                    <!-- start message -->
                    <a href="@message.URLPath">
                        <div class="pull-left">
                            <!-- User Image -->
                            <img src="@message.AvatarURL" class="img-circle" alt="ui">
                        </div>
                        <!-- Message title and timestamp -->
                        <h4>
                            @message.DisplayName
                            <small><i class="fa fa-clock-o"></i> @message.TimeSpan</small>
                        </h4>
                        <!-- The message -->
                        <p>@message.ShortDesc</p>
                    </a>
                </li>
            }
                <!-- end message -->
            </ul>
            <!-- /.menu -->
        </li>
        <li class="footer"><a href="#">See All Messages</a></li>
    </ul>
</li>
```

Views/ Shared/ Components/ MenuNotification/ Default.cshtml

```
@model IEnumerable<Message>
<li class="dropdown notifications-menu">
    <!-- Menu toggle button -->
    <a href="#" class="dropdown-toggle" data-toggle="dropdown">
        <i class="fa fa-bell-o"></i>
        <span class="label label-warning">@Model.Count()</span>
    </a>
    <ul class="dropdown-menu">
        <li class="header">You have @Model.Count() notifications</li>
        <li>
            <!-- Inner Menu: contains the notifications -->
            <ul class="menu">
            @foreach (var message in Model)
            {
                <li>
                    <!-- start notification -->
                    <a href="@message.URLPath">
                        <i class="@message.FontAwesomeIcon"></i> @message.ShortDesc
                    </a>
                </li>
            }
                <!-- end notification -->
            </ul>
        </li>
        <li class="footer"><a href="#">View all</a></li>
    </ul>
</li>
```

Views/ Shared/ Components/ MenuTask/ Default.cshtml

```
@model IEnumerable<Message>
```

```
<li class="dropdown tasks-menu">
    <!-- Menu Toggle Button -->
    <a href="#" class="dropdown-toggle" data-toggle="dropdown">
        <i class="fa fa-flag-o"></i>
        <span class="label label-danger">@Model.Count()</span>
    </a>
    <ul class="dropdown-menu">
        <li class="header">You have @Model.Count() tasks</li>
        <li>
            <!-- Inner menu: contains the tasks -->
            <ul class="menu">
                @foreach (var message in Model)
                {
                <li>
                    <!-- Task item -->
                    <a href="#">
                        <!-- Task title and progress text -->
                        <h3>
                            @message.ShortDesc
                            <small class="pull-right">@(message.Percentage)%</small>
                        </h3>
                        <!-- The progress bar -->
                        <div class="progress xs">
                            <!-- Change the css width attribute to simulate progress -->
                            <div class="progress-bar progress-bar-aqua" style="width: 20%"
role="progressbar" aria-valuenow="@message.Percentage" aria-valuemin="0" aria-valuemax="100">
                                <span class="sr-only">@(message.Percentage)% Complete</span>
                            </div>
                        </div>
                    </a>
                </li>
                }
                <!-- end task item -->
            </ul>
        </li>
        <li class="footer">
            <a href="#">View all tasks</a>
        </li>
    </ul>
</li>
```

Views/ Shared/ Components/ PageAlert/ Default.cshtml

```
@model IEnumerable<Message>

@foreach (var item in Model)
{
    if (item.Type == "error")
    {
        <div class="alert alert-danger alert-dismissible">
            <button type="button" class="close" data-dismiss="alert" aria-
hidden="true">×</button>
            <b><i class="icon fa fa-ban"></i> Error!</b> @Html.Raw(@item.ShortDesc)
        </div>
    }

    if (item.Type == "info")
    {
        <div class="alert alert-info alert-dismissible">
            <button type="button" class="close" data-dismiss="alert" aria-
hidden="true">×</button>
            <b><i class="icon fa fa-info"></i> Info!</b> @Html.Raw(@item.ShortDesc)
        </div>
    }

    if (item.Type == "warning")
    {
        <div class="alert alert-warning alert-dismissible">
            <button type="button" class="close" data-dismiss="alert" aria-
hidden="true">×</button>
            <b><i class="icon fa fa-warning"></i> Warning!</b> @Html.Raw(@item.ShortDesc)
        </div>
    }

    if (item.Type == "success")
    {
        <div class="alert alert-success alert-dismissible">
            <button type="button" class="close" data-dismiss="alert" aria-
hidden="true">×</button>
            <b><i class="icon fa fa-check"></i> Success!</b> @Html.Raw(@item.ShortDesc)
        </div>
    }
}
```

Views/ Shared/ Components/ PageHeader/ Default.cshtml

```
@model Tuple<string, string>
<h1>
    @Model.Item1
    <small>@Model.Item2</small>
</h1>
```

Views/ Shared/ Components/ Sidebar/ Default.cshtml

```
@model IEnumerable<dotnetcorepms.Models.SidebarMenu>
<aside class="main-sidebar">
    <!-- sidebar: style can be found in sidebar.less -->
    <section class="sidebar">
        <!-- search form (Optional) -->
        @*<form action="#" method="get" class="sidebar-form">
            <div class="input-group">
                <input type="text" name="q" class="form-control" placeholder="Search...">
                <span class="input-group-btn">
                    <button type="submit" name="search" id="search-btn" class="btn btn-flat">
                        <i class="fa fa-search"></i>
                    </button>
                </span>
            </div>
        </form>*@
        <!-- /.search form -->
        <!-- Sidebar Menu -->
        <ul class="sidebar-menu">
            @foreach (var menu in Model as IEnumerable<SidebarMenu>)
            {
                @*if (menu.Type == SidebarMenuType.Header)
                {
                    <li class="header">@menu.Name</li>
                }
                else*@ if (menu.Type == SidebarMenuType.Link)
                {
                    var active = string.Empty;
                    if (menu.URLPath != "/" &&
@Context.Request.Path.Value.Contains(menu.URLPath))
                    {
                        active = "active";
                    }
                    <li class="@active">
                        <a href="@menu.URLPath">
                            <i class="@menu.IconClassName"></i>
                            <span>@menu.Name</span>
                            @*<span class="pull-right-container">
                                @{
                                    if (menu.LinkCounter.Item1 > 0)
                                    {
                                        <small class="label pull-right bg-
blue">@menu.LinkCounter.Item1</small>
                                    }
                                    if (menu.LinkCounter.Item2 > 0)
                                    {
                                        <small class="label pull-right bg-
green">@menu.LinkCounter.Item2</small>
                                    }
                                    if (menu.LinkCounter.Item3 > 0)
                                    {
                                        <small class="label pull-right bg-
red">@menu.LinkCounter.Item3</small>
                                    }
                                }
                            </span>*@
                        </a>
                    </li>

                    }
                    else if (menu.Type == SidebarMenuType.Tree)
                    {
                        var active = string.Empty;
                        if (menu.TreeChild.Any(x =>
@Context.Request.Path.Value.Contains(x.URLPath)))
                        {
                            active = "active";
                        }

                    <li class="treeview @active">
                        <a href="#">
                            <i class="@menu.IconClassName"></i> <span>@menu.Name</span>
                            <span class="pull-right-container">
                                <i class="fa fa-angle-left pull-right"></i>
                            </span>
                        </a>
                        <ul class="treeview-menu">
                            @foreach (SidebarMenu subMenu in menu.TreeChild)
                            {
                                active = string.Empty;
                                if (Context.Request.Path.Value.Contains(subMenu.URLPath))
                                {
                                    active = "active";
                                }
                                <li class="@active"><a href="@subMenu.URLPath"><i
class="@subMenu.IconClassName"></i><span>@subMenu.Name</span>
</a></li>
                            }
                        </ul>
                    </li>
                }
            }
        </ul>
        <!-- /.sidebar-menu -->
    </section>
    <!-- /.sidebar -->
</aside>
```

Views/ Shared/ Components/ _AdminLayout.cshtml

```
<!DOCTYPE html>
<!--
This is a starter template page. Use this page to start your new project from
scratch. This page gets rid of all links and provides the needed markup only.
-->
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Project Management System</title>
    <!-- Tell the browser to be responsive to screen width -->
    <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no"
name="viewport">
    <environment names="Development">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
        <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.css" />
        <link rel="stylesheet" href="~/lib/Ionicons/css/ionicons.css" />
        <link rel="stylesheet" href="~/css/AdminLTE.css" />
        <link rel="stylesheet" href="~/css/skins/skin-blue.css" />
        <link rel="stylesheet" href="~/css/site.css" />
    </environment>
    <environment names="Staging,Production">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.min.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/lib/Ionicons/css/ionicons.min.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/css/AdminLTE.min.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/css/skins/skin-blue.min.css" asp-append-version="true"
/>
        <link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />
    </environment>
    <environment names="Development">
        <script src="~/lib/jquery/dist/jquery.js" asp-append-version="true"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.js" asp-append-version="true"></script>
        <script src="~/js/app.js" asp-append-version="true"></script>
        <script src="~/lib/PACE/pace.js" asp-append-version="true"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
    </environment>
    <environment names="Staging,Production">
        <script src="~/lib/jquery/dist/jquery.min.js" asp-append-version="true"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.min.js" asp-append-
version="true"></script>
        <script src="~/js/app.min.js" asp-append-version="true"></script>
        <script src="~/lib/PACE/pace.min.js" asp-append-version="true"></script>
        <script src="~/js/site.min.js" asp-append-version="true"></script>
    </environment>
</head>
<body class="hold-transition skin-blue sidebar-mini">
    <div class="wrapper">
        <!-- Main Header -->
        @await Component.InvokeAsync("Header")
        <!-- Left side column. contains the logo and sidebar -->
        @await Component.InvokeAsync("Sidebar")
        <!-- Content Wrapper. Contains page content -->
        <div class="content-wrapper">
            <!-- Content Header (Page header) -->
            @*<section class="content-header">
                @await Component.InvokeAsync("PageHeader")
                @await Component.InvokeAsync("Breadcrumb")
            </section>*@
            <!-- Main content -->
            <section class="content">
                <!-- Your Page Content Here -->
                @RenderBody()
            </section>
            <!-- /.content -->
        </div>
        <!-- /.content-wrapper -->
        <!-- Main Footer -->
        @await Component.InvokeAsync("Footer")
        <!-- Control Sidebar -->
        @await Component.InvokeAsync("ControlSidebar")
        <!-- /.control-sidebar -->
        <!-- Add the sidebar's background. This div must be placed
             immediately after the control sidebar -->
        <div class="control-sidebar-bg"></div>
    </div>
    <!-- ./wrapper -->
    <!-- Optionally, you can add Slimscroll and FastClick plugins.
    Both of these plugins are recommended to enhance the
    user experience. Slimscroll is required when using the
    fixed layout. -->
    @RenderSection("scripts", required: false)
    <script>
                        //useSubmitClass();
    </script>
</body>
</html>
```

Views/ Shared/ Components/ _Layout.cshtml

```
<!DOCTYPE html>
<!--
This is a starter template page. Use this page to start your new project from
scratch. This page gets rid of all links and provides the needed markup only.
-->
```

```
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Project Management System</title>
    <!-- Tell the browser to be responsive to screen width -->
    <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" name="viewport">
    <environment names="Development">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
        <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.css" />
        <link rel="stylesheet" href="~/lib/Ionicons/css/ionicons.css" />
        <link rel="stylesheet" href="~/css/AdminLTE.css" />
        <link rel="stylesheet" href="~/css/skins/skin-blue.css" />
        <link rel="stylesheet" href="~/lib/jquery-comments/jquery-comments.css" />
        <link rel="stylesheet" href="~/css/site.css" />
        <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/v/dt/jszip-2.5.0/dt-1.10.18/b-1.5.2/b-colvis-1.5.2/b-flash-1.5.2/b-html5-1.5.2/b-print-1.5.2/sl-1.2.6/datatables.min.css" />
    </environment>
    <environment names="Staging,Production">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.min.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/lib/Ionicons/css/ionicons.min.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/css/AdminLTE.min.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/css/skins/skin-blue.min.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/lib/jquery-comments/jquery-comments.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />
    </environment>
    <environment names="Development">
        <script src="~/lib/jquery/dist/jquery.js" asp-append-version="true"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.js" asp-append-version="true"></script>
        <script src="~/js/app.js" asp-append-version="true"></script>
        <script src="~/lib/PACE/pace.js" asp-append-version="true"></script>
        <script src="~/lib/jquery-comments/jquery-comments.js" asp-append-version="true"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
        <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.36/pdfmake.min.js"></script>
        <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.36/vfs_fonts.js"></script>
        <script type="text/javascript" src="https://cdn.datatables.net/v/dt/jszip-2.5.0/dt-1.10.18/b-1.5.2/b-colvis-1.5.2/b-flash-1.5.2/b-html5-1.5.2/b-print-1.5.2/sl-1.2.6/datatables.min.js"></script>
    </environment>
    <environment names="Staging,Production">
        <script src="~/lib/jquery/dist/jquery.min.js" asp-append-version="true"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.min.js" asp-append-version="true"></script>
        <script src="~/js/app.min.js" asp-append-version="true"></script>
        <script src="~/lib/PACE/pace.min.js" asp-append-version="true"></script>
        <script src="~/lib/jquery-comments/jquery-comments.js" asp-append-version="true"></script>
        <script src="~/js/site.min.js" asp-append-version="true"></script>
    </environment>
</head>
<body class="hold-transition skin-blue sidebar-mini">
    <div class="wrapper">
        <!-- Main Header -->
        @await Component.InvokeAsync("Header")
        <!-- Left side column. contains the logo and sidebar -->
        @await Component.InvokeAsync("Sidebar")
        <!-- Content Wrapper. Contains page content -->
        <div class="content-wrapper">
            <!-- Content Header (Page header) -->
            @*<section class="content-header">
                @await Component.InvokeAsync("PageHeader")
                @await Component.InvokeAsync("Breadcrumb")
            </section>*@
            <!-- Main content -->
            <section class="content">
                <!-- Your Page Content Here -->
                @RenderBody()
            </section>
            <!-- /.content -->
        </div>
        <!-- /.content-wrapper -->
        <!-- Main Footer -->
        @await Component.InvokeAsync("Footer")
        <!-- Control Sidebar -->
        @await Component.InvokeAsync("ControlSidebar")
        <!-- /.control-sidebar -->
        <!-- Add the sidebar's background. This div must be placed
             immediately after the control sidebar -->
        <div class="control-sidebar-bg"></div>
    </div>
    <!-- ./wrapper -->
    <!-- Optionally, you can add Slimscroll and FastClick plugins.
    Both of these plugins are recommended to enhance the
    user experience. Slimscroll is required when using the
    fixed layout. -->
    @RenderSection("scripts", required: false)
    <script>
                //useSubmitClass();
    </script>
</body>
</html>
```

Views/ Shared/ Components/ _NormalLayout.cshtml

```
<!DOCTYPE html>
<!--
This is a starter template page. Use this page to start your new project from
scratch. This page gets rid of all links and provides the needed markup only.
-->
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Project Management System</title>
    <!-- Tell the browser to be responsive to screen width -->
    <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no"
name="viewport">
    <environment names="Development">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
        <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.css" />
        <link rel="stylesheet" href="~/lib/Ionicons/css/ionicons.css" />
        <link rel="stylesheet" href="~/css/AdminLTE.css" />
        <link rel="stylesheet" href="~/css/skins/skin-blue.css" />
        <link rel="stylesheet" href="~/css/site.css" />
    </environment>
    <environment names="Staging,Production">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.min.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/lib/Ionicons/css/ionicons.min.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/css/AdminLTE.min.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/css/skins/skin-blue.min.css" asp-append-version="true"
/>
        <link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />
    </environment>
    <environment names="Development">
        <script src="~/lib/jquery/dist/jquery.js" asp-append-version="true"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.js" asp-append-version="true"></script>
        <script src="~/js/app.js" asp-append-version="true"></script>
        <script src="~/lib/PACE/pace.js" asp-append-version="true"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
    </environment>
    <environment names="Staging,Production">
        <script src="~/lib/jquery/dist/jquery.min.js" asp-append-version="true"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.min.js" asp-append-
version="true"></script>
        <script src="~/js/app.min.js" asp-append-version="true"></script>
        <script src="~/lib/PACE/pace.min.js" asp-append-version="true"></script>
        <script src="~/js/site.min.js" asp-append-version="true"></script>
    </environment>
</head>
<body class="hold-transition skin-blue sidebar-mini">
    <div class="wrapper">
        <!-- Main Header -->
        @await Component.InvokeAsync("Header")
        <!-- Left side column. contains the logo and sidebar -->
        @await Component.InvokeAsync("Sidebar")
        <!-- Content Wrapper. Contains page content -->
        <div class="content-wrapper">
            <!-- Content Header (Page header) -->
            @*<section class="content-header">
                    @await Component.InvokeAsync("PageHeader")
                    @await Component.InvokeAsync("Breadcrumb")
                </section>*@
            <!-- Main content -->
            <section class="content">
                <!-- Your Page Content Here -->
                @RenderBody()
            </section>
            <!-- /.content -->
        </div>
        <!-- /.content-wrapper -->
        <!-- Main Footer -->
        @await Component.InvokeAsync("Footer")
        <!-- Control Sidebar -->
        @await Component.InvokeAsync("ControlSidebar")
        <!-- /.control-sidebar -->
        <!-- Add the sidebar's background. This div must be placed
             immediately after the control sidebar -->
        <div class="control-sidebar-bg"></div>
    </div>
    <!-- ./wrapper -->
    <!-- Optionally, you can add Slimscroll and FastClick plugins.
    Both of these plugins are recommended to enhance the
    user experience. Slimscroll is required when using the
    fixed layout. -->
    @RenderSection("scripts", required: false)
    <script>
                                //useSubmitClass();
    </script>
</body>
</html>
```

Views/ Shared/ Components/ _ValidationScriptsPartial.cshtml

```
<environment names="Development">
    <script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>
    <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>
</environment>
<environment names="Staging,Production">
    <script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.min.js"
            asp-fallback-src="~/lib/jquery-validation/dist/jquery.validate.min.js"
            asp-fallback-test="window.jQuery && window.jQuery.validator"></script>
    <script
src="https://ajax.aspnetcdn.com/ajax/jquery.validation.unobtrusive/3.2.6/jquery.validate.unobt
rusive.min.js"
            asp-fallback-src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"
            asp-fallback-test="window.jQuery && window.jQuery.validator &&
window.jQuery.validator.unobtrusive"></script>
</environment>
```

Views/ Shared/ Components/ Error.cshtml

```
@{
    ViewData["Title"] = "403 Forbidden";
}
<h1 class="text-danger">403 Forbidden</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>
<h3>Unauthorized Access</h3>
<p>
   Sorry, you are not authorized to access this page. Please contact your administrator.
</p>
```

Views/ Shared/ UserProfile/ Profile.cshtml

```
@model dotnetcorepms.Models.UsersModel
@{
    Layout = "~/Views/Shared/_LayoutCustomer.cshtml";
}
```

```
<div class="container">
    <br />
    <div class="panel panel-default">
        <div class="panel-heading">Profile</div>
        <div class="panel-body">
                <div class="row">
                    <div class="col-lg-4">
                        <label class="control-label" asp-for="Name">Name</label>
                        <input asp-for="Name" readonly="readonly" type="text" class="form-
control" />
                    </div>
                </div>
                <div class="row">
                    <div class="col-lg-4">
                        <label class="control-label" asp-for="EmailID">EmailID</label>
                        <input asp-for="EmailID" readonly="readonly" type="text" class="form-
control" />
                    </div>
                </div>
                <div class="row">
                    <div class="col-lg-4">
                        <label class="control-label" asp-for="Username">Username</label>
                        <input asp-for="Username" readonly="readonly" type="text" class="form-
control" />
                    </div>
                </div>
        </div>
    </div>
</div>
```

Views/ Shared/ Users/ Create.cshtml

```
@model dotnetcorepms.Models.UsersViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Users</h3>
    <form method="post" asp-controller="Users" asp-action="Create">
        @Html.AntiForgeryToken()
        @Html.ValidationSummary(true)

        <div class="panel panel-default">
            <div class="panel-heading">
                Create
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.Name">Name*</label>
                        <input asp-for="dbModel.Name" type="text" class="form-control" />
                        <span asp-validation-for="dbModel.Name" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
```

```
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.EmailID">Email*</label>
                        <input asp-for="dbModel.EmailID" type="text" class="form-control" />
                        <span asp-validation-for="dbModel.EmailID" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-
for="dbModel.Username">Username*</label>
                        <input asp-for="dbModel.Username" type="text" class="form-control" />
                        <span asp-validation-for="dbModel.Username" class="text-
danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-
for="dbModel.Password">Password*</label>
                        <input asp-for="dbModel.Password" type="password" class="form-control"
/>

                        <span asp-validation-for="dbModel.Password" class="text-
danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.RoleID">Role*</label>
                        <select asp-for="dbModel.RoleID" class="form-control"
                                asp-items="@(new SelectList(Model.ddlRoleLst,"Key",
"Value",0))"></select>
                        <span asp-validation-for="dbModel.RoleID" class="text-danger"></span>
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Save">
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Shared/ Users/ Delete.cshtml

```
@model dotnetcorepms.Models.UsersViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Users</h3>
    <form method="post" asp-controller="Users" asp-action="Delete">
        @Html.HiddenFor(m => m.dbModel.ID)
        @Html.HiddenFor(m => m.dbModel.Password)
        @Html.HiddenFor(m => m.dbModel.ConfirmPassword)
        @Html.HiddenFor(m => m.dbModel.CreatedOn)
        @Html.AntiForgeryToken()
        <!-- CSRF protection to prevent phising / fake pages -->
        @Html.ValidationSummary(true)
        <!-- checks if the  -->

        <div class="panel panel-default">
            <div class="panel-heading">
                Are you sure you want to Delete?
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.Name">Name*</label>
                        @Html.TextBoxFor(model => model.dbModel.Name, new { @class = "form-
control" })
                        <span asp-validation-for="dbModel.Name" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.EmailID">Email*</label>
                        @Html.TextBoxFor(model => model.dbModel.EmailID, new { @class = "form-
control" })
                        <span asp-validation-for="dbModel.EmailID" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-
for="dbModel.Username">Username*</label>
                        @Html.TextBoxFor(model => model.dbModel.Username, new { @class =
"form-control" })
                        <span asp-validation-for="dbModel.Username" class="text-
danger"></span>
                    </div>
                </div>
                @*<div class="row">
                        <div class="col-xs-12 form-group">
                            <label class="control-label" asp-
for="dbModel.Password">Password*</label>
                            @Html.TextBoxFor(model => model.dbModel.Password, new { @class =
"form-control" })
                            <span asp-validation-for="dbModel.Password" class="text-
danger"></span>
                        </div>
                    </div>*@
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.RoleID">Role*</label>
                        @Html.DropDownListFor(model => model.dbModel.RoleID, new
SelectList(Model.ddlRoleLst, "Key", "Value"), new { @class = "form-control" })
                        <span asp-validation-for="dbModel.RoleID" class="text-danger"></span>
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Delete">
        @Html.ActionLink("Cancel", "Index", "Users", null, new { @class = "btn btn-danger" })
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Shared/ Users/ Edit.cshtml

```
@model dotnetcorepms.Models.UsersViewModel
<!-- Main content -->
<section class="content">
    <h3 class="page-title">Users</h3>
    <form method="post" asp-controller="Users" asp-action="Edit">
        @Html.HiddenFor(m => m.dbModel.ID)
        @Html.HiddenFor(m => m.dbModel.Password)
        @Html.HiddenFor(m => m.dbModel.ConfirmPassword)
        @Html.HiddenFor(m => m.dbModel.CreatedOn)
        @Html.AntiForgeryToken()
        <!-- CSRF protection to prevent phising / fake pages -->
        @Html.ValidationSummary(true)
        <div class="panel panel-default">
            <div class="panel-heading">
                Edit
            </div>

            <div class="panel-body">
                @if (TempData["MessageRegistration"] != null)
                {
                    <p class="alert alert-success"
id="successMessage">@TempData["MessageRegistration"]</p>
                }
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.Name">Name*</label>
                        @Html.TextBoxFor(model => model.dbModel.Name, new { @class = "form-
control" })

                        <span asp-validation-for="dbModel.Name" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.EmailID">Email*</label>
                        @Html.TextBoxFor(model => model.dbModel.EmailID, new { @class = "form-
control" })

                        <span asp-validation-for="dbModel.EmailID" class="text-danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-
for="dbModel.Password">Password*</label>
                        @Html.TextBoxFor(model => model.dbModel.Username, new { @class =
"form-control" })
                        <span asp-validation-for="dbModel.Password" class="text-
danger"></span>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 form-group">
                        <label class="control-label" asp-for="dbModel.RoleID">Role*</label>
                        @Html.DropDownListFor(model => model.dbModel.RoleID, new
SelectList(Model.ddlRoleLst, "Key", "Value"), new { @class = "form-control" })
                        <span asp-validation-for="dbModel.RoleID" class="text-danger"></span>
                    </div>
                </div>
            </div>
        </div>

        <input class="btn btn-danger" type="submit" value="Save">
    </form>

    <!-- </div>
    </div> -->
</section>
```

Views/ Shared/ Users/ Index.cshtml

```
@model dotnetcorepms.Models.UsersViewModelLst
@{
    ViewData["Title"] = "Index";
}

<h3 class="page-title">Users</h3>
<p>
    <a href="@Url.Action("Create","Users")" class="btn btn-success">Add new</a>
</p>
@if (TempData["MessageRegistration"] != null)
{
    <p class="alert alert-success" id="successMessage">@TempData["MessageRegistration"]</p>
}
<div class="panel panel-default">
    <div class="panel-heading">List</div>
    <div class="panel-body table-responsive">
        <table id="data-table" class="table table-striped table-bordered select">
            <thead>
                <tr>
                    <th><input name="select_all" type="checkbox"></th>
                    <th>Name</th>
                    <th>Email</th>
                    <th>Role</th>
                    <th>&nbsp;</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var item in Model.dbModelLst)
                {
                <tr id="@item.ID">
                    <td id="@item.ID">@item.ID</td>
                    <td>@item.Name</td>
                    <td>@item.EmailID</td>
                    <td>@(Model.ddlRoleLst.Where(x => int.Parse(x.Key.ToString()) ==
item.RoleID).Select(x => x.Value).FirstOrDefault())</td>
                    <td>
                    @Html.ActionLink("View", "View", "Users", new { id = item.ID }, new {
@class = "btn btn-xs btn-primary" })
                    @Html.ActionLink("Edit", "Edit", "Users", new { id = item.ID }, new {
@class = "btn btn-xs btn-info" })
                    @Html.ActionLink("Delete", "Delete", "Users", new { id = item.ID }, new {
@class = "btn btn-xs btn-danger" })</td>
                </tr>
                }
            </tbody>
        </table>
    </div>
</div>
```

Views/ Shared/ Users/ View.cshtml

```
@model dotnetcorepms.Models.UsersViewModel
@{
    ViewData["Title"] = "View";
}
<section class="content">

    <!-- <div class="row">
        <div class="col-md-12"> -->


    <h3 class="page-title">Users</h3>

    <div class="panel panel-default">
        <div class="panel-heading">
            View
        </div>

        <div class="panel-body table-responsive">
            <div class="row">
                <div class="col-md-6">
                    <table class="table table-bordered table-striped">
                        <tbody>
                            <tr>
                                <th>Name</th>
                                <td field-key="name">@(Model.dbModel.Name)</td>
                            </tr>
                            <tr>
                                <th>Email</th>
                                <td field-key="email">@(Model.dbModel.EmailID)</td>
                            </tr>
                            <tr>
                                <th>Role</th>
                                <td field-key="role">@(Model.ddlRoleLst.Where(x =>
x.Key.ToString() == Model.dbModel.RoleID.ToString()).Select(x =>
x.Value.ToString()).FirstOrDefault())</td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>

            <p>&nbsp;</p>
            @Html.ActionLink("Back to list", "Index", "Users", null, new { @class = "btn btn-
default" })
        </div>
    </div>

    <!-- </div>
    </div> -->
</section>
```

Views/ Shared/ _ViewImports.cshtml

```
using dotnetcorepms
@using dotnetcorepms.Models
@addTagHelper "*, Microsoft.AspNetCore.Mvc.TagHelpers"
@using dotnetcorepms.Common
```

Views/ Shared/ _ViewStart.cshtml

```
@{
    Layout = "_Layout";
}
```

dotnetcorepms.Interfaces/ ICommon.cs

```
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Text;

namespace dotnetcorepms.Interfaces
{
    public interface ICommon
    {
        List<PairModel> GetPairModel(string args, long id = 0);
        List<PairModel> GetPairModelWithDefault(string args, long id = 0);
    }
 }
```

dotnetcorepms.Interfaces/ IDocuments.cs

```
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace dotnetcorepms.Interfaces
{
    public interface IDocuments
    {
        IQueryable<Documents> getAllDocuments();
        bool checkDocumentnameExists(string docName);
        int addDocument(Documents entity);
        Documents getDocument(int id);
        int Commit(Documents model, int mode);
    }
}
```

dotnetcorepms.Interfaces/ IForum.cs

```
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Text;

namespace dotnetcorepms.Interfaces
{
    public interface IForum
    {
        IEnumerable<Forums> GetComments();
        IEnumerable<Forums> GetCommentsByID(int id);
        int Commit(Forums model, int mode);
    }
}
```

dotnetcorepms.Interfaces/ ILogin.cs

```
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dotnetcorepms.Interfaces
{
    public interface ILogin
    {
        Users ValidateUser(string email, string passWord);
        bool UpdatePassword(Users Registration);
    }
}
```

dotnetcorepms.Interfaces/ INotes.cs

```
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace dotnetcorepms.Interfaces
{
    public interface INotes
    {
        IQueryable<Notes> getAllNotes();
        bool checkNotenameExists(string noteName);
        int addNote(Notes entity);
        int Commit(Notes model, int mode);
        Notes getNote(int idNote);
    }
}
```

dotnetcorepms.Interfaces/ IRoles.cs

```
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dotnetcorepms.Interfaces
{
    public interface IRoles
    {
        int getRolesofUserbyRolename(string Rolename);
        IQueryable<Roles> getAllRoles();
        bool checkRolenameExists(string roleName);
        int addRole(Roles entity);
        Roles getRole(int id);
        int Commit(Roles model, int mode);
    }
}
```

dotnetcorepms.Interfaces/ IUsers.cs

```
using dotnetcorepms.Models;
using System.Linq;

namespace dotnetcorepms.Interfaces
{
    public interface IUsers
    {
        int AddUser(Users entity);
        void AddAdmin(Users entity);
        bool CheckUserNameExists(string Username);
        Users Userinformation(int UserID);
        IQueryable<UsersModel> UserinformationList(string sortColumn, string sortColumnDir, 
string Search);
        IQueryable<UsersModel> getAllUsers();
        int Commit(Users model, int mode);
    }
}
```

Dotnetcorepms.Repositories/ CommonRepo.cs

```
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Text;
using System.Linq;

namespace dotnetcorepms.Repositories
{
    public class CommonRepo : ICommon
    {
        private DatabaseContext _context;

        public CommonRepo(DatabaseContext context)
        {
            _context = context;
        }

        public List<PairModel> GetPairModel(string args, long id)
        {
            List<PairModel> _pairModel = new List<PairModel>();
            switch (args)
            {
                case "RoleName":
                    _pairModel = (from c in _context.Roles
                                  select new PairModel { Key = c.RoleID, Value = c.Rolename
}).ToList<PairModel>();
                    return _pairModel;
                case "Users":
                    _pairModel = (from c in _context.Users
                                  select new PairModel { Key = c.ID, Value = c.Name
}).ToList<PairModel>();
                    return _pairModel;
                case "Notes":
                    _pairModel = (from c in _context.Notes
                                  select new PairModel { Key = c.id, Value = c.note
}).ToList<PairModel>();
                    return _pairModel;
                default:
                    return _pairModel;
            }
        }

        public List<PairModel> GetPairModelWithDefault(string args, long id = 0)
        {
            List<PairModel> result = GetPairModel(args, id);
            result.Add(new PairModel { Key = "0", Value = "---Select---" });
            return result;
        }
    }
}
```

Dotnetcorepms.Repositories/ DatabaseContext.cs

```
using dotnetcorepms.Models;
using Microsoft.EntityFrameworkCore;

namespace dotnetcorepms.Repositories
{
    public class DatabaseContext : DbContext
    {
        public DatabaseContext(DbContextOptions<DatabaseContext> options) : base(options)
        {

        }

        public DbSet<Users> Users { get; set; }
        public DbSet<Roles> Roles { get; set; }
        public DbSet<Forums> Forums { get; set; }
        public DbSet<Documents> Documents { get; set; }
        public DbSet<Notes> Notes { get; set; }
    }
}
```

Dotnetcorepms.Repositories/ DocumentsRepo.cs

```
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace dotnetcorepms.Repositories
{
    public class DocumentsRepo : IDocuments
    {
        private DatabaseContext _context;

        public DocumentsRepo(DatabaseContext context)
        {
            _context = context;
        }

        public IQueryable<Documents> getAllDocuments()
        {
            var result = (from c in _context.Documents
                          select new Documents
                          {
                              id = c.id,
                              name = c.name,
                              description = c.description,
                              file = c.file,
                              created_at = c.created_at,
                              updated_at = c.updated_at,
                              user_id = c.user_id
                          });
            return result;
        }

        public Documents getDocument(int id)
        {
            var result = (from document in _context.Documents
                          where document.id == id
                          select new Documents
                          {
                              created_at = document.created_at,
                              description = document.description,
                              file = document.file,
                              id = document.id,
                              user_id = document.user_id,
                              name = document.name,
                              updated_at = document.updated_at
                          }).SingleOrDefault();
            return result;
        }

        public bool checkDocumentnameExists(string docName)
        {
            var result = (from c in _context.Documents
                          where c.name.ToLower() == docName.ToLower()
                          select c).Count();

            if (result > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public int addDocument(Documents entity)
        {
            _context.Documents.Add(entity);
            return _context.SaveChanges();
        }

        public int Commit(Documents model, int mode)
        {
            int identity = 0;
            // Only persist if any tickets to add or modify.
            if (model == null) return identity;
            // Persist any new or modified tickets.
            if (mode == 0)
            {
                model.created_at = model.updated_at = DateTime.Now;
                _context.Documents.Add(model);
                _context.SaveChanges();
                identity = model.id;
            }
            //Update the existing entity
            else if (mode == 1)
            {
                model.created_at = model.updated_at = DateTime.Now;
                _context.Documents.Attach(model);
                _context.Entry(model).State =
Microsoft.EntityFrameworkCore.EntityState.Modified;
                _context.SaveChanges();
                identity = model.id; ;
            }
            //Remove the existing entity
            else if (mode == 2)
            {
                _context.Documents.Attach(model);
                _context.Entry(model).State =
Microsoft.EntityFrameworkCore.EntityState.Deleted;
                _context.SaveChanges();
            }
            return identity;
        }
    }
}
```

Dotnetcorepms.Repositories/ ForumRepo.cs

```
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace dotnetcorepms.Repositories
{
    public class ForumRepo : IForum
```

```
{
    private DatabaseContext _context;

    public ForumRepo(DatabaseContext context)
    {
        _context = context;
    }

    public IEnumerable<Forums> GetComments()
    {
        return (from c in _context.Forums
                select new Forums
                {
                    content = c.content,
                    created = c.created,
                    //created_by_admin = c.created_by_admin,
                    created_by_current_user = c.created_by_current_user,
                    fullname = c.fullname,
                    ID = c.ID,
                    modified = c.modified,
                    parent = c.parent,
                    profile_picture_url = c.profile_picture_url,
                    upvote_count = c.upvote_count,
                    user_has_upvoted = c.user_has_upvoted,
                }).ToList();
    }

    public IEnumerable<Forums> GetCommentsByID(int id)
    {
        return (from c in _context.Forums
                where c.ID == id
                select new Forums
                {
                    content = c.content,
                    created = c.created,
                    // created_by_admin = c.created_by_admin,
                    created_by_current_user = c.created_by_current_user,
                    fullname = c.fullname,
                    ID = c.ID,
                    modified = c.modified,
                    parent = c.parent,
                    profile_picture_url = c.profile_picture_url,
                    upvote_count = c.upvote_count,
                    user_has_upvoted = c.user_has_upvoted,
                }).ToList();
    }

    public int Commit(Forums model, int mode)
    {
        int identity = 0;
        // Only persist if any tickets to add or modify.
        if (model == null) return identity;
            // Persist any new or modified tickets.
            if (mode == 0)
            {
                model.created = model.modified = DateTime.Now;
                _context.Forums.Add(model);
                _context.SaveChanges();
```

```
                    identity = model.ID;
                }
                //Update the existing entity
                else if (mode == 1)
                {
                    model.created = model.modified = DateTime.Now;
                    _context.Forums.Attach(model);
                    _context.Entry(model).State =
Microsoft.EntityFrameworkCore.EntityState.Modified;
                    _context.SaveChanges();
                    identity = model.ID; ;
                }
                //Remove the existing entity
                else if (mode == 2)
                {
                    _context.Forums.Attach(model);
                    _context.Entry(model).State =
Microsoft.EntityFrameworkCore.EntityState.Deleted;
                    _context.SaveChanges();
                }
            return identity;
        }
    }

}
```

Dotnetcorepms.Repositories/ LoginRepo.cs

```
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using System;
using System.Linq;

namespace dotnetcorepms.Repositories
{
    public class LoginRepo : ILogin
    {
        private DatabaseContext _context;

        public LoginRepo(DatabaseContext context)
        {
            _context = context;
        }

        public Users ValidateUser(string email, string passWord)
        {
            try
            {
                var validate = (from c in _context.Users
                                where c.EmailID == email && c.Password == passWord
                                select c).SingleOrDefault();

                return validate;
            }
            catch (Exception)
            {

                throw;
            }
        }

        public bool UpdatePassword(Users Registration)
        {
            _context.Users.Attach(Registration);
            _context.Entry(Registration).Property(x => x.Password).IsModified = true;
            int result = _context.SaveChanges();

            if(result > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }


    }
}
```

Dotnetcorepms.Repositories/ NotesRepo.cs

```
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace dotnetcorepms.Repositories
{
    public class NotesRepo : INotes
    {
        private DatabaseContext _context;

        public NotesRepo(DatabaseContext context)
        {
            _context = context;
        }

        public IQueryable<Notes> getAllNotes()
        {
            var result = (from c in _context.Notes
                          select new Notes
                          {
                              id = c.id,
                              note = c.note,
                              created_at = c.created_at,
                              updated_at = c.updated_at,
                              user_id = c.user_id
                          });
            return result;
        }

        public bool checkNotenameExists(string noteName)
        {
            var result = (from c in _context.Notes
                          where c.note.ToLower().ToString() == noteName.ToLower().ToString()
                          select c).Count();

            if (result > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public Notes getNote(int idNote)
        {
            var result = (from note in _context.Notes
                          where note.id == idNote
                          select new Notes
                          {
                              id = note.id,
                              created_at= note.created_at,
                              updated_at = note.updated_at,
                              note = note.note,
                              user_id = note.user_id,
                          }).SingleOrDefault();
            return result;
        }

        public int addNote(Notes entity)
        {
            _context.Notes.Add(entity);
            return _context.SaveChanges();
        }

        public int Commit(Notes model, int mode)
        {
            int identity = 0;
            // Only persist if any tickets to add or modify.
            if (model == null) return identity;
            // Persist any new or modified tickets.
            if (mode == 0)
            {
                model.created_at = model.updated_at = DateTime.Now;
                _context.Notes.Add(model);
                _context.SaveChanges();
                identity = model.id;
            }
            //Update the existing entity
            else if (mode == 1)
            {
                model.created_at = model.updated_at = DateTime.Now;
                _context.Notes.Attach(model);
                _context.Entry(model).State =
Microsoft.EntityFrameworkCore.EntityState.Modified;
                _context.SaveChanges();
                identity = model.id; ;
            }
            //Remove the existing entity
            else if (mode == 2)
            {
                _context.Notes.Attach(model);
                _context.Entry(model).State =
Microsoft.EntityFrameworkCore.EntityState.Deleted;
                _context.SaveChanges();
            }
            return identity;
        }
    }
}
```

Dotnetcorepms.Repositories/ RolesRepo.cs

```
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dotnetcorepms.Repositories
{
    public class RolesRepo : IRoles
    {
        private DatabaseContext _context;

        public RolesRepo(DatabaseContext context)
        {
            _context = context;
        }

        public int getRolesofUserbyRolename(string Rolename)
        {
            var roleID = (from role in _context.Roles
                          where role.Rolename == Rolename
                          select role.RoleID).SingleOrDefault();

            return roleID;
        }

        public IQueryable<Roles> getAllRoles()
        {
            var result = (from c in _context.Roles
                          select new Roles
                          {
                              RoleID = c.RoleID,
                              Rolename = c.Rolename
                          });
            return result;
        }

        public Roles getRole(int id)
        {
            var result = (from role in _context.Roles
                          where role.RoleID == id
                          select new Roles
                          {
                              RoleID = role.RoleID,
                              Rolename = role.Rolename,
                              Created_At = role.Created_At,
                              Updated_At = role.Updated_At
                          }).SingleOrDefault();
            return result;
        }

        public bool checkRolenameExists(string roleName)
        {
            var result = (from c in _context.Roles
                          where c.Rolename.ToLower() == roleName.ToLower()
                          select c).Count();

            if (result > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public int addRole(Roles entity)
        {
            _context.Roles.Add(entity);
            return _context.SaveChanges();
        }

        public int Commit(Roles model, int mode)
        {
            int identity = 0;
            // Only persist if any tickets to add or modify.
            if (model == null) return identity;
            // Persist any new or modified tickets.
            if (mode == 0)
            {
                model.Created_At = model.Updated_At = DateTime.Now;
                _context.Roles.Add(model);
                _context.SaveChanges();
                identity = model.RoleID;
            }
            //Update the existing entity
            else if (mode == 1)
            {
                model.Created_At = model.Updated_At = DateTime.Now;
                _context.Roles.Attach(model);
                _context.Entry(model).State =
Microsoft.EntityFrameworkCore.EntityState.Modified;
                _context.SaveChanges();
                identity = model.RoleID; ;
            }
            //Remove the existing entity
            else if (mode == 2)
            {
                _context.Roles.Attach(model);
                _context.Entry(model).State = 
Microsoft.EntityFrameworkCore.EntityState.Deleted;
                _context.SaveChanges();
            }
            return identity;
        }
    }
}
```

Dotnetcorepms.Repositories/ UsersRepo.cs

```
using dotnetcorepms.Interfaces;
using dotnetcorepms.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Dynamic;

namespace dotnetcorepms.Repositories
{
    public class UsersRepo : IUsers
    {
        private DatabaseContext _context;

        public UsersRepo(DatabaseContext context)
        {
            _context = context;
        }

        public void AddAdmin(Users entity)
        {
            _context.Users.Add(entity);
            _context.SaveChanges();
        }

        public int AddUser(Users entity)
        {
            _context.Users.Add(entity);
            return _context.SaveChanges();
        }

        public bool CheckUserNameExists(string Username)
        {
            var result = (from user in _context.Users
                          where user.Username == Username
                          select user).Count();

            if (result > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public Users Userinformation(int UserID)
        {
            var result = (from user in _context.Users
                          where user.ID == UserID
                          select new Users
                          {
                            Name =user.Name,
                            EmailID =user.EmailID,
                            CreatedOn = user.CreatedOn.Value,
                            Username = user.Username,
                            Password = user.Password,
                            ConfirmPassword = user.ConfirmPassword,
                            ID = user.ID,
                            RoleID = user.RoleID
                          }).SingleOrDefault();
            return result;
        }

        public IQueryable<UsersModel> UserinformationList(string sortColumn, string
sortColumnDir, string Search)
        {
            var IQueryableReg = (from user in _context.Users
                                 select new UsersModel
                                 {
                                     Name = user.Name,
                                     EmailID = user.EmailID,
                                     CreatedOn = user.CreatedOn.Value.ToString("dd/MM/yyyy"),
                                     Username = user.Username
                                 });
            if (!(string.IsNullOrEmpty(sortColumn) && string.IsNullOrEmpty(sortColumnDir)))
            {
               // IQueryableReg = IQueryableReg.OrderBy(sortColumn + " " + sortColumnDir);
            }
            if (!string.IsNullOrEmpty(Search))
            {
                IQueryableReg = IQueryableReg.Where(m => m.Username == Search || m.EmailID ==
Search);
            }

            return IQueryableReg;
        }

        public IQueryable<UsersModel> getAllUsers()
        {
            var user = (from c in _context.Users
                         select new UsersModel
                         {
                             ID = c.ID,
                             Name = c.Name,
                             EmailID = c.EmailID,
                             RoleID = c.RoleID
                         });
            return user;
        }
        public int Commit(Users model, int mode)
        {
            int identity = 0;
            // Only persist if any tickets to add or modify.
            if (model == null) return identity;
            // Persist any new or modified tickets.
            if (mode == 0)// Create
            {
                model.CreatedOn = DateTime.Now;
                _context.Users.Add(model);
                _context.SaveChanges();
                identity = model.ID;
            }
            //Update the existing entity
            else if (mode == 1)
            {
                model.CreatedOn = DateTime.Now;
                _context.Users.Attach(model);
                _context.Entry(model).State =
Microsoft.EntityFrameworkCore.EntityState.Modified;
                _context.SaveChanges();
                identity = model.ID; ;
            }
            //Remove the existing entity
            else if (mode == 2)
            {
                _context.Users.Attach(model);
                _context.Entry(model).State =
Microsoft.EntityFrameworkCore.EntityState.Deleted;
                _context.SaveChanges();
            }
            return identity;
        }
    }
}
```

Appendix H: Codes of PMS in Laravel

App/ Http/ Controllers/ Admin/ PMSDocumentController.php

```
<?php

namespace App\Http\Controllers\Admin;

use App\PMSDocument;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Gate;
use App\Http\Controllers\Controller;
use App\Http\Requests\Admin\StorePMSDocumentsRequest;
use App\Http\Requests\Admin\UpdatePMSDocumentsRequest;
use App\Http\Controllers\Traits\FileUploadTrait;

/**
     The PMSDocumentController is responsible for adding the document to the Project
Management System (PMS) by the user where the document is shared with other users to view and
download. The document may be a PDF, or Text file and it can be only edited and deleted by the
user or owner of the document.
     */

class PMSDocumentController extends Controller
{
    use FileUploadTrait;

    /**
     * Display a listing of PMSDocument.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //if the current users do not have access
        if (! Gate::allows('pms_document_access')) {
            return abort(401);
        }

        $pms_documents = PMSDocument::all();

        return view('admin.pms_documents.index', compact('pms_documents'));
    }
    /**
     * Show the form for creating new PMSDocument.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        if (! Gate::allows('pms_document_create')) {
            return abort(401);
        }

        $users = \App\User::get()->pluck('name', 'id')-
>prepend(trans('pms.pms_please_select'), '');

        return view('admin.pms_documents.create', compact('users'));
    }

    /**
     * Store a newly created PMSDocument in storage.
     *
     * @param  \App\Http\Requests\StorePMSDocumentsRequest  $request
     * @return \Illuminate\Http\Response
     */
    public function store(StorePMSDocumentsRequest $request)
    {
        if (! Gate::allows('pms_document_create')) {
            return abort(401);
        }
        $request = $this->saveFiles($request);
        $pms_document = PMSDocument::create($request->all());
        return redirect()->route('admin.pms_documents.index');
    }

    /**
     * Show the form for editing PMSDocument.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        if (! Gate::allows('pms_document_edit')) {
            return abort(401);
        }

        $users = \App\User::get()->pluck('name', 'id')-
>prepend(trans('pms.pms_please_select'), '');

        $pms_document = PMSDocument::findOrFail($id);

        return view('admin.pms_documents.edit', compact('pms_document', 'users'));
    }

    /**
     * Update PMSDocument in storage.
     *
     * @param  \App\Http\Requests\UpdatePMSDocumentsRequest  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(UpdatePMSDocumentsRequest $request, $id)
    {
        if (! Gate::allows('pms_document_edit')) {
            return abort(401);
        }
        $request = $this->saveFiles($request);
        $pms_document = PMSDocument::findOrFail($id);
        $pms_document->update($request->all());
        return redirect()->route('admin.pms_documents.index');
    }

    /**
     * Display PMSDocument.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        if (! Gate::allows('pms_document_view')) {
            return abort(401);
        }
        $pms_document = PMSDocument::findOrFail($id);

        return view('admin.pms_documents.show', compact('pms_document'));
    }
    /**
     * Remove CrmDocument from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        if (! Gate::allows('pms_document_delete')) {
            return abort(401);
        }
        $pms_document = PMSDocument::findOrFail($id);
        $pms_document->delete();

        return redirect()->route('admin.pms_documents.index');
    }

    /**
     * Delete all selected PMSDocument at once.
     *
     * @param Request $request
     */
    public function massDestroy(Request $request)
    {
        if (! Gate::allows('pms_document_delete')) {
            return abort(401);
        }
        if ($request->input('ids')) {
            $entries = PMSDocument::whereIn('id', $request->input('ids'))->get();

            foreach ($entries as $entry) {
                $entry->delete();
            }
        }
    }

}
```

App/ Http/ Controllers/ Admin/ PMSForumController.php

```
<?php

namespace App\Http\Controllers\Admin;
```

```
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

/**
     *The PMSForumController is responsible for providing common platform for all the users or
members in the PMS to discuss on a common topic.
     */

class PMSForumController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return view('admin.pms_forums.index');
    }
}
```

App/ Http/ Controllers/ Admin/ PMSNoteController.php

```
<?php

namespace App\Http\Controllers\Admin;

use App\PMSNote;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Gate;
use App\Http\Controllers\Controller;
use App\Http\Requests\Admin\StorePMSNotesRequest;
use App\Http\Requests\Admin\UpdatePMSNotesRequest;
/**
     The PMSNote is responsible for adding notes so that the user can add his/her note and
share the note among all the members in the PMS.
     */

class PMSNoteController extends Controller
{
    /**
```

```
     * Display a listing of PMSNote.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        if (! Gate::allows('pms_note_access')) {
            return abort(401);
        }

        $pms_notes = PMSNote::all();
        return view('admin.pms_notes.index', compact('pms_notes'));
    }

    /**
     * Show the form for creating new PMSNote.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        if (! Gate::allows('pms_note_create')) {
            return abort(401);
        }

         $users = \App\User::get()->pluck('name', 'id')-
>prepend(trans('pms.pms_please_select'), '');

        return view('admin.pms_notes.create', compact('users'));
    }

    /**
     * Store a newly created PMSNote in storage.
     *
     * @param  \App\Http\Requests\StorePMSNotesRequest  $request
     * @return \Illuminate\Http\Response
     */
    public function store(StorePMSNotesRequest $request)
    {
        if (! Gate::allows('pms_note_create')) {
            return abort(401);
        }
        $pms_note = pmsNote::create($request->all());
        return redirect()->route('admin.pms_notes.index');
    }

    /**
     * Show the form for editing PMSNote.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        if (! Gate::allows('pms_note_edit')) {
            return abort(401);
        }

        $users = \App\User::get()->pluck('name', 'id')-
>prepend(trans('pms.pms_please_select'), '');

        $pms_note = PMSNote::findOrFail($id);

        return view('admin.pms_notes.edit', compact('pms_note', 'users'));
    }

    /**
     * Update PMSNote in storage.
     *
     * @param  \App\Http\Requests\UpdatePMSNotesRequest  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(UpdatePMSNotesRequest $request, $id)
    {
        if (! Gate::allows('pms_note_edit')) {
            return abort(401);
        }
        $pms_note = PMSNote::findOrFail($id);
        $pms_note->update($request->all());
        return redirect()->route('admin.pms_notes.index');
    }

    /**
     * Display PMSNote.
     *
```

```
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    if (! Gate::allows('pms_note_view')) {
        return abort(401);
    }
    $pms_note = PMSNote::findOrFail($id);

    return view('admin.pms_notes.show', compact('pms_note'));
}

/**
 * Remove PMSNote from storage.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    if (! Gate::allows('pms_note_delete')) {
        return abort(401);
    }
    $pms_note = PMSNote::findOrFail($id);
    $pms_note->delete();

    return redirect()->route('admin.pms_notes.index');
}

/**
 * Delete all selected PMSNote at once.
 *
 * @param Request $request
 */
public function massDestroy(Request $request)
{
    if (! Gate::allows('pms_note_delete')) {
        return abort(401);
    }
    if ($request->input('ids')) {
        $entries = PMSNote::whereIn('id', $request->input('ids'))->get();
```

```
            foreach ($entries as $entry) {
                $entry->delete();
            }
        }
    }

}
```

App/ Http/ Controllers/ Admin/ RolesController.php

```
<?php

namespace App\Http\Controllers\Admin;

use App\Role;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Gate;
use App\Http\Controllers\Controller;
use App\Http\Requests\Admin\StoreRolesRequest;
use App\Http\Requests\Admin\UpdateRolesRequest;

/**
    The role controller is responsible to manage roles in the PMS. The role can be either a
super user (administrator) or a normal user where the super user has full privileges to add,
delete, edit and read unlike the normal user who has limited privileges throughout the PMS.
*/

class RolesController extends Controller
{
    /**
     * Display a listing of Role.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        if (! Gate::allows('role_access')) {
            return abort(401);
        }

                $roles = Role::all();

        return view('admin.roles.index', compact('roles'));
    }
```

```php
/**
 * Show the form for creating new Role.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    if (! Gate::allows('role_create')) {
        return abort(401);
    }
    return view('admin.roles.create');
}

/**
 * Store a newly created Role in storage.
 *
 * @param  \App\Http\Requests\StoreRolesRequest  $request
 * @return \Illuminate\Http\Response
 */
public function store(StoreRolesRequest $request)
{
    if (! Gate::allows('role_create')) {
        return abort(401);
    }
    $role = Role::create($request->all());
    return redirect()->route('admin.roles.index');
}

/**
 * Show the form for editing Role.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    if (! Gate::allows('role_edit')) {
        return abort(401);
    }
    $role = Role::findOrFail($id);

    return view('admin.roles.edit', compact('role'));
}

/**
 * Update Role in storage.
 *
 * @param  \App\Http\Requests\UpdateRolesRequest  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(UpdateRolesRequest $request, $id)
{
    if (! Gate::allows('role_edit')) {
        return abort(401);
    }
    $role = Role::findOrFail($id);
    $role->update($request->all());

    return redirect()->route('admin.roles.index');
}

/**
 * Display Role.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    if (! Gate::allows('role_view')) {
        return abort(401);
    }
    $users = \App\User::where('role_id', $id)->get();

    $role = Role::findOrFail($id);

    return view('admin.roles.show', compact('role', 'users'));
}

/**
 * Remove Role from storage.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
```

```
     */
    public function destroy($id)
    {
        if (! Gate::allows('role_delete')) {
            return abort(401);
        }
        $role = Role::findOrFail($id);
        $role->delete();

        return redirect()->route('admin.roles.index');
    }

    /**
     * Delete all selected Role at once.
     *
     * @param Request $request
     */
    public function massDestroy(Request $request)
    {
        if (! Gate::allows('role_delete')) {
            return abort(401);
        }
        if ($request->input('ids')) {
            $entries = Role::whereIn('id', $request->input('ids'))->get();

            foreach ($entries as $entry) {
                $entry->delete();
            }
        }
    }

}
```

App/ Http/ Controllers/ Admin/ UsersController.php

```
<?php

namespace App\Http\Controllers\Admin;

use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Gate;
use App\Http\Controllers\Controller;
use App\Http\Requests\Admin\StoreUsersRequest;
use App\Http\Requests\Admin\UpdateUsersRequest;

/**
    The UsersController is responsible for managing the users in the PMS. It can be accessed
only with administrator/super user privileges. It can be used to create new user, delete the
existing user, update the information of the existing user or list out the available users in
the PMS.
*/

class UsersController extends Controller
{
    /**
     * Display a listing of User.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        if (! Gate::allows('user_access')) {
            return abort(401);
        }

                $users = User::all();
             //  print_r($users);
        return view('admin.users.index', compact('users'));
    }

    /**
     * Show the form for creating new User.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        if (! Gate::allows('user_create')) {
            return abort(401);
        }

        $roles = \App\Role::get()->pluck('title', 'id')-
>prepend(trans('pms.pms_please_select'), '');

        return view('admin.users.create', compact('roles'));
    }

    /**
     * Store a newly created User in storage.
     *
     * @param  \App\Http\Requests\StoreUsersRequest  $request
     * @return \Illuminate\Http\Response
     */
    public function store(StoreUsersRequest $request)
    {
        if (! Gate::allows('user_create')) {
            return abort(401);
        }
        $user = User::create($request->all());


        return redirect()->route('admin.users.index');
    }

    /**
     * Show the form for editing User.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        if (! Gate::allows('user_edit')) {
            return abort(401);
        }

        $roles = \App\Role::get()->pluck('title', 'id')-
>prepend(trans('pms.pms_please_select'), '');

        $user = User::findOrFail($id);

        return view('admin.users.edit', compact('user', 'roles'));
    }

    /**
     * Update User in storage.
     *
     * @param  \App\Http\Requests\UpdateUsersRequest  $request
```

```
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(UpdateUsersRequest $request, $id)
{
    if (! Gate::allows('user_edit')) {
        return abort(401);
    }
    $user = User::findOrFail($id);
    $user->update($request->all());


    return redirect()->route('admin.users.index');
}

/**
 * Display User.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    if (! Gate::allows('user_view')) {
        return abort(401);
    }
    $user = User::findOrFail($id);

    return view('admin.users.show', compact('user'));
}

/**
 * Remove User from storage.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    if (! Gate::allows('user_delete')) {
        return abort(401);
    }
    $user = User::findOrFail($id);
```

```
        $user->delete();

        return redirect()->route('admin.users.index');
    }

    /**
     * Delete all selected User at once.
     *
     * @param Request $request
     */
    public function massDestroy(Request $request)
    {
        if (! Gate::allows('user_delete')) {
            return abort(401);
        }
        if ($request->input('ids')) {
            $entries = User::whereIn('id', $request->input('ids'))->get();

            foreach ($entries as $entry) {
                $entry->delete();
            }
        }
    }

}
```

App/ Http/ Controllers/ Auth/ ChangePasswordController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use Illuminate\Http\Request;
use Hash;
use Validator;

class ChangePasswordController extends Controller
{

    /**
     * Create a new controller instance.
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Where to redirect users after password is changed.
     *
     * @var string $redirectTo
     */
    protected $redirectTo = '/change_password';

    /**
     * Change password form
     *
     * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
     */
    public function showChangePasswordForm()
    {
        $user = Auth::getUser();

        return view('auth.change_password', compact('user'));
    }

    /**
     * Change password.
     *
     * @param Request $request
     * @return $this|\Illuminate\Http\RedirectResponse
     */
    public function changePassword(Request $request)
    {
        $user = Auth::getUser();
        $this->validator($request->all())->validate();
        if (Hash::check($request->get('current_password'), $user->password)) {
            $user->password = $request->get('new_password');
            $user->save();
            return redirect($this->redirectTo)->with('success', 'Password change
successfully!');
        } else {
            return redirect()->back()->withErrors('Current password is incorrect');
        }
    }

    /**
     * Get a validator for an incoming change password request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'current_password' => 'required',
            'new_password' => 'required|min:6|confirmed',
        ]);
    }
}
```

App/ Http/ Controllers/ Auth/ ForgotPasswordController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\SendsPasswordResetEmails;

class ForgotPasswordController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Password Reset Controller
    |--------------------------------------------------------------------------
    |
    | This controller is responsible for handling password reset emails and
    | includes a trait which assists in sending these notifications from
    | your application to your users. Feel free to explore this trait.
    |
    */

    use SendsPasswordResetEmails;

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

App/ Http/ Controllers/ Auth/ LoginController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;
use Socialite;
use Auth;
use App\User;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login / registration.
     *
     * @var string
     */
    protected $redirectTo = '/admin/home';

    /**
```

```
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest', ['except' => 'logout']);
    }


}
```

App/ Http/ Controllers/ Auth/ RegisterController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\User;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Validator;
use Illuminate\Foundation\Auth\RegistersUsers;

class RegisterController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Register Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles the registration of new users as well as their
    | validation and creation. By default this controller uses a trait to
    | provide this functionality without requiring any additional code.
    |
    */

    use RegistersUsers;

    /**
     * Where to redirect users after registration.
     *
     * @var string
     */
    protected $redirectTo = '/admin/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6|confirmed',
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => bcrypt($data['password']),
        ]);
    }
}
```

App/ Http/ Controllers/ Auth/ ResetPasswordController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ResetsPasswords;

class ResetPasswordController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Password Reset Controller
    |--------------------------------------------------------------------------
    |
    | This controller is responsible for handling password reset requests
    | and uses a simple trait to include this behavior. You're free to
    | explore this trait and override any methods you wish to tweak.
    |
    */

    use ResetsPasswords;

    /**
     * Where to redirect users after resetting their password.
     *
     * @var string
     */
    protected $redirectTo = '/admin/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

App/ Http/ Controllers/ Traits/ UploadFileTrait.php

```php
<?php
```

```
namespace App\Http\Controllers\Traits;

use Illuminate\Http\Request;
use Intervention\Image\Facades\Image;

/**This is a helper file in Laravel to help the application to upload the file into the
application server. This is responsible to create a new folder if it doesn’t exist and upload
the user uploaded file based on the file type. For instance, if the user uploaded file is an
image file, it resizes the file size and uploads into the server. It is also responsible for
managing the file permission so that it is accessible to the application users.
*/

trait FileUploadTrait
{

    /**
     * File upload trait used in controllers to upload files
     */
    public function saveFiles(Request $request)
    {

        $uploadPath = public_path(env('UPLOAD_PATH'));
        $thumbPath = public_path(env('UPLOAD_PATH').'/thumb');
        if (! file_exists($uploadPath)) {
            mkdir($uploadPath, 0775);
            mkdir($thumbPath, 0775);
        }

        $finalRequest = $request;

        foreach ($request->all() as $key => $value) {
            if ($request->hasFile($key)) {
                if ($request->has($key . '_max_width') && $request->has($key . '_max_height'))
{
                    // Check file width
                    $filename = time() . '-' . $request->file($key)->getClientOriginalName();
                    $file     = $request->file($key);
                    $image    = Image::make($file);
                    if (! file_exists($thumbPath)) {
                        mkdir($thumbPath, 0775, true);
                    }
                    Image::make($file)->resize(50, 50)->save($thumbPath . '/' . $filename);
                    $width  = $image->width();
                    $height = $image->height();
                    if ($width > $request->{$key . '_max_width'} && $height > $request->{$key
. '_max_height'}) {
                        $image->resize($request->{$key . '_max_width'}, $request->{$key .
'_max_height'});
                    } elseif ($width > $request->{$key . '_max_width'}) {
                        $image->resize($request->{$key . '_max_width'}, null, function
($constraint) {
                            $constraint->aspectRatio();
                        });
                    } elseif ($height > $request->{$key . '_max_width'}) {
                        $image->resize(null, $request->{$key . '_max_height'}, function
($constraint) {
                            $constraint->aspectRatio();
                        });
                    }
                    $image->save($uploadPath . '/' . $filename);
                    $finalRequest = new Request(array_merge($finalRequest->all(), [$key =>
$filename]));
                } else {
                    $filename = time() . '-' . $request->file($key)->getClientOriginalName();
                    $request->file($key)->move($uploadPath, $filename);
                    $finalRequest = new Request(array_merge($finalRequest->all(), [$key =>
$filename]));
                }
            }
        }

        return $finalRequest;
    }
}
```

App/ Http/ Controllers/ Controller.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Routing\Controller as BaseController;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
```

```
/**
Controller is the base controller class which provides few convenience methods such as the
middleware methods, which may be used to attach middleware to controller actions. Simply put,
it is a basis for providing good functionalities to newly created controllers.
*/

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
}
```

App/ Http/ Controllers/ HomeController.php

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests;
use Illuminate\Http\Request;
use App\User;

class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $count_users = User::all();
        return view('home', compact('count_users'));
    }
}
```

App/ Http/ Controllers/ Middleware/ EncryptCookies.php

```php
<?php

namespace App\Http\Middleware;

use Illuminate\Cookie\Middleware\EncryptCookies as Middleware;

class EncryptCookies extends Middleware
{
    /**
     * The names of the cookies that should not be encrypted.
     *
     * @var array
     */
    protected $except = [
        //
    ];
}
```

App/ Http/ Controllers/ Middleware/ RedirectIfAuthenticated.php

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\Auth;

class RedirectIfAuthenticated
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @param  string|null  $guard
     * @return mixed
     */
    public function handle($request, Closure $next, $guard = null)
    {
        if (Auth::guard($guard)->check()) {
            return redirect('/admin/home');
        }

        return $next($request);
    }
}
```

App/ Http/ Controllers/ Middleware/ TrimStrings.php

```php
<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\TrimStrings as Middleware;

class TrimStrings extends Middleware
{
    /**
     * The names of the attributes that should not be trimmed.
     *
     * @var array
     */
    protected $except = [
        'password',
        'password_confirmation',
    ];
}
```

App/ Http/ Controllers/ Middleware/ VerifyCsrfToken.php

```php
<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as Middleware;

class VerifyCsrfToken extends Middleware
{
    /**
     * The URIs that should be excluded from CSRF verification.
     *
     * @var array
     */
    protected $except = [
        //
    ];
}
```

App/ Http/ Requests/ Admin/ StorePMSDocumentsRequest.php

```php
<?php
namespace App\Http\Requests\Admin;

use Illuminate\Foundation\Http\FormRequest;

/**
```

```
StorePMSDocumentsRequest.php (This applies to all the files under Requests/Admin folder)
StorePMSDocumentsRequest validates the form data obtained from the form submitted by the user.
It validates the data based on the rules written in the file. It successfully executes the
database operation if the validation is successful but denies the database operation if the
given rules are not validated. For instance, if a form field requires numeric data to be
filled but the user supplies special character then the application will issue a warning that
the input data is not valid.

*/

class StorePMSDocumentsRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'user_id' => 'required',
            'file' => 'required',
        ];
    }
}
```

App/ Http/ Requests/ Admin/ StorePMSNotesRequest.php

```
<?php
namespace App\Http\Requests\Admin;

use Illuminate\Foundation\Http\FormRequest;

/**
StorePMSDocumentsRequest.php (This applies to all the files under Requests/Admin folder)
StorePMSDocumentsRequest validates the form data obtained from the form submitted by the user.
It validates the data based on the rules written in the file. It successfully executes the
database operation if the validation is successful but denies the database operation if the
given rules are not validated. For instance, if a form field requires numeric data to be
filled but the user supplies special character then the application will issue a warning that
the input data is not valid.
*/
```

```
class StorePMSNotesRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'user_id' => 'required',
        ];
    }
}
```

App/ Http/ Requests/ Admin/ StoreRolesRequest.php

```
<?php
namespace App\Http\Requests\Admin;

use Illuminate\Foundation\Http\FormRequest;

/**
StorePMSDocumentsRequest.php (This applies to all the files under Requests/Admin folder)
StorePMSDocumentsRequest validates the form data obtained from the form submitted by the user.
It validates the data based on the rules written in the file. It successfully executes the
database operation if the validation is successful but denies the database operation if the
given rules are not validated. For instance, if a form field requires numeric data to be
filled but the user supplies special character then the application will issue a warning that
the input data is not valid.
*/

class StoreRolesRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'title' => 'required',
        ];
    }
}
```

App/ Http/ Requests/ Admin/ StoreUsersRequest.php

```
<?php
namespace App\Http\Requests\Admin;

use Illuminate\Foundation\Http\FormRequest;

/**
StorePMSDocumentsRequest.php (This applies to all the files under Requests/Admin folder)
StorePMSDocumentsRequest validates the form data obtained from the form submitted by the user.
It validates the data based on the rules written in the file. It successfully executes the
database operation if the validation is successful but denies the database operation if the
given rules are not validated. For instance, if a form field requires numeric data to be
filled but the user supplies special character then the application will issue a warning that
the input data is not valid.
*/

class StoreUsersRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'name' => 'required',
            'email' => 'required|email|unique:users,email',
            'password' => 'required',
            'role_id' => 'required',
        ];
    }
}
```

App/ Http/ Requests/ Admin/ UpdatePMSDocumentsRequest.php

```
<?php
namespace App\Http\Requests\Admin;

use Illuminate\Foundation\Http\FormRequest;

/**
StorePMSDocumentsRequest.php (This applies to all the files under Requests/Admin folder)
StorePMSDocumentsRequest validates the form data obtained from the form submitted by the user.
It validates the data based on the rules written in the file. It successfully executes the
database operation if the validation is successful but denies the database operation if the
given rules are not validated. For instance, if a form field requires numeric data to be
filled but the user supplies special character then the application will issue a warning that
the input data is not valid.
*/

class UpdatePMSDocumentsRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [

            'customer_id' => 'required',
        ];
    }
}
```

App/ Http/ Requests/ Admin/ UpdatePMSNotesRequest.php

```
<?php
namespace App\Http\Requests\Admin;

use Illuminate\Foundation\Http\FormRequest;
/**
StorePMSDocumentsRequest.php (This applies to all the files under Requests/Admin folder)
StorePMSDocumentsRequest validates the form data obtained from the form submitted by the user.
It validates the data based on the rules written in the file. It successfully executes the
database operation if the validation is successful but denies the database operation if the
given rules are not validated. For instance, if a form field requires numeric data to be
filled but the user supplies special character then the application will issue a warning that
the input data is not valid.
*/

class UpdatePMSNotesRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [

            'user_id' => 'required',
        ];
    }
}
```

App/ Http/ Requests/ Admin/ UpdateRolesRequest.php

```
<?php
namespace App\Http\Requests\Admin;

use Illuminate\Foundation\Http\FormRequest;

/**
```

```
StorePMSDocumentsRequest.php (This applies to all the files under Requests/Admin folder)
StorePMSDocumentsRequest validates the form data obtained from the form submitted by the user.
It validates the data based on the rules written in the file. It successfully executes the
database operation if the validation is successful but denies the database operation if the
given rules are not validated. For instance, if a form field requires numeric data to be
filled but the user supplies special character then the application will issue a warning that
the input data is not valid.
*/

class UpdateRolesRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [

            'title' => 'required',
        ];
    }
}
```

App/ Http/ Requests/ Admin/ UpdateUsersRequest.php

```
<?php
namespace App\Http\Requests\Admin;

use Illuminate\Foundation\Http\FormRequest;

/**
StorePMSDocumentsRequest.php (This applies to all the files under Requests/Admin folder)
StorePMSDocumentsRequest validates the form data obtained from the form submitted by the user.
It validates the data based on the rules written in the file. It successfully executes the
database operation if the validation is successful but denies the database operation if the
given rules are not validated. For instance, if a form field requires numeric data to be
filled but the user supplies special character then the application will issue a warning that
the input data is not valid.
*/
```

```
class UpdateUsersRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [

            'name' => 'required',
            'email' => 'required|email|unique:users,email,'.$this->route('user'),
            'role_id' => 'required',
        ];
    }
}
```

App/ Providers/ AppServiceProvider.php

```
<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Schema;
use Laravel\Dusk\DuskServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        // To fix the migration problem
        Schema::defaultStringLength(191);
    }

    /**
```

```
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        if ($this->app->environment('local', 'testing')) {
            $this->app->register(DuskServiceProvider::class);
        }

    }
}
```

App/ Providers/ AuthServiceProvider.php

```
<?php

namespace App\Providers;

use App\Role;
use App\User;
use Illuminate\Support\Facades\Gate;
use Illuminate\Foundation\Support\Providers\AuthServiceProvider as ServiceProvider;

class AuthServiceProvider extends ServiceProvider
{
    /**
     * The policy mappings for the application.
     *
     * @var array
     */
    protected $policies = [
        'App\Model' => 'App\Policies\ModelPolicy',
    ];

    /**
     * Register any authentication / authorization services.
     *
     * @return void
     */
    public function boot()
    {
        $this->registerPolicies();

        $user = \Auth::user();


        // Auth gates for: User management
        Gate::define('user_management_access', function ($user) {
            return in_array($user->role_id, [1]);
        });
```

```
// Auth gates for: Roles
Gate::define('role_access', function ($user) {
    return in_array($user->role_id, [1]);
});
Gate::define('role_create', function ($user) {
    return in_array($user->role_id, [1]);
});
Gate::define('role_edit', function ($user) {
    return in_array($user->role_id, [1]);
});
Gate::define('role_view', function ($user) {
    return in_array($user->role_id, [1]);
});
Gate::define('role_delete', function ($user) {
    return in_array($user->role_id, [1]);
});

// Auth gates for: Users
Gate::define('user_access', function ($user) {
    return in_array($user->role_id, [1]);
});
Gate::define('user_create', function ($user) {
    return in_array($user->role_id, [1]);
});
Gate::define('user_edit', function ($user) {
    return in_array($user->role_id, [1]);
});
Gate::define('user_view', function ($user) {
    return in_array($user->role_id, [1]);
});
Gate::define('user_delete', function ($user) {
    return in_array($user->role_id, [1]);
});

// Auth gates for: Basic PMS
Gate::define('basic_pms_access', function ($user) {
    return in_array($user->role_id, [1, 2]);
});

// Auth gates for: PMS statuses
Gate::define('pms_status_access', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_status_create', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_status_edit', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_status_view', function ($user) {
    return in_array($user->role_id, [1, 2]);
```

```
});
Gate::define('pms_status_delete', function ($user) {
    return in_array($user->role_id, [1]);
});

// Auth gates for: PMS customers
Gate::define('pms_customer_access', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_customer_create', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_customer_edit', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_customer_view', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_customer_delete', function ($user) {
    return in_array($user->role_id, [1]);
});

// Auth gates for: PMS notes
Gate::define('pms_note_access', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_note_create', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_note_edit', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_note_view', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_note_delete', function ($user) {
    return in_array($user->role_id, [1]);
});

// Auth gates for: PMS documents
Gate::define('pms_document_access', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_document_create', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_document_edit', function ($user) {
    return in_array($user->role_id, [1, 2]);
});
Gate::define('pms_document_view', function ($user) {
    return in_array($user->role_id, [1, 2]);
```

```
        });
        Gate::define('pms_document_delete', function ($user) {
            return in_array($user->role_id, [1]);
        });

    }
}
```

App/ Providers/ PMSDocument.php (Model)

```
<?php
namespace App;

use Illuminate\Database\Eloquent\Model;

/**
 * Class PMSDocument
 *
 * @package App
 * @property string $customer
 * @property string $name
 * @property text $description
 * @property string $file
*/

/**
PMSDocument.php (Same for all the models): This file consists of PMSDocument class which
inherits its functionalities from the Model class. PMSDocument consists of field names such as
name, description, file, user_id which are used to interact with the corresponding field names
in the database using the Eloquent ORM in Laravel. These field names are bound in the forms
which in turn carry the data from the user and interact with the database.
*/

class PMSDocument extends Model
{
    protected $fillable = ['name', 'description', 'file', 'user_id'];
    protected $hidden = [];



    /**
     * Set to null if empty
     * @param $input
     */
    public function setCustomerIdAttribute($input)
    {
        $this->attributes['user_id'] = $input ? $input : null;
    }

    public function user()
    {
        return $this->belongsTo(User::class, 'user_id');
    }

}
```

App/ PMSNote.php (Model)

```
<?php
namespace App;

use Illuminate\Database\Eloquent\Model;

/**
 * Class PMSNote
 *
 * @package App
 * @property string $customer
 * @property text $note
*/
class PMSNote extends Model
{
    protected $fillable = ['note', 'user_id'];
    protected $hidden = [];



    /**
     * Set to null if empty
     * @param $input
     */
    public function setCustomerIdAttribute($input)
    {
        $this->attributes['user_id'] = $input ? $input : null;
    }

    public function user()
    {
        return $this->belongsTo(User::class, 'user_id');
    }
```

App/ Role.php (Model)

```
<?php
namespace App;

use Illuminate\Database\Eloquent\Model;

/**
 * Class Role
 *
 * @package App
 * @property string $title
```

```
*/
class Role extends Model
{
    protected $fillable = ['title'];
    protected $hidden = [];
}
```

App/ User.php (Model)

```
<?php
namespace App;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Illuminate\Auth\Notifications\ResetPassword;
use Hash;

/**
 * Class User
 *
 * @package App
 * @property string $name
 * @property string $email
 * @property string $password
 * @property string $role
 * @property string $remember_token
*/
class User extends Authenticatable
{
    use Notifiable;
    protected $fillable = ['name', 'email', 'password', 'remember_token', 'role_id'];
    protected $hidden = ['password', 'remember_token'];


    /**
     * Hash password
     * @param $input
     */
    public function setPasswordAttribute($input)
    {
        if ($input)
            $this->attributes['password'] = app('hash')->needsRehash($input) ?
Hash::make($input) : $input;
    }

    /**
     * Set to null if empty
     * @param $input
     */
    public function setRoleIdAttribute($input)
    {
        $this->attributes['role_id'] = $input ? $input : null;
    }

    public function role()
    {
        return $this->belongsTo(Role::class, 'role_id');
    }

    public function sendPasswordResetNotification($token)
    {
       $this->notify(new ResetPassword($token));
    }

     /**
     * @return array
     */
    public function authorAttributes()
    {
        return [
            'name' => $this->name,
            'email' => $this->email,
            'url' => $this->url,    // optional
            'avatar' => 'gravatar', // optional
        ];
    }
}
```

Database/ factories/ PMSDocumentFactory.php

```
<?php

$factory->define(App\PMSDocument::class, function (Faker\Generator $faker) {
    return [
        "customer_id" => factory('App\PMSCustomer')->create(),
        "name" => $faker->name,
        "description" => $faker->name,
    ];
});
```

Database/ factories/ PMSNoteFactory.php

```
<?php

$factory->define(App\PMSNote::class, function (Faker\Generator $faker) {
    return [
        "customer_id" => factory('App\PMSCustomer')->create(),
        "note" => $faker->name,
    ];
});
```

Database/ factories/ RoleFactory.php

```
<?php
```

```
$factory->define(App\Role::class, function (Faker\Generator $faker) {
    return [
        "title" => $faker->name,
    ];
});
```

Database/ factories/ UserFactory.php

```
<?php

/*
|--------------------------------------------------------------------------
| Model Factories
|--------------------------------------------------------------------------
|
| This directory should contain each of the model factory definitions for
| your application. Factories provide a convenient way to generate new
| model instances for testing / seeding your application's database.
|
*/

$factory->define(App\User::class, function (Faker\Generator $faker) {
    return [
        "name" => $faker->name,
        "email" => $faker->safeEmail,
        "password" => str_random(10),
        "role_id" => factory('App\Role')->create(),
        "remember_token" => $faker->name,
    ];
});
```

Resources/ views/ admin/ pms_documents/ create.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.pms-documents.title')</h3>
    {!! Form::open(['method' => 'POST', 'route' => ['admin.pms_documents.store'], 'files' =>
true,]) !!}

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_create')
        </div>

        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('user_id', trans('pms.pms-documents.fields.member').'*',
['class' => 'control-label']) !!}
```

```
          {!! Form::select('user_id', $users, old('user_id'), ['class' => 'form-
control select2', 'required' => '']) !!}
          <p class="help-block"></p>
          @if($errors->has('user_id'))
            <p class="help-block">
              {{ $errors->first('user_id') }}
            </p>
          @endif
        </div>
      </div>
      <div class="row">
        <div class="col-xs-12 form-group">
          {!! Form::label('name', trans('pms.pms-documents.fields.name').'',
['class' => 'control-label']) !!}
          {!! Form::text('name', old('name'), ['class' => 'form-control',
'placeholder' => '']) !!}
          <p class="help-block"></p>
          @if($errors->has('name'))
            <p class="help-block">
              {{ $errors->first('name') }}
            </p>
          @endif
        </div>
      </div>
      <div class="row">
        <div class="col-xs-12 form-group">
          {!! Form::label('description', trans('pms.pms-
documents.fields.description').'', ['class' => 'control-label']) !!}
          {!! Form::textarea('description', old('description'), ['class' => 'form-
control ', 'placeholder' => '']) !!}
          <p class="help-block"></p>
          @if($errors->has('description'))
            <p class="help-block">
              {{ $errors->first('description') }}
            </p>
          @endif
        </div>
      </div>
      <div class="row">
        <div class="col-xs-12 form-group">
          {!! Form::label('file', trans('pms.pms-documents.fields.file').'*',
['class' => 'control-label']) !!}
          {!! Form::hidden('file', old('file')) !!}
          {!! Form::file('file', ['class' => 'form-control', 'required' => '']) !!}
          {!! Form::hidden('file_max_size', 10240) !!}
          <p class="help-block"></p>
          @if($errors->has('file'))
            <p class="help-block">
              {{ $errors->first('file') }}
            </p>
          @endif
```

```
                </div>
            </div>

        </div>
    </div>

    {!! Form::submit(trans('pms.pms_save'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ pms_documents/ edit.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.pms-documents.title')</h3>

    {!! Form::model($pms_document, ['method' => 'PUT', 'route' =>
['admin.pms_documents.update', $pms_document->id], 'files' => true,]) !!}

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_edit')
        </div>

        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('user_id', trans('pms.pms-documents.fields.member').'*',
['class' => 'control-label']) !!}
                    {!! Form::select('user_id', $users, old('user_id'), ['class' => 'form-
control select2', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('user_id'))
                        <p class="help-block">
                            {{ $errors->first('user_id') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('name', trans('pms.pms-documents.fields.name').'',
['class' => 'control-label']) !!}
                    {!! Form::text('name', old('name'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('name'))
                        <p class="help-block">
                            {{ $errors->first('name') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('description', trans('pms.pms-
documents.fields.description').'', ['class' => 'control-label']) !!}
                    {!! Form::textarea('description', old('description'), ['class' => 'form-
control ', 'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('description'))
                        <p class="help-block">
                            {{ $errors->first('description') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('file', trans('pms.pms-documents.fields.file').'*',
['class' => 'control-label']) !!}
                    {!! Form::hidden('file', old('file')) !!}
                    @if ($pms_document->file)
                        <a href="{{ asset(env('UPLOAD_PATH').'/' . $pms_document->file) }}"
target="_blank">Download file</a>
                    @endif
                    {!! Form::file('file', ['class' => 'form-control']) !!}
                    {!! Form::hidden('file_max_size', 10240) !!}
                    <p class="help-block"></p>
                    @if($errors->has('file'))
                        <p class="help-block">
                            {{ $errors->first('file') }}
                        </p>
                    @endif
                </div>
            </div>

        </div>
    </div>

    {!! Form::submit(trans('pms.pms_update'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ pms_documents/ index.blade.php

```
@inject('request', 'Illuminate\Http\Request')
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.pms-documents.title')</h3>
    @can('pms_document_create')
    <p>
        <a href="{{ route('admin.pms_documents.create') }}" class="btn btn-
success">@lang('pms.pms_add_new')</a>

    </p>
    @endcan

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_list')
        </div>

        <div class="panel-body table-responsive">
            <table class="table table-bordered table-striped {{ count($pms_documents) > 0 ?
'datatable' : '' }} @can('pms_document_delete') dt-select @endcan">
                <thead>
                    <tr>
                        @can('pms_document_delete')
                            <th style="text-align:center;"><input type="checkbox" id="select-
all" /></th>
                        @endcan

                        <th>@lang('pms.pms-documents.fields.member')</th>
                        <th>@lang('pms.pms-documents.fields.name')</th>
                        <th>@lang('pms.pms-documents.fields.description')</th>
                        <th>@lang('pms.pms-documents.fields.file')</th>
                                                <th>&nbsp;</th>

                    </tr>
                </thead>

                <tbody>
                    @if (count($pms_documents) > 0)
                        @foreach ($pms_documents as $pms_document)
                            <tr data-entry-id="{{ $pms_document->id }}">
                                @can('pms_document_delete')
                                    <td></td>
                                @endcan

                                <td field-key='user'>{{ $pms_document->user->name or ''
}}</td>
                                <td field-key='name'>{{ $pms_document->name }}</td>
                                <td field-key='description'>{!! $pms_document->description
!!}</td>
                                <td field-key='file'>@if($pms_document->file)<a href="{{
asset(env('UPLOAD_PATH').'/' . $pms_document->file) }}" target="_blank">Download
file</a>@endif</td>
                                                                <td>
                                    @can('pms_document_view')
                                    <a href="{{
route('admin.pms_documents.show',[$pms_document->id]) }}" class="btn btn-xs btn-
primary">@lang('pms.pms_view')</a>
                                    @endcan
                                    @can('pms_document_edit')
                                    <a href="{{
route('admin.pms_documents.edit',[$pms_document->id]) }}" class="btn btn-xs btn-
info">@lang('pms.pms_edit')</a>
                                    @endcan
                                    @can('pms_document_delete')
{!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'DELETE',
                                        'onsubmit' => "return
confirm('".trans("pms.pms_are_you_sure")."');",
                                        'route' => ['admin.pms_documents.destroy',
$pms_document->id])) !!}
                                    {!! Form::submit(trans('pms.pms_delete'), array('class' =>
'btn btn-xs btn-danger')) !!}
                                    {!! Form::close() !!}
                                    @endcan
                                </td>

                            </tr>
                        @endforeach
                    @else
                        <tr>
                            <td colspan="9">@lang('pms.pms_no_entries_in_table')</td>
                        </tr>
                    @endif
                </tbody>
            </table>
        </div>
    </div>
@stop

@section('javascript')
    <script>
        @can('pms_document_delete')
            window.route_mass_crud_entries_destroy = '{{
route('admin.pms_documents.mass_destroy') }}';
        @endcan

    </script>
@endsection
```

Resources/ views/ admin/ pms_documents/ show.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.pms-documents.title')</h3>
    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_view')
        </div>

        <div class="panel-body table-responsive">
            <div class="row">
                <div class="col-md-6">
                    <table class="table table-bordered table-striped">
                        <tr>
                            <th>@lang('pms.pms-documents.fields.member')</th>
                            <td field-key='customer'>{{ $pms_document->customer->first_name or
'' }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-documents.fields.name')</th>
                            <td field-key='name'>{{ $pms_document->name }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-documents.fields.description')</th>
                            <td field-key='description'>{!! $pms_document->description
!!}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-documents.fields.file')</th>
                            <td field-key='file'>@if($pms_document->file)<a href="{{
asset(env('UPLOAD_PATH').'/' . $pms_document->file) }}" target="_blank">Download
file</a>@endif</td>
                        </tr>
                    </table>
                </div>
            </div>

            <p>&nbsp;</p>

            <a href="{{ route('admin.pms_documents.index') }}" class="btn btn-
default">@lang('pms.pms_back_to_list')</a>
        </div>
    </div>
@stop
```

Resources/ views/ admin/ pms_forums/ create.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.pms-customers.title')</h3>
    {!! Form::open(['method' => 'POST', 'route' => ['admin.pms_customer.store']]) !!}

    <div class="panel panel-default">
        <div class="panel-heading">
```

```
            @lang('pms.pms_create')
        </div>

        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('first_name', trans('pms.pms-customers.fields.first-
name').'*', ['class' => 'control-label']) !!}
                    {!! Form::text('first_name', old('first_name'), ['class' => 'form-
control', 'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('first_name'))
                        <p class="help-block">
                            {{ $errors->first('first_name') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('last_name', trans('pms.pms-customers.fields.last-
name').'', ['class' => 'control-label']) !!}
                    {!! Form::text('last_name', old('last_name'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('last_name'))
                        <p class="help-block">
                            {{ $errors->first('last_name') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('pms_status_id', trans('pms.pms-customers.fields.pms-
status').'*', ['class' => 'control-label']) !!}
                    {!! Form::select('pms_status_id', $pms_statuses, old('pms_status_id'),
['class' => 'form-control select2', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('pms_status_id'))
                        <p class="help-block">
                            {{ $errors->first('pms_status_id') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('email', trans('pms.pms-customers.fields.email').'',
['class' => 'control-label']) !!}
```

```
                    {!! Form::email('email', old('email'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('email'))
                        <p class="help-block">
                            {{ $errors->first('email') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('phone', trans('pms.pms-customers.fields.phone').'',
['class' => 'control-label']) !!}
                    {!! Form::text('phone', old('phone'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('phone'))
                        <p class="help-block">
                            {{ $errors->first('phone') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('address', trans('pms.pms-customers.fields.address').'',
['class' => 'control-label']) !!}
                    {!! Form::text('address', old('address'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('address'))
                        <p class="help-block">
                            {{ $errors->first('address') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('skype', trans('pms.pms-customers.fields.skype').'',
['class' => 'control-label']) !!}
                    {!! Form::text('skype', old('skype'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('skype'))
                        <p class="help-block">
                            {{ $errors->first('skype') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('website', trans('pms.pms-customers.fields.website').'',
['class' => 'control-label']) !!}
                    {!! Form::text('website', old('website'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('website'))
                        <p class="help-block">
                            {{ $errors->first('website') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('description', trans('pms.pms-
customers.fields.description').'', ['class' => 'control-label']) !!}
                    {!! Form::textarea('description', old('description'), ['class' => 'form-
control ', 'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('description'))
                        <p class="help-block">
                            {{ $errors->first('description') }}
                        </p>
                    @endif
                </div>
            </div>

        </div>
    </div>

    {!! Form::submit(trans('pms.pms_save'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ pms_forums/ edit.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.pms-customers.title')</h3>

    {!! Form::model($pms_customer, ['method' => 'PUT', 'route' =>
['admin.pms_customers.update', $pms_customer->id]]) !!}

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.qa_edit')
        </div>
        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('first_name', trans('pms.pms-customers.fields.first-name').'*', ['class' => 'control-label']) !!}
                    {!! Form::text('first_name', old('first_name'), ['class' => 'form-control', 'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('first_name'))
                        <p class="help-block">
                            {{ $errors->first('first_name') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('last_name', trans('pms.pms-customers.fields.last-name').'', ['class' => 'control-label']) !!}
                    {!! Form::text('last_name', old('last_name'), ['class' => 'form-control', 'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('last_name'))
                        <p class="help-block">
                            {{ $errors->first('last_name') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('pms_status_id', trans('pms.pms-customers.fields.pms-status').'*', ['class' => 'control-label']) !!}
                    {!! Form::select('pms_status_id', $pms_statuses, old('pms_status_id'), ['class' => 'form-control select2', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('pms_status_id'))
                        <p class="help-block">
                            {{ $errors->first('pms_status_id') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('email', trans('pms.pms-customers.fields.email').'', ['class' => 'control-label']) !!}
                    {!! Form::email('email', old('email'), ['class' => 'form-control', 'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('email'))
```

```
                        <p class="help-block">
                            {{ $errors->first('email') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('phone', trans('pms.pms-customers.fields.phone').'',
['class' => 'control-label']) !!}
                    {!! Form::text('phone', old('phone'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('phone'))
                        <p class="help-block">
                            {{ $errors->first('phone') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('address', trans('pms.pms-customers.fields.address').'',
['class' => 'control-label']) !!}
                    {!! Form::text('address', old('address'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('address'))
                        <p class="help-block">
                            {{ $errors->first('address') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('skype', trans('pms.pms-customers.fields.skype').'',
['class' => 'control-label']) !!}
                    {!! Form::text('skype', old('skype'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('skype'))
                        <p class="help-block">
                            {{ $errors->first('skype') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
```

```
                    {!! Form::label('website', trans('pms.pms-customers.fields.website').'',
['class' => 'control-label']) !!}
                    {!! Form::text('website', old('website'), ['class' => 'form-control',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('website'))
                        <p class="help-block">
                            {{ $errors->first('website') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('description', trans('pms.pms-
customers.fields.description').'', ['class' => 'control-label']) !!}
                    {!! Form::textarea('description', old('description'), ['class' => 'form-
control ', 'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('description'))
                        <p class="help-block">
                            {{ $errors->first('description') }}
                        </p>
                    @endif
                </div>
            </div>

        </div>
    </div>

    {!! Form::submit(trans('pms.qa_update'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ pms_forums/ index.blade.php

```
@extends('layouts.app')

@section('content')
    <!-- Content Header (Page header) -->
    <section class="content-header">
      <h1>
        @lang('pms.pms_forum')
        <small>@lang('pms.pms_forum_text')</small>
      </h1>
      <ol class="breadcrumb">
        <li><a href="#"><i class="fa fa-dashboard"></i> Home</a></li>
        <li class="active">Forum</li>
      </ol>
    </section>

    <div class="pad margin no-print">
```

```
      <div class="callout callout-info" style="margin-bottom: 0!important;">
        <h4><i class="fa fa-info-circle"></i> &nbsp;
        Welcome {{{ isset(Auth::user()->name) ? Auth::user()->name : Auth::user()->email }}},
      </h4>
        to the discussion forum of project related topics with other PMS users.
      </div>
    </div>

    <!-- Main content -->
    <section class="invoice">
        <div id="forum-app"></div>
    </section>
    <!-- /.content -->
    <div class="clearfix"></div>
@endsection
```

Resources/ views/ admin/ pms_forums/ show.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.pms-customers.title')</h3>

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_view')
        </div>

        <div class="panel-body table-responsive">
            <div class="row">
                <div class="col-md-6">
                    <table class="table table-bordered table-striped">
                        <tr>
                            <th>@lang('pms.pms-customers.fields.first-name')</th>
                            <td field-key='first_name'>{{ $pms_customer->first_name }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-customers.fields.last-name')</th>
                            <td field-key='last_name'>{{ $pms_customer->last_name }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-customers.fields.pms-status')</th>
                            <td field-key='pms_status'>{{ $pms_customer->pms_status->title or
'' }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-customers.fields.email')</th>
                            <td field-key='email'>{{ $pms_customer->email }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-customers.fields.phone')</th>
                            <td field-key='phone'>{{ $pms_customer->phone }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-customers.fields.address')</th>
                            <td field-key='address'>{{ $pms_customer->address }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-customers.fields.skype')</th>
                            <td field-key='skype'>{{ $pms_customer->skype }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-customers.fields.website')</th>
                            <td field-key='website'>{{ $pms_customer->website }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-customers.fields.description')</th>
                            <td field-key='description'>{!! $pms_customer->description 
!!}</td>
                        </tr>
                    </table>
                </div>
            </div><!-- Nav tabs -->
<ul class="nav nav-tabs" role="tablist">

<li role="presentation" class="active"><a href="#pms_notes" aria-controls="pms_notes" 
role="tab" data-toggle="tab">Notes</a></li>
<li role="presentation" class=""><a href="#pms_documents" aria-controls="pms_documents" 
role="tab" data-toggle="tab">Documents</a></li>
</ul>

<!-- Tab panes -->
<div class="tab-content">

<div role="tabpanel" class="tab-pane active" id="pms_notes">
<table class="table table-bordered table-striped {{ count($pms_notes) > 0 ? 'datatable' : '' 
}}">
    <thead>
        <tr>
            <th>@lang('pms.pms-notes.fields.customer')</th>
                        <th>@lang('pms.pms-notes.fields.note')</th>
                                                <th>&nbsp;</th>

        </tr>
    </thead>

    <tbody>
        @if (count($pms_notes) > 0)
            @foreach ($pms_notes as $pms_note)
                <tr data-entry-id="{{ $pms_note->id }}">
                    <td field-key='customer'>{{ $pms_note->customer->first_name or '' }}</td>
                                <td field-key='note'>{!! $pms_note->note !!}</td>
                                                                <td>
```

```
                                    @can('view')
                                    <a href="{{ route('pms_notes.show',[$pms_note->id]) }}"
class="btn btn-xs btn-primary">@lang('pms.pms_view')</a>
                                    @endcan
                                    @can('edit')
                                    <a href="{{ route('pms_notes.edit',[$pms_note->id]) }}"
class="btn btn-xs btn-info">@lang('pms.pms_edit')</a>
                                    @endcan
                                    @can('delete')
{!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'DELETE',
                                        'onsubmit' => "return
confirm('".trans("pms.pms_are_you_sure")."');",
                                        'route' => ['pms_notes.destroy', $pms_note->id])) !!}
                                    {!! Form::submit(trans('pms.pms_delete'), array('class' =>
'btn btn-xs btn-danger')) !!}
                                    {!! Form::close() !!}
                                    @endcan
                                </td>

                </tr>
            @endforeach
        @else
            <tr>
                <td colspan="7">@lang('pms.pms_no_entries_in_table')</td>
            </tr>
        @endif
    </tbody>
</table>
</div>
<div role="tabpanel" class="tab-pane " id="pms_documents">
<table class="table table-bordered table-striped {{ count($pms_documents) > 0 ? 'datatable' :
'' }}">
    <thead>
        <tr>
            <th>@lang('pms.pms-documents.fields.customer')</th>
                        <th>@lang('pms.pms-documents.fields.name')</th>
                        <th>@lang('pms.pms-documents.fields.description')</th>
                        <th>@lang('pms.pms-documents.fields.file')</th>
                                                <th>&nbsp;</th>

        </tr>
    </thead>

    <tbody>
        @if (count($pms_documents) > 0)
            @foreach ($pms_documents as $pms_document)
                <tr data-entry-id="{{ $pms_document->id }}">
                    <td field-key='customer'>{{ $pms_document->customer->first_name or ''
}}</td>
```

```
                                <td field-key='name'>{{ $pms_document->name }}</td>
                                <td field-key='description'>{!! $pms_document->description
!!}</td>
                                <td field-key='file'>@if($pms_document->file)<a href="{{
asset(env('UPLOAD_PATH').'/' . $pms_document->file) }}" target="_blank">Download
file</a>@endif</td>
                                                                <td>
                                    @can('view')
                                    <a href="{{ route('pms_documents.show',[$pms_document-
>id]) }}" class="btn btn-xs btn-primary">@lang('pms.pms_view')</a>
                                    @endcan
                                    @can('edit')
                                    <a href="{{ route('pms_documents.edit',[$pms_document-
>id]) }}" class="btn btn-xs btn-info">@lang('pms.pms_edit')</a>
                                    @endcan
                                    @can('delete')
{!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'DELETE',
                                        'onsubmit' => "return
confirm('".trans("pms.pms_are_you_sure")."');",
                                        'route' => ['pms_documents.destroy', $pms_document-
>id])) !!}
                                    {!! Form::submit(trans('pms.pms_delete'), array('class' =>
'btn btn-xs btn-danger')) !!}
                                    {!! Form::close() !!}
                                    @endcan
                                </td>

                </tr>
            @endforeach
        @else
            <tr>
                <td colspan="9">@lang('pms.pms_no_entries_in_table')</td>
            </tr>
        @endif
    </tbody>
</table>
</div>
</div>

            <p>&nbsp;</p>

            <a href="{{ route('admin.pms_customers.index') }}" class="btn btn-
default">@lang('pms.pms_back_to_list')</a>
        </div>
    </div>
@stop
```

Resources/ views/ admin/ pms_notes/ create.blade.php

```
@extends('layouts.app')
```

```
@section('content')
    <h3 class="page-title">@lang('pms.pms-notes.title')</h3>
    {!! Form::open(['method' => 'POST', 'route' => ['admin.pms_notes.store']]) !!}

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_create')
        </div>

        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('user_id', trans('pms.pms-documents.fields.member').'*',
['class' => 'control-label']) !!}
                    {!! Form::select('user_id', $users, old('user_id'), ['class' => 'form-
control select2', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('user_id'))
                        <p class="help-block">
                            {{ $errors->first('user_id') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('note', trans('pms.pms-notes.fields.note').'', ['class' =>
'control-label']) !!}
                    {!! Form::textarea('note', old('note'), ['class' => 'form-control ',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('note'))
                        <p class="help-block">
                            {{ $errors->first('note') }}
                        </p>
                    @endif
                </div>
            </div>

        </div>
    </div>

    {!! Form::submit(trans('pms.pms_save'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ pms_notes/ edit.blade.php

```
@extends('layouts.app')

@section('content')
```

```
    <h3 class="page-title">@lang('pms.pms-notes.title')</h3>

    {!! Form::model($pms_note, ['method' => 'PUT', 'route' => ['admin.pms_notes.update',
$pms_note->id]]) !!}

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_edit')
        </div>

        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('user_id', trans('pms.pms-documents.fields.member').'*',
['class' => 'control-label']) !!}
                    {!! Form::select('user_id', $users, old('user_id'), ['class' => 'form-
control select2', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('user_id'))
                        <p class="help-block">
                            {{ $errors->first('user_id') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('note', trans('pms.pms-notes.fields.note').'', ['class' =>
'control-label']) !!}
                    {!! Form::textarea('note', old('note'), ['class' => 'form-control ',
'placeholder' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('note'))
                        <p class="help-block">
                            {{ $errors->first('note') }}
                        </p>
                    @endif
                </div>
            </div>

        </div>
    </div>

    {!! Form::submit(trans('pms.pms_update'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ pms_notes/ index.blade.php

```
@inject('request', 'Illuminate\Http\Request')
@extends('layouts.app')
```

```
@section('content')
    <h3 class="page-title">@lang('pms.pms-notes.title')</h3>
    @can('pms_note_create')
    <p>
        <a href="{{ route('admin.pms_notes.create') }}" class="btn btn-
success">@lang('pms.pms_add_new')</a>

    </p>
    @endcan



    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_list')
        </div>

        <div class="panel-body table-responsive">
            <table class="table table-bordered table-striped {{ count($pms_notes) > 0 ?
'datatable' : '' }} @can('pms_note_delete') dt-select @endcan">
                <thead>
                    <tr>
                        @can('pms_note_delete')
                            <th style="text-align:center;"><input type="checkbox" id="select-
all" /></th>
                        @endcan

                        <th>@lang('pms.pms-documents.fields.member')</th>
                        <th>@lang('pms.pms-notes.fields.note')</th>
                                                <th>&nbsp;</th>

                    </tr>
                </thead>

                <tbody>
                    @if (count($pms_notes) > 0)
                        @foreach ($pms_notes as $pms_note)
                            <tr data-entry-id="{{ $pms_note->id }}">
                                @can('pms_note_delete')
                                    <td></td>
                                @endcan

                                <td field-key='user'>{{ $pms_note->user->name or '' }}</td>
                                <td field-key='note'>{!! $pms_note->note !!}</td>
                                                                <td>
                                    @can('pms_note_view')
                                    <a href="{{ route('admin.pms_notes.show',[$pms_note->id])
}}" class="btn btn-xs btn-primary">@lang('pms.pms_view')</a>
                                    @endcan
                                    @can('pms_note_edit')
                                    <a href="{{ route('admin.pms_notes.edit',[$pms_note->id])
}}" class="btn btn-xs btn-info">@lang('pms.pms_edit')</a>
                                    @endcan
                                    @can('pms_note_delete')
{!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'DELETE',
                                        'onsubmit' => "return
confirm('".trans("pms.pms_are_you_sure")."');",
                                        'route' => ['admin.pms_notes.destroy', $pms_note-
>id])) !!}
                                    {!! Form::submit(trans('pms.pms_delete'), array('class' =>
'btn btn-xs btn-danger')) !!}
                                    {!! Form::close() !!}
                                    @endcan
                                </td>

                            </tr>
                        @endforeach
                    @else
                        <tr>
                            <td colspan="7">@lang('pms.pms_no_entries_in_table')</td>
                        </tr>
                    @endif
                </tbody>
            </table>
        </div>
    </div>
@stop

@section('javascript')
    <script>
        @can('pms_note_delete')
            window.route_mass_crud_entries_destroy = '{{ route('admin.pms_notes.mass_destroy')
}}';
        @endcan

    </script>
@endsection
```

Resources/ views/ admin/ pms_notes/ show.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.pms-notes.title')</h3>

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_view')
        </div>
        <div class="panel-body table-responsive">
            <div class="row">
                <div class="col-md-6">
                    <table class="table table-bordered table-striped">
                        <tr>
                            <th>@lang('pms.pms-documents.fields.member')</th>
                            <td field-key='user'>{{ $pms_note->user->name or '' }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.pms-notes.fields.note')</th>
                            <td field-key='note'>{!! $pms_note->note !!}</td>
                        </tr>
                    </table>
                </div>
            </div>

            <p>&nbsp;</p>

            <a href="{{ route('admin.pms_notes.index') }}" class="btn btn-
default">@lang('pms.pms_back_to_list')</a>
        </div>
    </div>
@stop
```

Resources/ views/ admin/ roles/ create.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.roles.title')</h3>
    {!! Form::open(['method' => 'POST', 'route' => ['admin.roles.store']]) !!}

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_create')
        </div>

        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('title', trans('pms.roles.fields.title').'*', ['class' =>
'control-label']) !!}
                    {!! Form::text('title', old('title'), ['class' => 'form-control',
'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('title'))
                        <p class="help-block">
                            {{ $errors->first('title') }}
                        </p>
                    @endif
                </div>
            </div>
        </div>
    </div>

    {!! Form::submit(trans('pms.pms_save'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ roles/ edit.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.roles.title')</h3>

    {!! Form::model($role, ['method' => 'PUT', 'route' => ['admin.roles.update', $role->id]])
!!}

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_edit')
        </div>

        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('title', trans('pms.roles.fields.title').'*', ['class' =>
'control-label']) !!}
                    {!! Form::text('title', old('title'), ['class' => 'form-control',
'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('title'))
                        <p class="help-block">
                            {{ $errors->first('title') }}
                        </p>
                    @endif
                </div>
            </div>

        </div>
    </div>

    {!! Form::submit(trans('pms.pms_update'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ roles/ index.blade.php

```
@inject('request', 'Illuminate\Http\Request')
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.roles.title')</h3>
    @can('role_create')
    <p>
        <a href="{{ route('admin.roles.create') }}" class="btn btn-
success">@lang('pms.pms_add_new')</a>

    </p>
    @endcan



    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_list')
        </div>

        <div class="panel-body table-responsive">
            <table class="table table-bordered table-striped {{ count($roles) > 0 ?
'datatable' : '' }} @can('role_delete') dt-select @endcan">
                <thead>
                    <tr>
                        @can('role_delete')
                            <th style="text-align:center;"><input type="checkbox" id="select-
all" /></th>
                        @endcan

                        <th>@lang('pms.roles.fields.title')</th>
                                                <th>&nbsp;</th>

                    </tr>
                </thead>

                <tbody>
                    @if (count($roles) > 0)
                        @foreach ($roles as $role)
                            <tr data-entry-id="{{ $role->id }}">
                                @can('role_delete')
                                    <td></td>
                                @endcan

                                <td field-key='title'>{{ $role->title }}</td>
                                                                <td>
                                    @can('role_view')
                                    <a href="{{ route('admin.roles.show',[$role->id]) }}"
class="btn btn-xs btn-primary">@lang('pms.pms_view')</a>
                                    @endcan
                                    @can('role_edit')
                                    <a href="{{ route('admin.roles.edit',[$role->id]) }}"
class="btn btn-xs btn-info">@lang('pms.pms_edit')</a>
                                    @endcan
                                    @can('role_delete')
{!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'DELETE',
                                        'onsubmit' => "return
confirm('".trans("pms.pms_are_you_sure")."');",
                                        'route' => ['admin.roles.destroy', $role->id])) !!}
                                    {!! Form::submit(trans('pms.pms_delete'), array('class' =>
'btn btn-xs btn-danger')) !!}
                                    {!! Form::close() !!}
                                    @endcan
                                </td>

                            </tr>
                        @endforeach
                    @else
                        <tr>
                            <td colspan="6">@lang('pms.pms_no_entries_in_table')</td>
                        </tr>
                    @endif
                </tbody>
            </table>
        </div>
    </div>
@stop

@section('javascript')
    <script>
        @can('role_delete')
            window.route_mass_crud_entries_destroy = '{{ route('admin.roles.mass_destroy')
}}';
        @endcan

    </script>
@endsection
```

Resources/ views/ admin/ roles/ show.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.roles.title')</h3>

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_view')
        </div>

        <div class="panel-body table-responsive">
            <div class="row">
                <div class="col-md-6">
                    <table class="table table-bordered table-striped">
                        <tr>
                            <th>@lang('pms.roles.fields.title')</th>
                            <td field-key='title'>{{ $role->title }}</td>
                        </tr>
                    </table>
                </div>
            </div><!-- Nav tabs -->
<ul class="nav nav-tabs" role="tablist">

<li role="presentation" class="active"><a href="#users" aria-controls="users" role="tab" data-
toggle="tab">Users</a></li>
</ul>

<!-- Tab panes -->
<div class="tab-content">

<div role="tabpanel" class="tab-pane active" id="users">
<table class="table table-bordered table-striped {{ count($users) > 0 ? 'datatable' : '' }}">
    <thead>
        <tr>
            <th>@lang('pms.users.fields.name')</th>
                        <th>@lang('pms.users.fields.email')</th>
                        <th>@lang('pms.users.fields.role')</th>
                                                <th>&nbsp;</th>

        </tr>
    </thead>

    <tbody>
        @if (count($users) > 0)
            @foreach ($users as $user)
                <tr data-entry-id="{{ $user->id }}">
                    <td field-key='name'>{{ $user->name }}</td>
                                <td field-key='email'>{{ $user->email }}</td>
                                <td field-key='role'>{{ $user->role->title or '' }}</td>
                                                                <td>
                                    @can('view')
                                    <a href="{{ route('users.show',[$user->id]) }}" class="btn
btn-xs btn-primary">@lang('pms.pms_view')</a>
                                    @endcan
                                    @can('edit')
                                    <a href="{{ route('users.edit',[$user->id]) }}" class="btn
btn-xs btn-info">@lang('pms.pms_edit')</a>
                                    @endcan
                                    @can('delete')
{!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'DELETE',
                                        'onsubmit' => "return
confirm('".trans("pms.pms_are_you_sure")."');",
                                        'route' => ['users.destroy', $user->id])) !!}
                                    {!! Form::submit(trans('pms.pms_delete'), array('class' =>
'btn btn-xs btn-danger')) !!}
```

```
                                    {!! Form::close() !!}
                                    @endcan
                                </td>

                </tr>
            @endforeach
        @else
            <tr>
                <td colspan="10">@lang('pms.pms_no_entries_in_table')</td>
            </tr>
        @endif
    </tbody>
</table>
</div>
</div>

            <p>&nbsp;</p>

            <a href="{{ route('admin.roles.index') }}" class="btn btn-
default">@lang('pms.pms_back_to_list')</a>
        </div>
    </div>
@stop
```

Resources/ views/ admin/ users/ create.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.users.title')</h3>
    {!! Form::open(['method' => 'POST', 'route' => ['admin.users.store']]) !!}

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_create')
        </div>

        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('name', trans('pms.users.fields.name').'*', ['class' =>
'control-label']) !!}
                    {!! Form::text('name', old('name'), ['class' => 'form-control',
'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('name'))
                        <p class="help-block">
                            {{ $errors->first('name') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('email', trans('pms.users.fields.email').'*', ['class' =>
'control-label']) !!}
                    {!! Form::email('email', old('email'), ['class' => 'form-control',
'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('email'))
                        <p class="help-block">
                            {{ $errors->first('email') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('password', trans('pms.users.fields.password').'*',
['class' => 'control-label']) !!}
                    {!! Form::password('password', ['class' => 'form-control', 'placeholder'
=> '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('password'))
                        <p class="help-block">
                            {{ $errors->first('password') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('role_id', trans('pms.users.fields.role').'*', ['class' =>
'control-label']) !!}
                    {!! Form::select('role_id', $roles, old('role_id'), ['class' => 'form-
control select2', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('role_id'))
                        <p class="help-block">
                            {{ $errors->first('role_id') }}
                        </p>
                    @endif
                </div>
            </div>

        </div>
    </div>

    {!! Form::submit(trans('pms.pms_save'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ users/ edit.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.users.title')</h3>

    {!! Form::model($user, ['method' => 'PUT', 'route' => ['admin.users.update', $user->id]])
!!}

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_edit')
        </div>

        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('name', trans('pms.users.fields.name').'*', ['class' =>
'control-label']) !!}
                    {!! Form::text('name', old('name'), ['class' => 'form-control',
'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('name'))
                        <p class="help-block">
                            {{ $errors->first('name') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('email', trans('pms.users.fields.email').'*', ['class' =>
'control-label']) !!}
                    {!! Form::email('email', old('email'), ['class' => 'form-control',
'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('email'))
                        <p class="help-block">
                            {{ $errors->first('email') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('password', trans('pms.users.fields.password').'*',
['class' => 'control-label']) !!}
                    {!! Form::password('password', ['class' => 'form-control', 'placeholder'
=> '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('password'))
```

```
                        <p class="help-block">
                            {{ $errors->first('password') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('role_id', trans('pms.users.fields.role').'*', ['class' =>
'control-label']) !!}
                    {!! Form::select('role_id', $roles, old('role_id'), ['class' => 'form-
control select2', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('role_id'))
                        <p class="help-block">
                            {{ $errors->first('role_id') }}
                        </p>
                    @endif
                </div>
            </div>

        </div>
    </div>

    {!! Form::submit(trans('pms.pms_update'), ['class' => 'btn btn-danger']) !!}
    {!! Form::close() !!}
@stop
```

Resources/ views/ admin/ users/ index.blade.php

```
@inject('request', 'Illuminate\Http\Request')
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.users.title')</h3>
    @can('user_create')
    <p>
        <a href="{{ route('admin.users.create') }}" class="btn btn-
success">@lang('pms.pms_add_new')</a>

    </p>
    @endcan



    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_list')
        </div>

        <div class="panel-body table-responsive">
```

```
            <table class="table table-bordered table-striped {{ count($users) > 0 ?
'datatable' : '' }} @can('user_delete') dt-select @endcan">
                <thead>
                    <tr>
                        @can('user_delete')
                            <th style="text-align:center;"><input type="checkbox" id="select-
all" /></th>
                        @endcan

                        <th>@lang('pms.users.fields.name')</th>
                        <th>@lang('pms.users.fields.email')</th>
                        <th>@lang('pms.users.fields.role')</th>
                                                <th>&nbsp;</th>

                    </tr>
                </thead>

                <tbody>
                    @if (count($users) > 0)
                        @foreach ($users as $user)
                            <tr data-entry-id="{{ $user->id }}">
                                @can('user_delete')
                                    <td></td>
                                @endcan

                                <td field-key='name'>{{ $user->name }}</td>
                                <td field-key='email'>{{ $user->email }}</td>
                                <td field-key='role'>{{ $user->role->title or '' }}</td>
                                                                <td>
                                    @can('user_view')
                                    <a href="{{ route('admin.users.show',[$user->id]) }}"
class="btn btn-xs btn-primary">@lang('pms.pms_view')</a>
                                    @endcan
                                    @can('user_edit')
                                    <a href="{{ route('admin.users.edit',[$user->id]) }}"
class="btn btn-xs btn-info">@lang('pms.pms_edit')</a>
                                    @endcan
                                    @can('user_delete')
{!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'DELETE',
                                        'onsubmit' => "return
confirm('".trans("pms.pms_are_you_sure")."');",
                                        'route' => ['admin.users.destroy', $user->id])) !!}
                                    {!! Form::submit(trans('pms.pms_delete'), array('class' =>
'btn btn-xs btn-danger')) !!}
                                    {!! Form::close() !!}
                                    @endcan
                                </td>

                            </tr>
                        @endforeach
                    @else
                        <tr>
                            <td colspan="10">@lang('pms.pms_no_entries_in_table')</td>
                        </tr>
                    @endif
                </tbody>
            </table>
        </div>
    </div>
@stop

@section('javascript')
    <script>
        @can('user_delete')
            window.route_mass_crud_entries_destroy = '{{ route('admin.users.mass_destroy')
}}';
        @endcan

    </script>
@endsection
```

Resources/ views/ admin/ users/ show.blade.php

```
@extends('layouts.app')

@section('content')
    <h3 class="page-title">@lang('pms.users.title')</h3>

    <div class="panel panel-default">
        <div class="panel-heading">
            @lang('pms.pms_view')
        </div>

        <div class="panel-body table-responsive">
            <div class="row">
                <div class="col-md-6">
                    <table class="table table-bordered table-striped">
                        <tr>
                            <th>@lang('pms.users.fields.name')</th>
                            <td field-key='name'>{{ $user->name }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.users.fields.email')</th>
                            <td field-key='email'>{{ $user->email }}</td>
                        </tr>
                        <tr>
                            <th>@lang('pms.users.fields.role')</th>
                            <td field-key='role'>{{ $user->role->title or '' }}</td>
                        </tr>
                    </table>
                </div>
            </div>

            <p>&nbsp;</p>

            <a href="{{ route('admin.users.index') }}" class="btn btn-
default">@lang('pms.pms_back_to_list')</a>
        </div>
    </div>
@stop
```

Resources/ views/layouts/ app.blade.php

```
@extends('layouts.auth')

@section('content')
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="panel panel-default">
                <div class="panel-heading">{{ ucfirst(config('app.name')) }}
@lang('pms.pms_login')</div>
                <div class="panel-body">

                    @if (count($errors) > 0)
                        <div class="alert alert-danger">
                            <strong>@lang('pms.pms_whoops')</strong>
@lang('pms.pms_there_were_problems_with_input'):
                            <br><br>
                            <ul>
                                @foreach ($errors->all() as $error)
                                    <li>{{ $error }}</li>
                                @endforeach
                            </ul>
                        </div>
                    @endif

                    <form class="form-horizontal"
                          role="form"
                          method="POST"
                          action="{{ url('login') }}">
                        <input type="hidden"
                               name="_token"
                               value="{{ csrf_token() }}">

                        <div class="form-group">
                            <label class="col-md-4 control-
label">@lang('pms.pms_email')</label>

                            <div class="col-md-6">
                                <input type="email"
                                       class="form-control"
                                       name="email"
                                       value="{{ old('email') }}">
                            </div>
                        </div>

                        <div class="form-group">
                            <label class="col-md-4 control-
label">@lang('pms.pms_password')</label>

                            <div class="col-md-6">
                                <input type="password"
                                       class="form-control"
                                       name="password">
                            </div>
                        </div>

                        <div class="form-group">
                            <div class="col-md-6 col-md-offset-4">
                                <a href="{{ route('auth.password.reset')
}}">@lang('pms.pms_forgot_password')</a>
                            </div>
                        </div>

                        <div class="form-group">
                            <div class="col-md-6 col-md-offset-4">
                                <label>
                                    <input type="checkbox"
                                           name="remember"> @lang('pms.pms_remember_me')
                                </label>
                            </div>
                        </div>

                        <div class="form-group">
                            <div class="col-md-6 col-md-offset-4">
                                <button type="submit"
                                        class="btn btn-primary"
                                        style="margin-right: 15px;">
                                    @lang('pms.pms_login')
                                </button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
@endsection
```

Resources/ views/layouts/ auth.blade.php

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    @include('partials.head')
</head>

<body class="page-header-fixed">

    <div style="margin-top: 10%;"></div>

    <div class="container-fluid">
        @yield('content')
    </div>

    <div class="scroll-to-top"
         style="display: none;">
        <i class="fa fa-arrow-up"></i>
    </div>

    @include('partials.javascripts')

</body>
</html>
```

Resources/ views/ partials/ footer.blade.php

```
  <footer class="main-footer">
    <div class="pull-right hidden-xs">
      <b>Project Management System</b> (PMS)
    </div>
    <strong>Copyright &copy; <?php echo date('Y'); ?> <a href="#">Alon Poudel</a>.</strong>
All rights
    reserved.
  </footer>
```

Resources/ views/ partials/ head.blade.php

```
<meta charset="utf-8">
<title>
    @lang('pms.pms_title')
</title>

<meta http-equiv="X-UA-Compatible"
      content="IE=edge">
<meta content="width=device-width, initial-scale=1.0"
      name="viewport"/>
<meta http-equiv="Content-type"
      content="text/html; charset=utf-8">

<!-- Tell the browser to be responsive to screen width -->
<meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no"
name="viewport">
<!-- Font Awesome -->
```

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
<!-- Ionicons -->
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/ionicons/2.0.1/css/ionicons.min.css">
<!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
<!--[if lt IE 9]>
<script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
<script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
<![endif]-->

<link href="{{ url('adminlte/bootstrap/css/bootstrap.min.css') }}" rel="stylesheet">
<link rel="stylesheet"
      href="{{ url('assets/css') }}/select2.min.css"/>
<link href="{{ url('adminlte/css/AdminLTE.min.css') }}" rel="stylesheet">
<link href="{{ url('adminlte/css/custom.css') }}" rel="stylesheet">
<link href="{{ url('adminlte/css/skins/skin-blue.min.css') }}" rel="stylesheet">
<link rel="stylesheet"
      href="https://code.jquery.com/ui/1.11.3/themes/smoothness/jquery-ui.css">
<link rel="stylesheet"
      href="//cdn.datatables.net/1.10.9/css/jquery.dataTables.min.css"/>
<link rel="stylesheet"
      href="https://cdn.datatables.net/select/1.2.0/css/select.dataTables.min.css"/>
<link rel="stylesheet"
      href="//cdn.datatables.net/buttons/1.2.4/css/buttons.dataTables.min.css"/>

<link rel="stylesheet" href="{{ url('adminlte/plugins/datetimepicker/bootstrap-
datetimepicker.min.css') }}">
<link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css">
<link href="{{ mix('comments.css', 'vendor/comments') }}" rel="stylesheet">
```

Resources/ views/ partials/ header.blade.php

```
<div class="page-header-inner">
    <div class="page-header-inner">
        <div class="navbar-header">
            <a href="{{ url('/') }}"
               class="navbar-brand">
                @lang('pms.pms_title')
            </a>
        </div>
        <a href="javascript:;"
           class="menu-toggler responsive-toggler"
           data-toggle="collapse"
           data-target=".navbar-collapse">
        </a>

        <div class="top-menu">
            <ul class="nav navbar-nav pull-right">
            </ul>
        </div>
    </div>
</div>
```

Resources/ views/ partials/ javascripts.blade.php

```
<script>
    window.deleteButtonTrans = '{{ trans("pms.pms_delete_selected") }}';
    window.copyButtonTrans = '{{ trans("pms.pms_copy") }}';
    window.csvButtonTrans = '{{ trans("pms.pms_csv") }}';
    window.excelButtonTrans = '{{ trans("pms.pms_excel") }}';
    window.pdfButtonTrans = '{{ trans("pms.pms_pdf") }}';
    window.printButtonTrans = '{{ trans("pms.pms_print") }}';
    window.colvisButtonTrans = '{{ trans("pms.pms_colvis") }}';
</script>
<script src="//code.jquery.com/jquery-1.11.3.min.js"></script>
<script src="//cdn.datatables.net/1.10.9/js/jquery.dataTables.min.js"></script>
<script src="//cdn.datatables.net/buttons/1.2.4/js/dataTables.buttons.min.js"></script>
<script src="//cdn.datatables.net/buttons/1.2.4/js/buttons.flash.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js"></script>
<script src="https://cdn.rawgit.com/bpampuch/pdfmake/0.1.18/build/pdfmake.min.js"></script>
<script src="https://cdn.rawgit.com/bpampuch/pdfmake/0.1.18/build/vfs_fonts.js"></script>
<script src="https://cdn.datatables.net/buttons/1.2.4/js/buttons.html5.min.js"></script>
<script src="https://cdn.datatables.net/buttons/1.2.4/js/buttons.print.min.js"></script>
<script src="https://cdn.datatables.net/buttons/1.2.4/js/buttons.colVis.min.js"></script>
<script src="https://cdn.datatables.net/select/1.2.0/js/dataTables.select.min.js"></script>
<script src="https://code.jquery.com/ui/1.11.3/jquery-ui.min.js"></script>
<script src="{{ url('adminlte/js') }}/bootstrap.min.js"></script>
<script src="{{ url('adminlte/js') }}/select2.full.min.js"></script>
<script src="{{ url('adminlte/js') }}/main.js"></script>
<script src="{{ url('adminlte/plugins/slimScroll/jquery.slimscroll.min.js') }}"></script>
<script src="{{ url('adminlte/plugins/fastclick/fastclick.js') }}"></script>
<script src="{{ url('adminlte/js/app.min.js') }}"></script>
<!-- <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.9.0/jquery.min.js"></script> -->
        <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/jquery.textcomplete/1.8.0/jquery.textcomplete.js">
</script>
<script src="{{ mix('comments.js', 'vendor/comments') }}"></script>
<script>
    window._token = '{{ csrf_token() }}';
</script>
<script>
    $.extend(true, $.fn.dataTable.defaults, {
        "language": {
            "url": "https://cdn.datatables.net/plug-ins/1.10.16/i18n/English.json"
        }
    });
</script>
```

```
<script>
    new Comments.default({
        el: '#forum-app',
        pageId: 'p_id'
    })
</script>

@yield('javascript')
```

Resources/ views/ partials/ sidebar.blade.php

```
@inject('request', 'Illuminate\Http\Request')
<!-- Left side column. contains the sidebar -->
<aside class="main-sidebar">
    <!-- sidebar: style can be found in sidebar.less -->
    <section class="sidebar">
        <ul class="sidebar-menu">



            <li class="{{ $request->segment(1) == 'home' ? 'active' : '' }}">
                <a href="{{ url('/') }}">
                    <i class="fa fa-dashboard"></i>
                    <span class="title">@lang('pms.pms_dashboard')</span>
                </a>
            </li>

            @can('user_management_access')
            <li class="treeview">
                <a href="#">
                    <i class="fa fa-users"></i>
                    <span>@lang('pms.user-management.title')</span>
                    <span class="pull-right-container">
                        <i class="fa fa-angle-left pull-right"></i>
                    </span>
                </a>
                <ul class="treeview-menu">
                    @can('role_access')
                    <li>
                        <a href="{{ route('admin.roles.index') }}">
                            <i class="fa fa-briefcase"></i>
                            <span>@lang('pms.roles.title')</span>
                        </a>
                    </li>@endcan

                    @can('user_access')
                    <li>
                        <a href="{{ route('admin.users.index') }}">
                            <i class="fa fa-user"></i>
                            <span>@lang('pms.users.title')</span>
                        </a>
                    </li>@endcan

                </ul>
            </li>@endcan

            @can('basic_pms_access')
            <li class="treeview">
                <a href="#">
                    <i class="fa fa-briefcase"></i>
                    <span>@lang('pms.basic-pms.title')</span>
                    <span class="pull-right-container">
                        <i class="fa fa-angle-left pull-right"></i>
                    </span>
                </a>
                <ul class="treeview-menu">
                    @can('pms_note_access')
                    <li>
                        <a href="{{ route('admin.pms_notes.index') }}">
                            <i class="fa fa-building-o"></i>
                            <span>@lang('pms.pms-notes.title')</span>
                        </a>
                    </li>@endcan

                    @can('pms_document_access')
                    <li>
                        <a href="{{ route('admin.pms_documents.index') }}">
                            <i class="fa fa-file"></i>
                            <span>@lang('pms.pms-documents.title')</span>
                        </a>
                    </li>@endcan

                </ul>
            </li>@endcan
 <li class="{{ $request->segment(1) == 'pms_forum' ? 'active' : '' }}">
                <a href="{{ route('admin.pms_forum.index') }}">
                    <i class="fa fa-group"></i>
                    <span class="title">@lang('pms.pms_forum')</span>
                </a>
            </li>
            <li class="{{ $request->segment(1) == 'change_password' ? 'active' : '' }}">
                <a href="{{ route('auth.change_password') }}">
                    <i class="fa fa-lock"></i>
                    <span class="title">@lang('pms.pms_change_password')</span>
                </a>
            </li>

            <li>
                <a href="#logout" onclick="$('#logout').submit();">
                    <i class="fa fa-sign-out"></i>
                    <span class="title">@lang('pms.pms_logout')</span>
                </a>
            </li>
        </ul>
    </section>
</aside>
```

Resources/ views/ partials/ topbar.blade.php

```
<header class="main-header">
    <!-- Logo -->
    <a href="{{ url('/admin/home') }}" class="logo"
       style="font-size: 15px;">
        <!-- mini logo for sidebar mini 50x50 pixels -->
        <span class="logo-mini">
           @lang('pms.pms_title')</span>
        <!-- logo for regular state and mobile devices -->
        <span class="logo-lg">
           @lang('pms.pms_title')</span>
    </a>
    <!-- Header Navbar: style can be found in header.less -->
    <nav class="navbar navbar-static-top">
        <!-- Sidebar toggle button-->
        <a href="#" class="sidebar-toggle" data-toggle="offcanvas" role="button">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
        </a>
    </nav>
</header>
```

Resources/ views/ home.blade.php

```
@extends('layouts.app')
  @section('content')
  <!-- Content Wrapper. Contains page content -->
    <!-- Content Header (Page header) -->
    <section class="content-header">
      <h1>
        @lang('pms.pms_dashboard')
        <small>Welcome, <b>{{{ isset(Auth::user()->name) ? Auth::user()->name : Auth::user()-
>email }}} </b> !</small>
      </h1>
      <ol class="breadcrumb">
        <li><a href="#"><i class="fa fa-dashboard"></i> Home</a></li>
        <li class="active">Dashboard</li>
      </ol>
    </section>

    <!-- Main content -->
    <section class="content">
```

```
      <!-- Small boxes (Stat box) -->
      <div class="row">

        <!-- ./col -->
      </div>
      <!-- /.row -->

    </section>
    <!-- /.content -->
@endsection
  <!-- /.content-wrapper -->
```

Routes/ web.php

```
<?php

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

/* Route::get('/', function () {
    return view('welcome');
});

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

Route::resource('category', 'CategoryController');

*/

Route::get('/', function () { return redirect('/admin/home'); });

// Authentication Routes...
$this->get('login', 'Auth\LoginController@showLoginForm')->name('auth.login');
$this->post('login', 'Auth\LoginController@login')->name('auth.login');
$this->post('logout', 'Auth\LoginController@logout')->name('auth.logout');

// Change Password Routes...
$this->get('change_password', 'Auth\ChangePasswordController@showChangePasswordForm')-
>name('auth.change_password');
```

```
$this->patch('change_password', 'Auth\ChangePasswordController@changePassword')-
>name('auth.change_password');

// Password Reset Routes...
$this->get('password/reset', 'Auth\ForgotPasswordController@showLinkRequestForm')-
>name('auth.password.reset');
$this->post('password/email', 'Auth\ForgotPasswordController@sendResetLinkEmail')-
>name('auth.password.reset');
$this->get('password/reset/{token}', 'Auth\ResetPasswordController@showResetForm')-
>name('password.reset');
$this->post('password/reset', 'Auth\ResetPasswordController@reset')-
>name('auth.password.reset');

Route::group(['middleware' => ['auth'], 'prefix' => 'admin', 'as' => 'admin.'], function () {
    Route::get('/home', 'HomeController@index');

    Route::resource('roles', 'Admin\RolesController');
    Route::post('roles_mass_destroy', ['uses' => 'Admin\RolesController@massDestroy', 'as' =>
'roles.mass_destroy']);

    Route::resource('users', 'Admin\UsersController');
    Route::post('users_mass_destroy', ['uses' => 'Admin\UsersController@massDestroy', 'as' =>
'users.mass_destroy']);

    Route::resource('pms_statuses', 'Admin\PMSStatusesController');
    Route::post('pms_statuses_mass_destroy', ['uses' =>
'Admin\PMSStatusesController@massDestroy', 'as' => 'pms_statuses.mass_destroy']);

    Route::resource('pms_customer', 'Admin\PMSCustomerController');
    Route::post('pms_customers_mass_destroy', ['uses' =>
'Admin\PMSCustomerController@massDestroy', 'as' => 'pms_customers.mass_destroy']);

    Route::resource('pms_notes', 'Admin\PMSNoteController');
    Route::post('pms_notes_mass_destroy', ['uses' => 'Admin\PMSNoteController@massDestroy',
'as' => 'pms_notes.mass_destroy']);

    Route::resource('pms_documents', 'Admin\PMSDocumentController');
    Route::post('pms_documents_mass_destroy', ['uses' =>
'Admin\PMSDocumentController@massDestroy', 'as' => 'pms_documents.mass_destroy']);

    Route::resource('pms_forum', 'Admin\PMSForumController');
    Route::post('pms_forums_mass_destroy', ['uses' => 'Admin\PMSForumController@massDestroy',
'as' => 'pms_forums.mass_destroy']);
});
```