10-2019

Personal Blog Based on Spring Boot

Ji Zhao
St Cloud State University, jzhao@stcloudstate.edu

Personal Blog Based on Spring Boot

by

Ji Zhao

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

October, 2019

Starred Paper Committee:
Jie Hu Meichsner, Chairperson
Ramnath Sarnath
Andrew A. Anda

Abstract

Web application development has played an important role in software Engineering. Model-View-Controller (MVC) pattern lays a foundation for developing web applications. The MVC architecture separates an application into different business logic (data presentation, data management and request handling). The Spring Framework is an application framework used by Java application and there are extensions for building web applications [1]. Spring Boot is a framework tool designed to simplify the initialization of Spring, which makes it easy to create stand-alone, production-grade Spring based applications [2]. Hibernate is an Object-Relational Mapping tool which maps the database tables to the Object classes. The goal of this project is to develop a personal blog that can be used to write and post articles, pictures and codes by the administrator. The blog also allows general users to read and comment on the blogs. The architecture of the proposed system will be based on Spring Boot and Hibernate.

Table of Contents

List of Tables

List of Figures

Figure Page

Figure Page

Chapter 1: Introduction

A web-based application is any program that is accessed over a network connection using HTTP, rather than existing within a device's memory [1]. Initially, web-based applications were developed by Java Server Pages which separated the application into front-end (for the view part, mostly HTML code) and back-end (for the application logic part, mostly Java). This segregation not only makes developing and environmental configurations complicated, but also caused maintenance issues.

Framework gained popularity as it implements the Model-View-Controller pattern. It offers better platform for better extensibility, scalability and code organization. Spring Boot is a framework that makes it easy to create Spring based applications. Spring Boot allows developers to select and install dependencies with minimum effort.

Database is commonly used to store information and data in all kinds of applications nowadays. Spring Boot support Object Relational Mapping (ORM), which maps Java classes to the corresponding database information. Hibernate is one of the ORM tool that supported by Spring Boot to avoid queries written in database.

In this project, a Personal Blog application has been implemented. The application allows the administrator to login and post articles, pictures and codes to the blog. And general users can read and comment on the blogs.

The contents of this paper are organized as follows: Chapter 2 gives an overview of Spring Boot and Hibernate frameworks. Chapter 3 presents the system design and functionalities of the Personal Blog with the application screen shots. Chapter 4 concludes the project and provides possible future enhancements. Finally, the project references are listed.

Chapter 2: Technology Overview

This chapter presents information about the Spring, Spring Boot and Hibernate frameworks – their brief introduction, features and architectures. The first section is about Spring and Spring Boot and the second section explains the Hibernate framework.

2.1 Spring and Spring Boot

Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications.

2.1.1 Spring Architecture

The architecture of Spring is shown in Figure 1. This figure is adapted from 'Introduction to Spring Framework' [3].

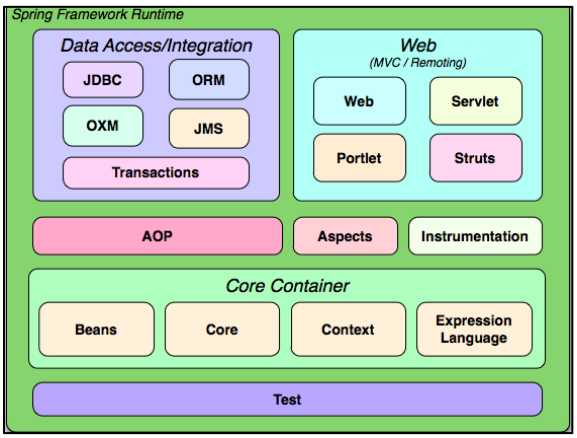

Figure 1: Overview of the Spring Framework

Spring framework consists of the following components [3]:

- **Core Container:** The Core container consists of the Core, Beans, Context and Expression Language modules**,** The Core and Beans modules provide the fundamental parts of the framework. The Context module builds on the solid base

provided by the Core and Beans modules. The Expression Language module provides a powerful expression language for querying and manipulating an object graph at runtime.

- **Data Access/Integration:** The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules. The JDBC module provides a JDBC-abstraction layer that removes the need to do tedious JDBC coding and parsing of database-vendor specific error codes. The ORM module provides integration layers for popular object-relational mapping APIs.
- **Web:** Spring's Web module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the Inverse of Control (IoC) container using servlet listeners and a web-oriented application context. It also contains the web-related parts of Spring's remoting support.
- **AOP and Instrumentation:** Spring's AOP module provides an AOP Alliance-compliant aspect-oriented programming implementation allowing you to define, for example, method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.
- **Test:** The Test module supports the testing of Spring components with JUnit or TestNG. reverse order to perform any post-processing logic. Once the result is generated, it is responsible to match the JSP or FreeMarker template in which the result should be rendered.

2.1.2 Spring Boot

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that developers can just run. Minimum configurations are required without the need for an entire Spring configuration setup. Spring Boot is designed to avoid complex XML configuration in Spring and to develop a production ready Spring Applications in an easier way [4].

2.1.3 Advantages of Spring Boot

Spring Boot has many advantages, some of which are mentioned below [5]:

- **Resolving Dependency Conflict**: Spring Boot helps in resolving dependency conflict. It identifies required dependencies and import them for you.
- **Compatible Version**: Spring Boot as information of **compatible version** for all dependencies. It minimizes the runtime **class-loader** issues.
- **Avoiding boilerplate code**: Spring Boot's "opinionated defaults configuration" approach helps you in configuring most important pieces behind the scene. Override them only when you need. Otherwise everything just works, perfectly. It helps in avoiding boilerplate code, annotations and XML configurations.
- **User Friendly**: provides embedded HTTP server Tomcat so that you can develop and test quickly.
- **Integration with IDE**: It has excellent integration with IDEs like eclipse and intelliJ idea.

2.2 Hibernate

In most application is stored as objects and represented with classes. Object-Relational Mapping (ORM) framework is used to map objects from the objects to the relational database. The ORM tool allows developers to perform database operations without writing raw SQL queries. ORM frameworks work by transforming one data representation to another from as illustrated in Figure 2. This figure is adapted from ‘Research on data persistence layer based on hibernate framework’ [6].

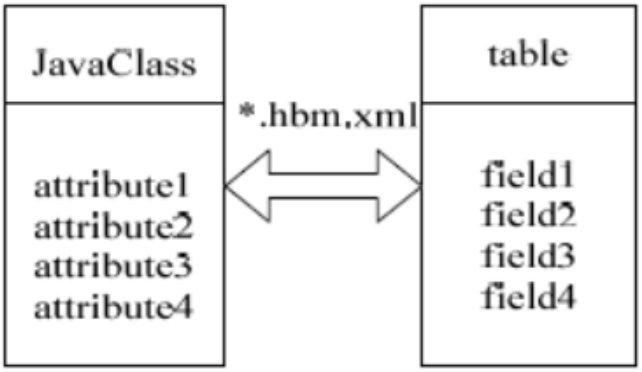

Figure 2: ORM Mapping Concept

Hibernate is one such ORM tool which integrates well with Spring Boot applications. It solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions. Hibernate resides between the Java objects and database system to perform all the work needed to persist the objects in the database.

2.2.1 Hibernate Architecture

3 shows the architecture of Hibernate framework. This figure is adapted from ‘Hibernate Architecture’ [7].

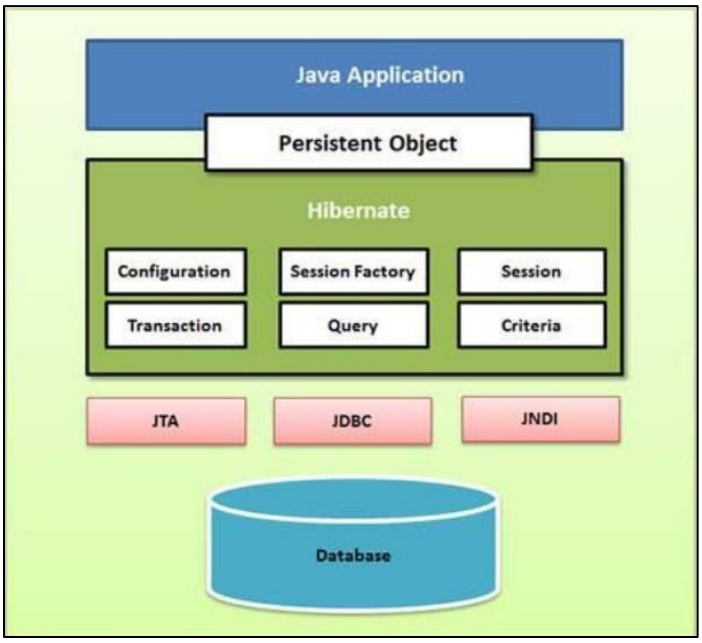

Figure 3: Hibernate Architecture

Hibernate Application Architecture involves in the following important class objects [7]:

- **Configuration:** The Configuration object is the first Hibernate object you create in any Hibernate application. It is usually created only once during application initialization. It represents a configuration or properties file required by the Hibernate.
- **Session Factory:** A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object. The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed.
- **Transaction Object:** A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality. Transactions in Hibernate are

handled by an underlying transaction manager and transaction (from JDBC or JTA). This is an optional object and Hibernate applications may choose not to use this interface, instead managing transactions in their own application code.

- **Query Object:** Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects. A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query.
- **Criteria Object:** Criteria objects are used to create and execute object-oriented criteria queries to retrieve objects.

2.2.2 Advantages of Hibernate

The advantages of Hibernate [7] are listed below:

- **Database Independent:** Hibernate is independent of the database engine at the backend. List of Hibernate Dialect are provided for connecting whatever database we prefer.
- **JPA Provider:** Java Persistence API (JPA) is a specification. A lot of implementations are available for JPA; Like EclipseLink, OpenJPA and much more. Hibernate is a standard ORM solution and it has a JPA capability. Hence, using of hibernate would help you leverage the all capabilities of ORM and a JPA in a JPA-specific project.

Chapter 3: The Personal Blog System

This chapter is a detailed description of the Personal Blog System. The first section of this chapter explains the different functionalities of the system. The second section lists the system design–Entity-Relation diagrams, Use Case diagrams, Flow Charts, Sequence Diagrams and application architecture. The third section presents the application screenshots.

3.1 System Functionality

The Blog System can be used by blog administrators and general visitors. The blog offers an interactive platform where administrators can write and post articles, pictures and codes to the blog, give articles different tags and classify the articles into various types.

The administrator can log in with username and password. The administrator can manage the current blogs, manage the current archives and manage the current tags. General users can view the home page content, and blogs under different archives and tags. The detailed functionality diagram is shown in Figure 4.

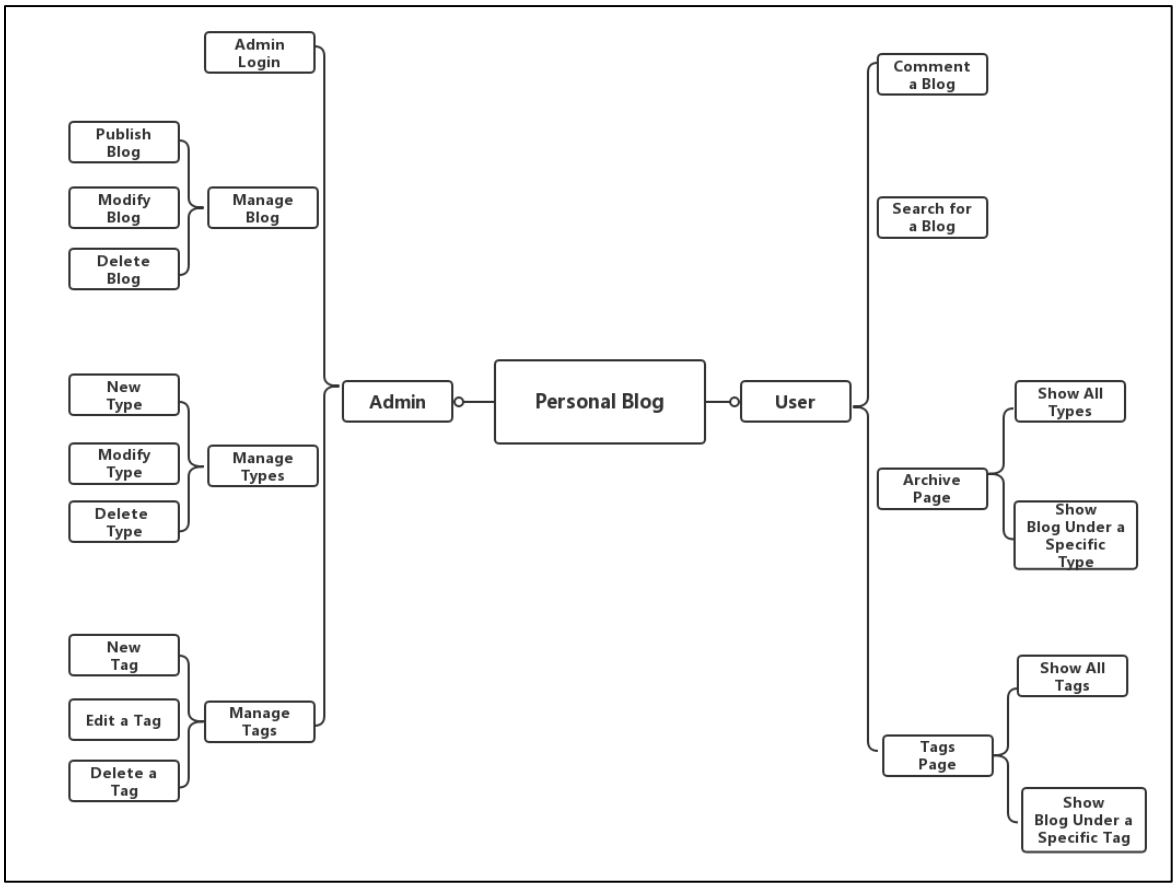

Figure 4*:* Use Case Diagram for Personal Blog

The Administrator subsystem provides the following functionalities:

Admin Login: An administrator can login with username and password to manage the content.

Manage Blog: An administrator can publish a new blog, modify the existing blogs, delete a blog and search for a blog.

Manage Types: An administrator can create a new archive, modify the existing archives, delete archives and search for an archive.

Manage Tags: An administrator can create a new tag, modify the existing tagged blogs, delete tags and search for a tag.

The User subsystem provides the following functionalities:

Comment Blog: A general user can comment a publish blog.

View Archive: A general user can view the archive of blogs

Search a Blog: A general user can search for a blog using keywords.

The different Use Cases of the Blog System are described below:

Login **Use case:**

Use Case Login is shown in Figure 5.

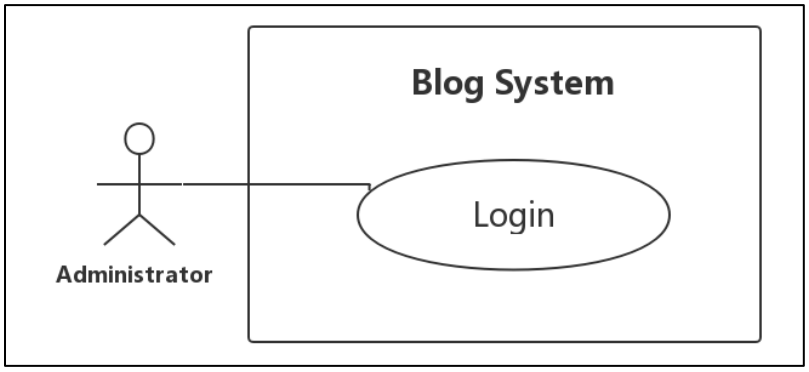

Figure 5. Login use Case for the Blog System

Table 1: *Login* use Case Description for the Blog System

Brief Description
The Login use case enables the Blog System Administrator to login into the system.
Step-by-Step Description 1. Enter the Username and Password at the login screen. 2. Validate the Username and Password entered by the user to display the administrator screen. Wrong Username or Password will stay on the login screen with error message.

Manage Blog **Use case:**

Use Case Manage Blog is shown in Figure 6.

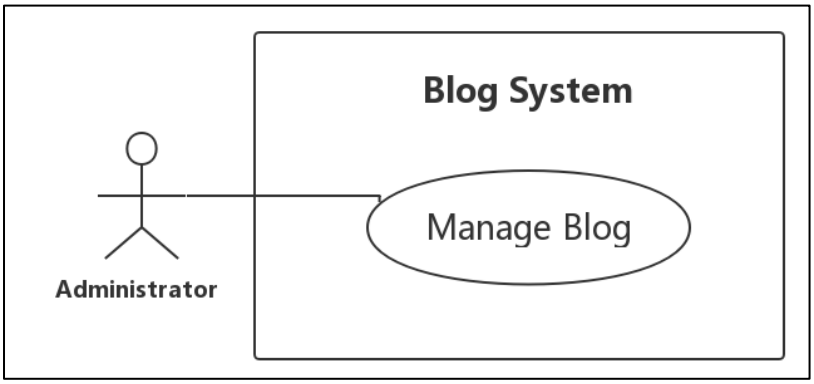

Figure 6: Manage Blog use Case for the Blog System

Table 2: Manage Blog use Case Description for the Blog System

Brief Description The Manage Blog use case enables the Blog System Administrator to manage the blog content.
Step-by-Step Description 1. Allow the following modifications to the current blog system: • Publish a new blog • Modify a current blog • Delete a current blog • Search for a blog

Manage Archive **Use case:**

Use Case Manage Archive is shown in Figure 7.

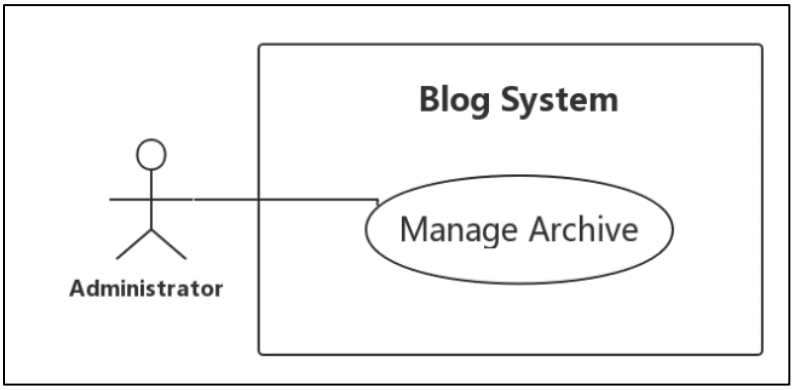

Figure 7: *Manage Type* use Case for the Blog System

Table 3: Manage Archive use Case Description for the Blog System

Brief Description The Manage Types use case enables the Blog System Administrator to manage the types for the blog.
Step-by-Step Description 1. Allow the following modifications to the current blog system: • Create a new type • Change a current blog article to a different type • Delete a current type

Manage Tags **Use case:**

Use Case Manage Tags is shown in Figure 8.

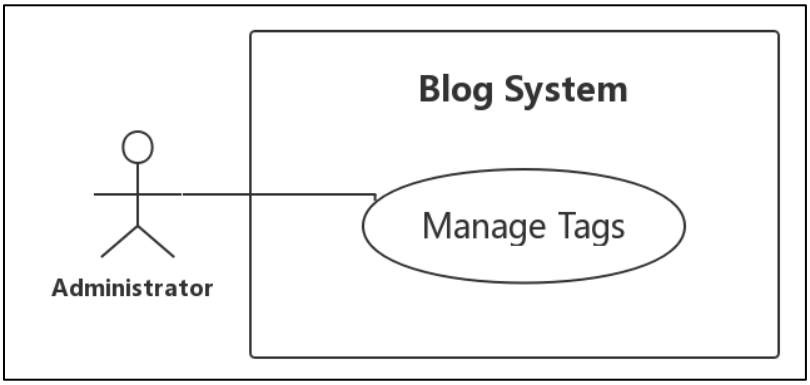

Figure 8 : Manage Tags use Case for the Blog System

Table 4: Manage Tags use Case Description for the Blog System

Brief Description The Manage Tags use case enables the Blog System Administrator to manage the tags for the blog.
Step-by-Step Description 1. Allow the following modifications to the current blog system: • Create a new tag • Change a current blog article to a different tag • Delete a current tag

General User Use **case:**

Use Case General User Comment is shown in Figure 9.

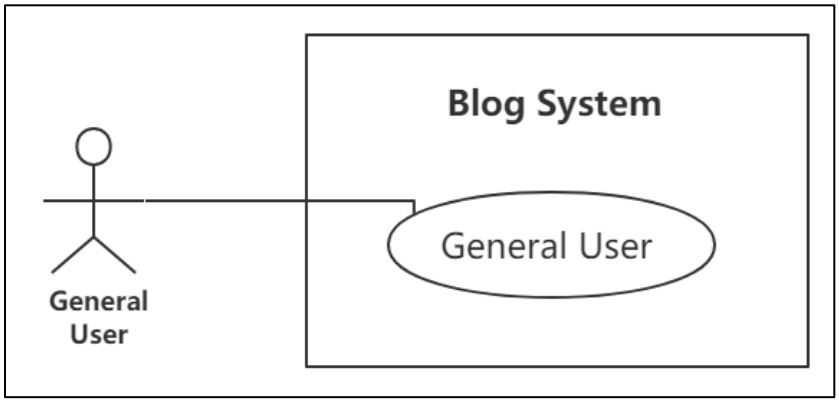

Figure 9: General User use Case for the Blog System

Table 5: General User use Case Description for the Blog System

Brief Description The General User use case enables the Blog System General User to view the blog content and search for a specific blog.
Step-by-Step Description Allow the following operations to the current blog system: • View blog content by Tags, Archive or Types • Comment a blog • Search for a blog using its keywords

3.2 System Design

The proposed application will have as shown in Figure 10.

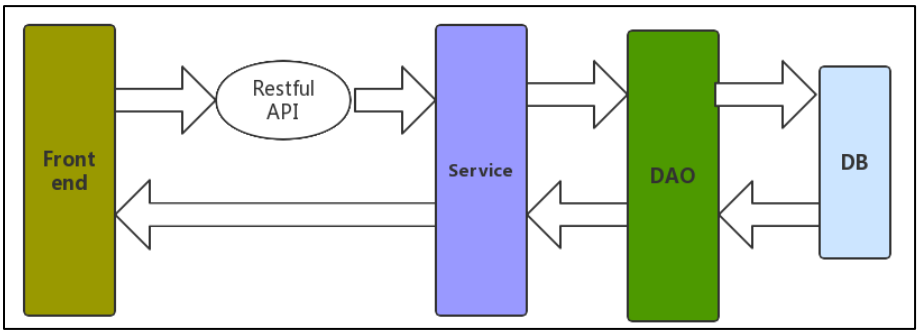

Figure 10: System Architecture

Design of the frontend. This layer will be the user interface. It is responsible for displaying information to user.

RESTful API. REST is the underlying architectural principle of the web. The amazing thing about the web is the fact that clients (browsers) and servers can interact in complex ways without the client knowing anything beforehand about the server and the resources it hosts. The key constraint is that the server and client must both agree on the *media* used, which in the case of the web is HTML. An API that adheres to the principles of *REST* does not require the client to know anything about the structure of the API. Rather, the server needs to provide whatever information the client needs to interact with the service [4].

Design of the service. This would be the service layer for the application. The application logic will be presented in this layer and it will be responsible for answering the call from the RESTful API and interacting with the DAO layer to receive data. This layer would be implemented by the Spring Boot framework.

Design of the DAO. This layer is responsible for adding and inserting entity objects into database, updating entity objects in database, and selecting entity objects from database.

3.3 System Modulo Design and Use Case Realization

The classes in the Blog System have been classified into three categories: Entity Classes, Boundary Classes and Controller Classes.

- An Entity class models the information that is long lived
- A Boundary class models the interaction between and its actors
- A Control class models complex computations and algorithms

The Entity–Relation (ER) diagram for entity classes is shown in Figure 11.

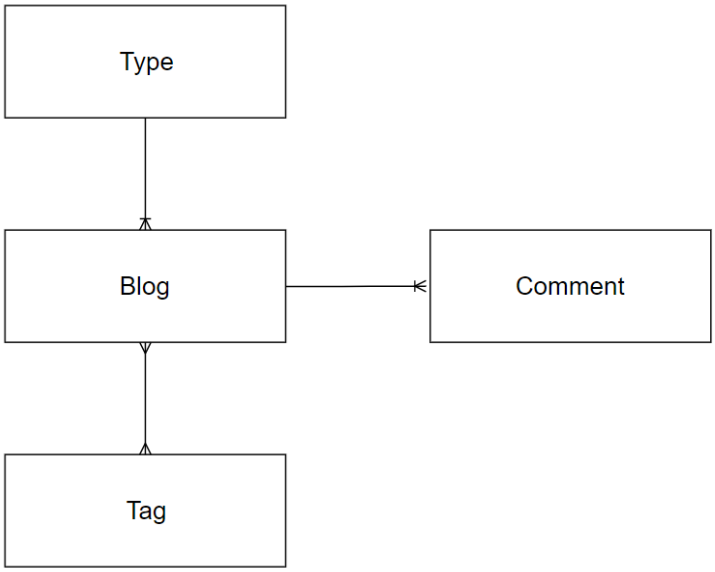

Figure 11: Entity Relation Diagram for Blog System

Administrator Login Modulo and Use Case Realization

Administrator login the Blog System through the login page by inputting the correct username and password. The flowchart Login is shown in Figure 12.

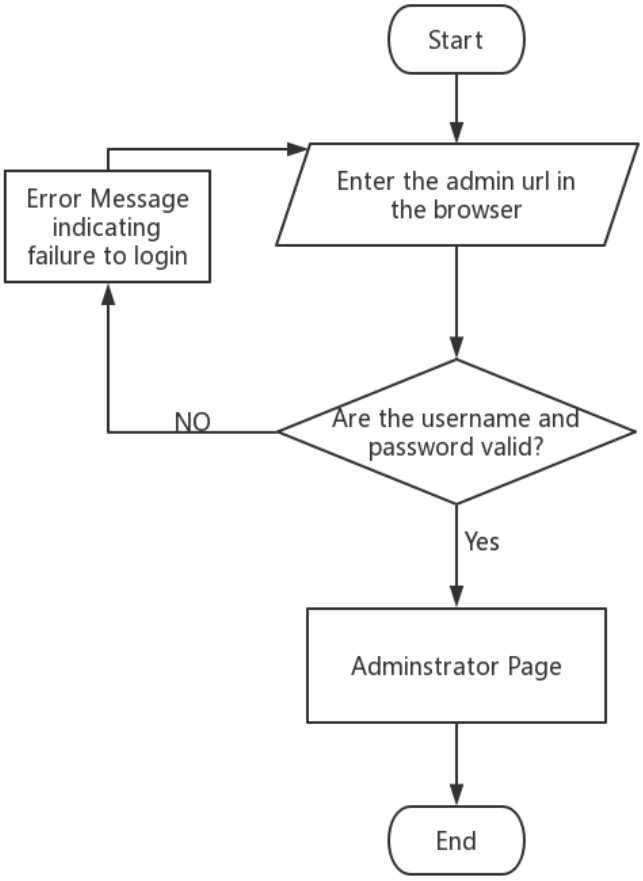

Figure 12: Login Flowchart for the Blog System

The Sequence diagram for the realization of the *Administrator Login* use case is shown in Figure 13.

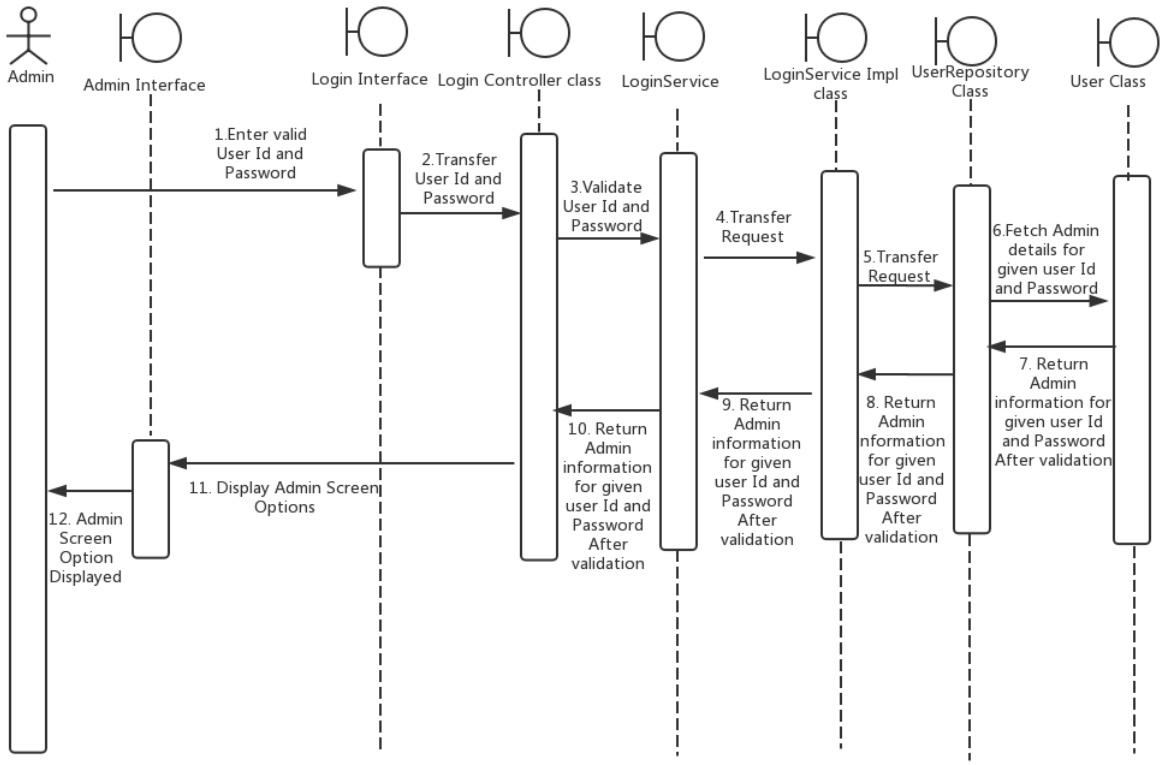

Figure 13: Sequence Diagram for ‘Administrator Login’

Manage Blog Modulo. This modulo contains a series operation towards blog content. The structure diagram is shown in Figure 14.

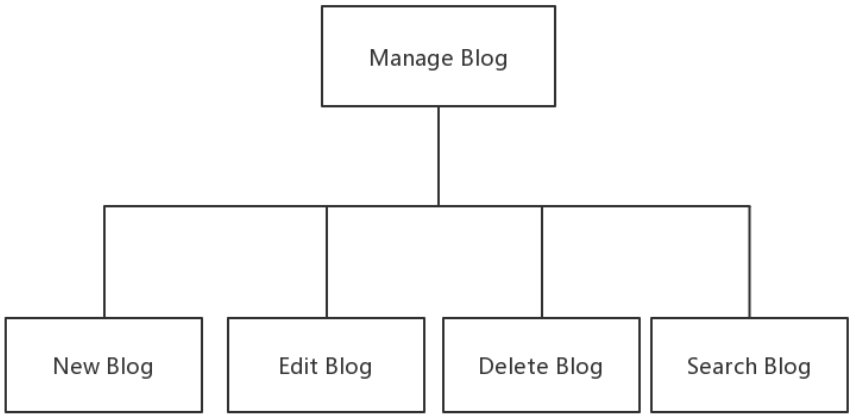

Figure 14: Manage Blog Structure Diagram for the Blog System

Publish blog. This function allows administrator to publish new articles, pictures and links after logging in to the blog. The Personal Blog System supports Markdown editor to satisfy different needs. The Personal Blog System allows administrator to tag each blog so similar types of blogs can be searched together. This function involves adding new content into the database and store the blog-tag relation into the database. The flowchart Publish Blog is shown in Figure 15.

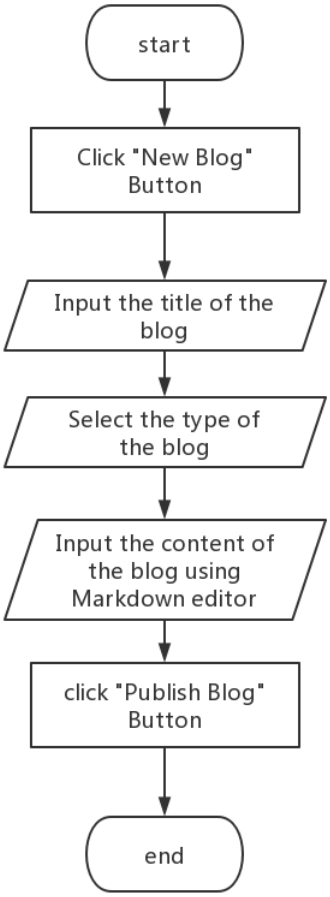

Figure 15: Publish Blog Flowchart for the Blog System

The sequence diagram for the realization of the *Search Blog* use case is shown in

Figure 16.

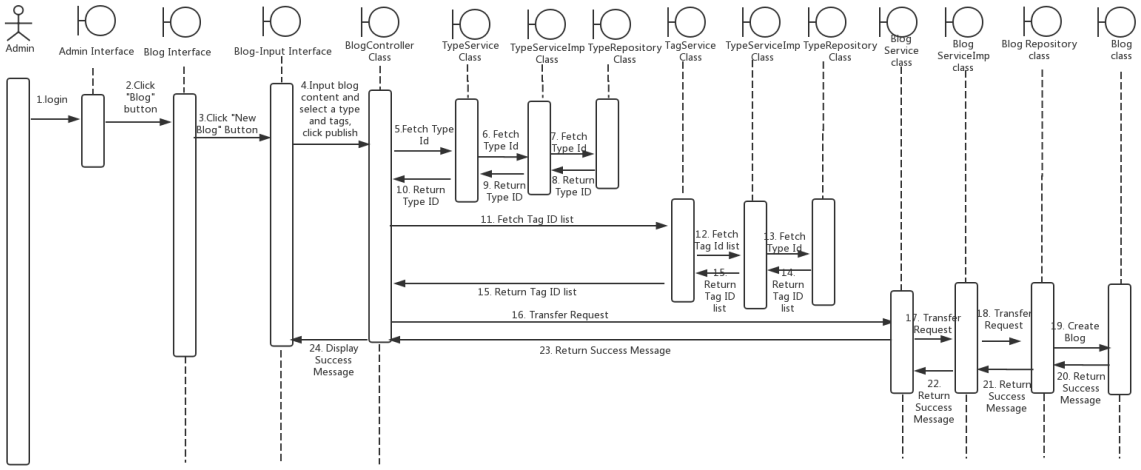

Figure 16: Publish Blog Sequence Diagram for the Blog System

Edit blog. This function allows administrator to edit the content of the existing blogs after logging in. This function involves obtaining information from the database and updating the information in the database according to the administrator's change. Also involves displaying the changed content on the screen. The flowchart Edit Blog is shown in Figure 17.

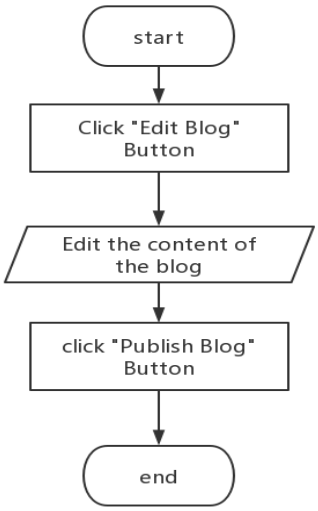

Figure 17: Edit Blog Sequence Diagram for the Blog System

The sequence diagram for the realization of the *Search Blog* use case is shown in Figure 18.

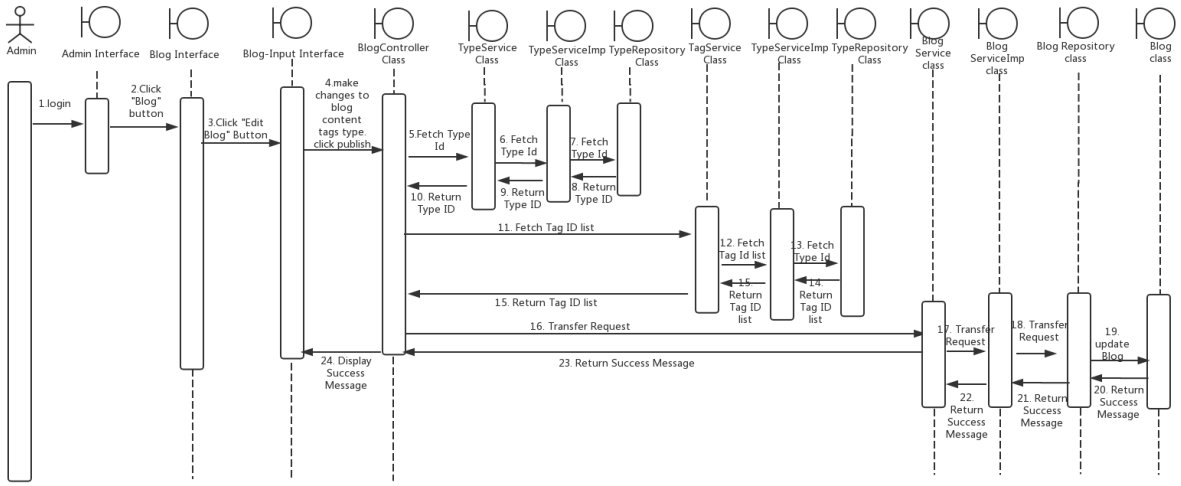

Figure 18: Edit Blog Sequence Diagram for the Blog System

Delete blog. This function allows administrator to delete the tags of the existing tags after logging in. This function involves obtaining information from the database and updating the information in the database according to the administrator's change. Also involves displaying the updated blog on the screen. The flowchart Delete a Tag is shown in Figure 19.

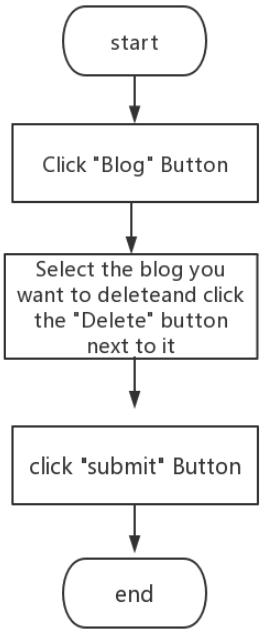

Figure 19: Delete Blog Flowchart for the Blog System

The Sequence diagram for the realization of the *Delete Blog* use case is shown in Figure 20.

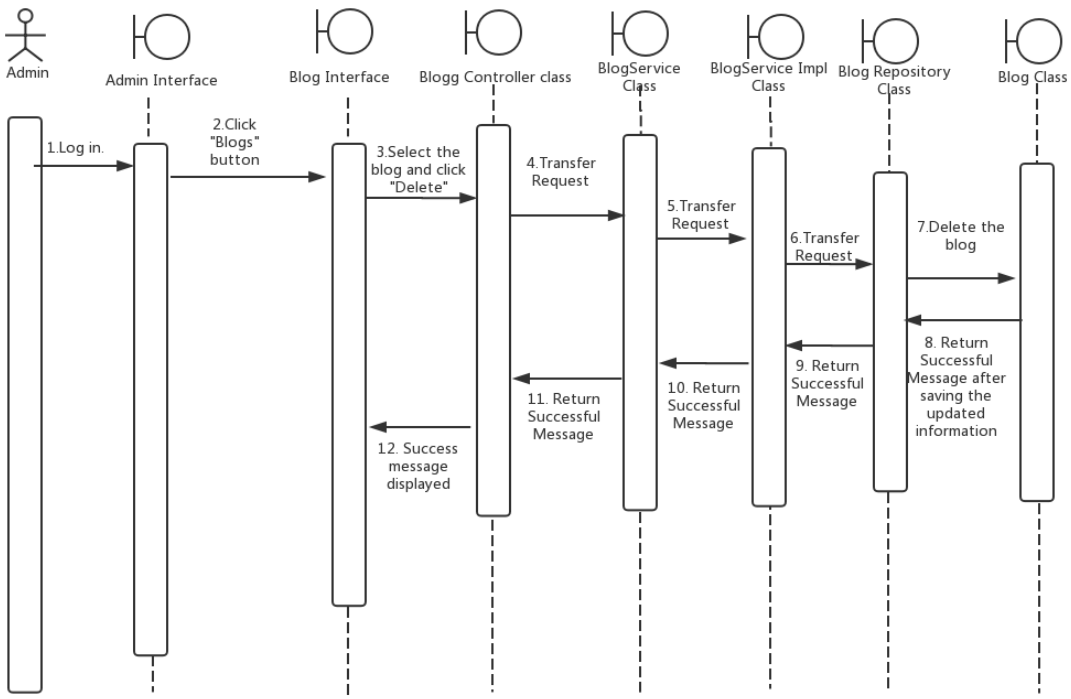

Figure 20: Delete Blog Sequence Diagram for the Blog System

Search blog. This function allows administrator to search for the blog based on the selection of blog type and input of the blog keywords. This function involves obtaining information from the database according to the key words, and display the information in the screen. The flowchart Search Blog is shown in Figure 21.

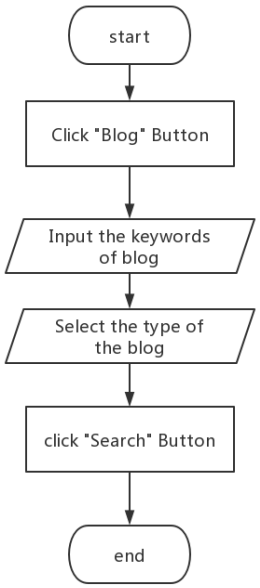

Figure 21: Search Blog Flowchart for the Blog System

The Sequence diagram for the realization of the *Search Blog* use case is shown in Figure 22.

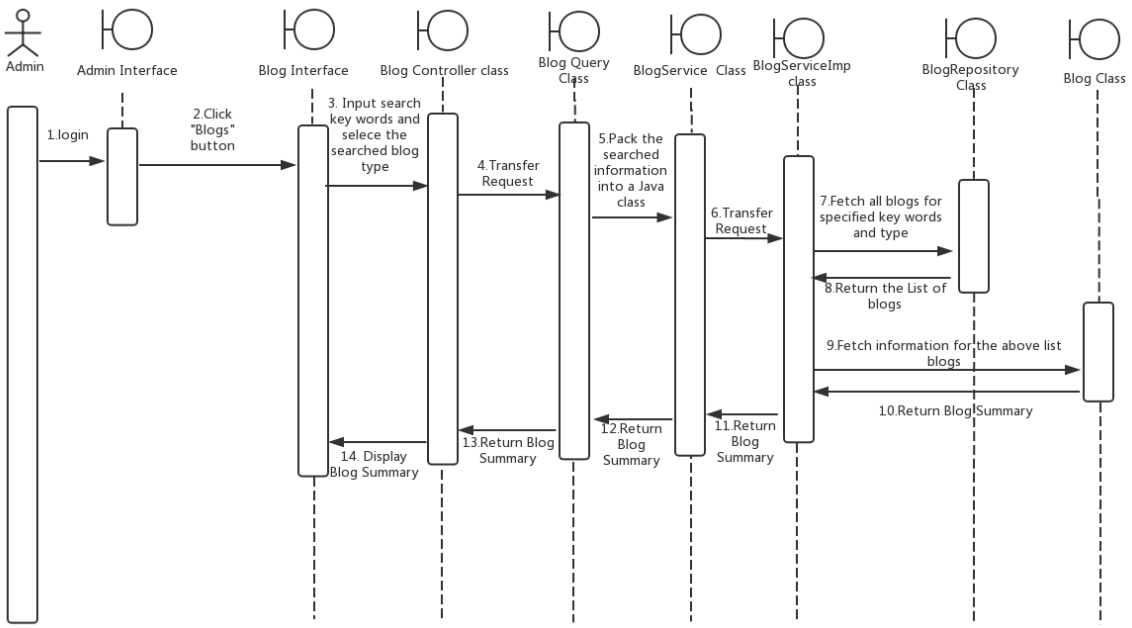

Figure 22: Search Blog Sequence Diagram for the Blog System

Manage Types Modulo and Use Case Realization

This modulo contains a series operation towards types for different blogs. The structure diagram is shown in Figure 23.

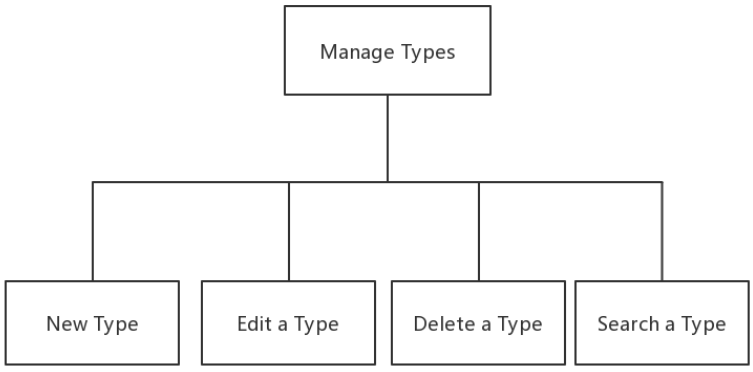

Figure 23: Manage Types Structure Diagram for the Blog System

New type. This function allows administrator to create a new blog type after logging in to the blog. The Personal Blog System allows administrator to categorize blogs into types so similar

blogs can be searched together. This function involves adding new content into the database. The flowchart New Tag is shown in Figure 24.

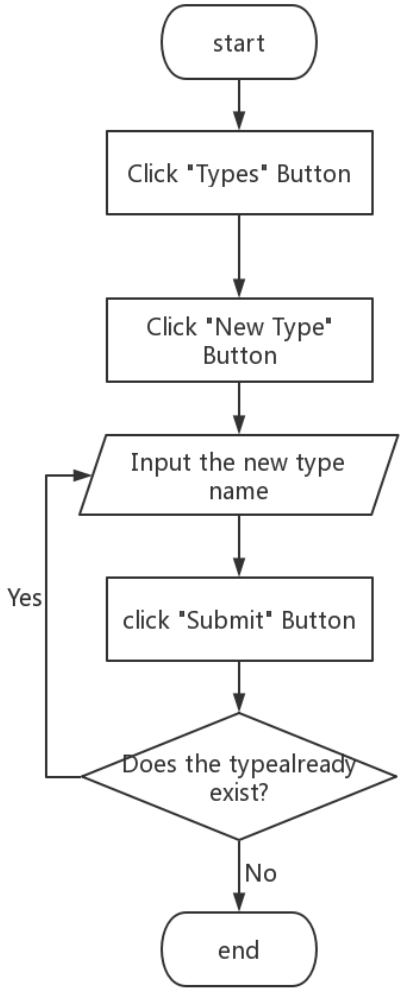

Figure 24: New Type Flowchart for the Blog System

The Sequence diagram for the realization of the *New Type* use case is shown in Figure 25.

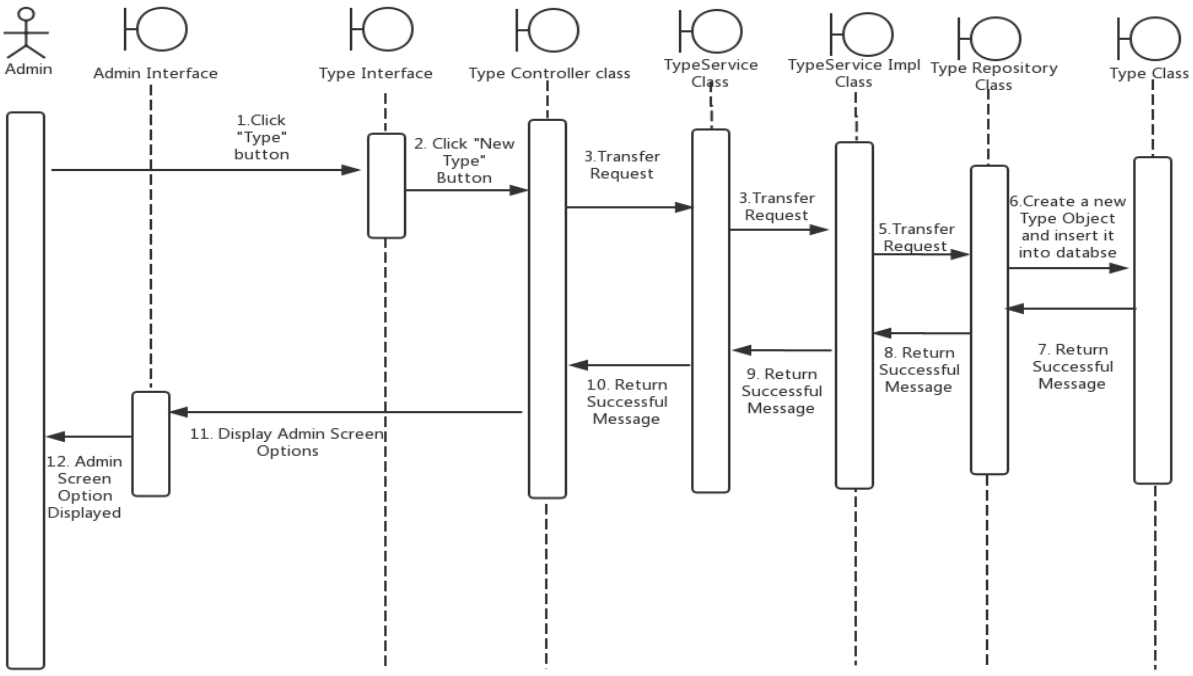

Figure 25: New Type Sequence Diagram for the Blog System

Edit types. This function allows administrator to edit a type name of the existing types after logging in. This function involves obtaining information from the database and updating the information in the database according to the administrator's change. Also involves displaying the changed content on the screen. The flowchart Edit Types is shown in Figure 26.

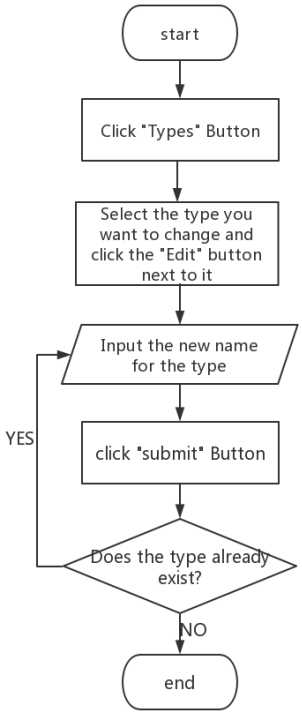

Figure 26: Edit Type Flow Chart for the Blog System

The Sequence diagram for the realization of the *Edit* Types use case is shown in Figure 27.

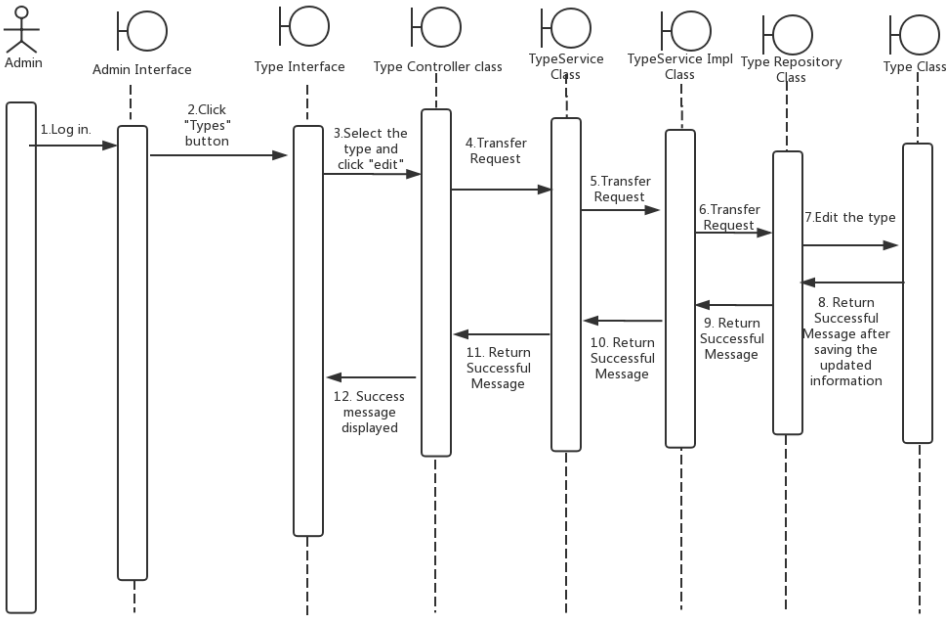

Figure 27: Edit Types Sequence Diagram for the Blog System

Delete type. This function allows administrator to delete the types of the existing tags after logging in. This function involves obtaining information from the database and updating the information in the database according to the administrator's change. Also involves displaying the updated blog on the screen. The flowchart Delete Types is shown in Figure 28.

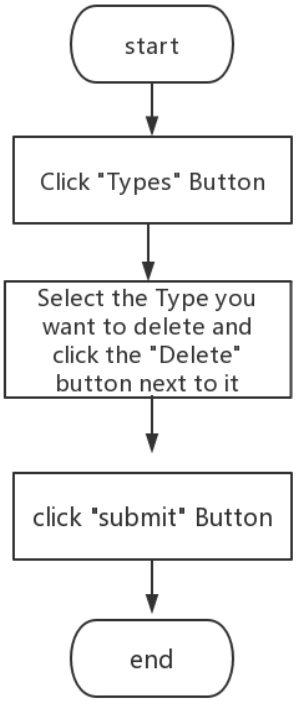

Figure 28: Delete Types Sequence Diagram for the Blog System

The Sequence diagram for the realization of the *Delete a Tag* use case is shown in Figure 29.

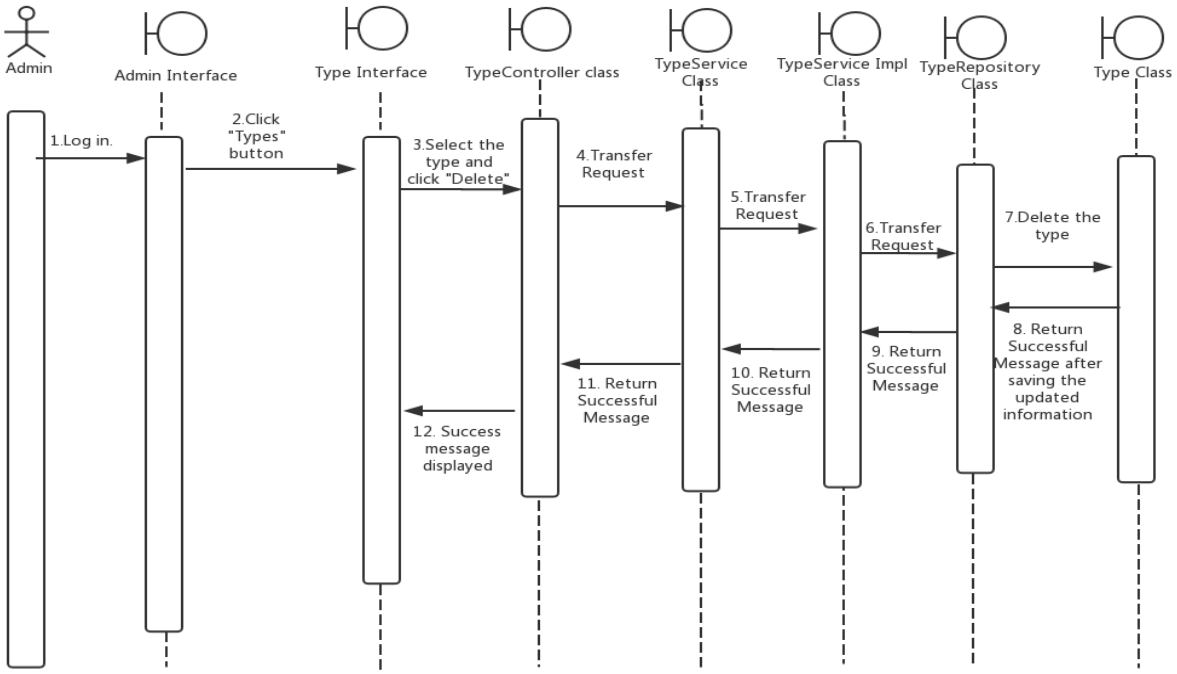

Figure 29: Delete a Type Sequence Diagram for the Blog System

Manage Tags Modulo and Use Case Realization

This modulo contains a series operation towards tags for different blogs. The structure diagram is shown in Figure 30.

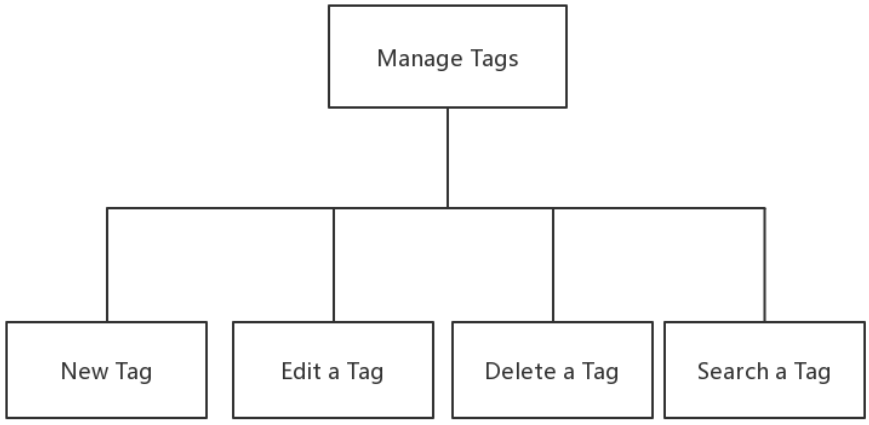

Figure 30: Manage Tags Structure Diagram for the Blog System

New tag. This function allows administrator to create a new blog tag after logging in to the blog. The Personal Blog System allows administrator to tag each blog so similar types of blogs can be searched together. This function involves adding new content into the database. The flowchart New Tag is shown in Figure 31.

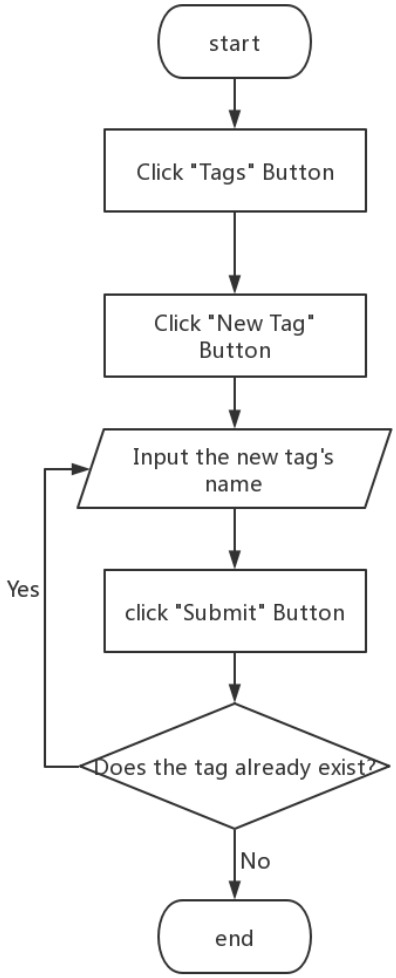

Figure 31: New Tag Flowchart for the Blog System

The Sequence diagram for the realization of the *New Tag* use case is shown in Figure 32.

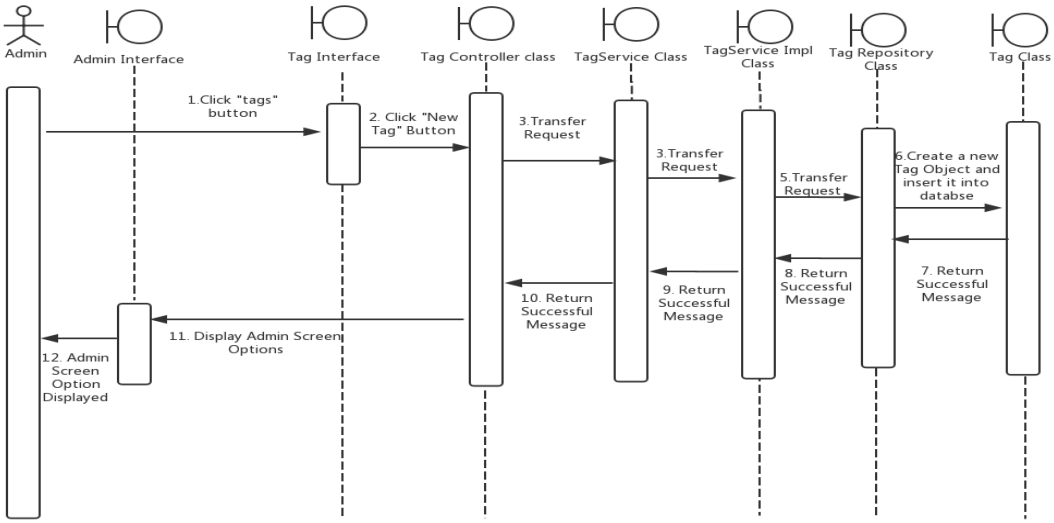

Figure 32: New Tag Sequence Diagram for the Blog System

Edit a tag. This function allows administrator to edit a tag name of the existing tags after logging in. This function involves obtaining information from the database and updating the information in the database according to the administrator's change. Also involves displaying the changed content on the screen. The flowchart Edit Tags is shown in Figure 33.

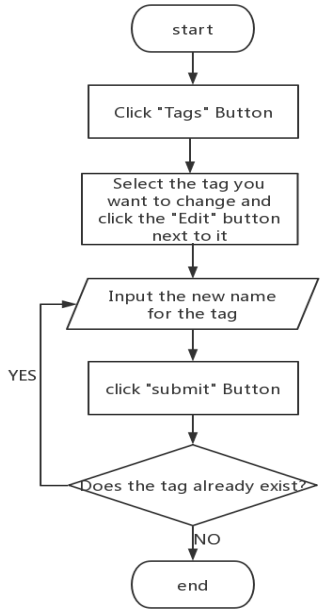

Figure 33: Edit Tags Flow Chart for the Blog System

The Sequence diagram for the realization of the *Edit a Tag* use case is shown in Figure 34.

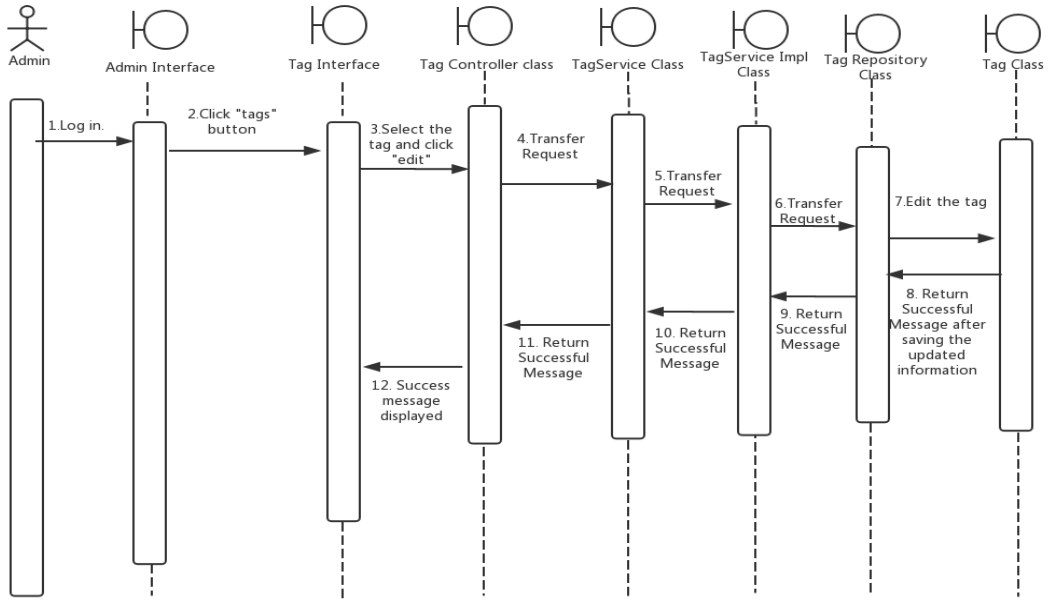

Figure 34: Edit a Tag Sequence Diagram for the Blog System

Delete a tag. This function allows administrator to delete the tags of the existing tags after logging in. This function involves obtaining information from the database and updating the information in the database according to the administrator's change. Also involves displaying the updated blog on the screen. The flowchart Delete Tag is shown in Figure 35.

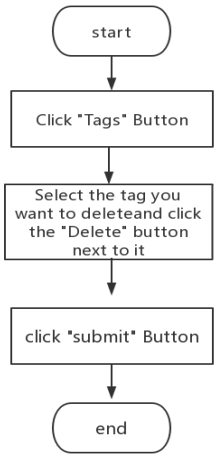

Figure 35: Delete Tag Flow Chart for the Blog System

The Sequence diagram for the realization of the *Delete a Tag* use case is shown in Figure 36.

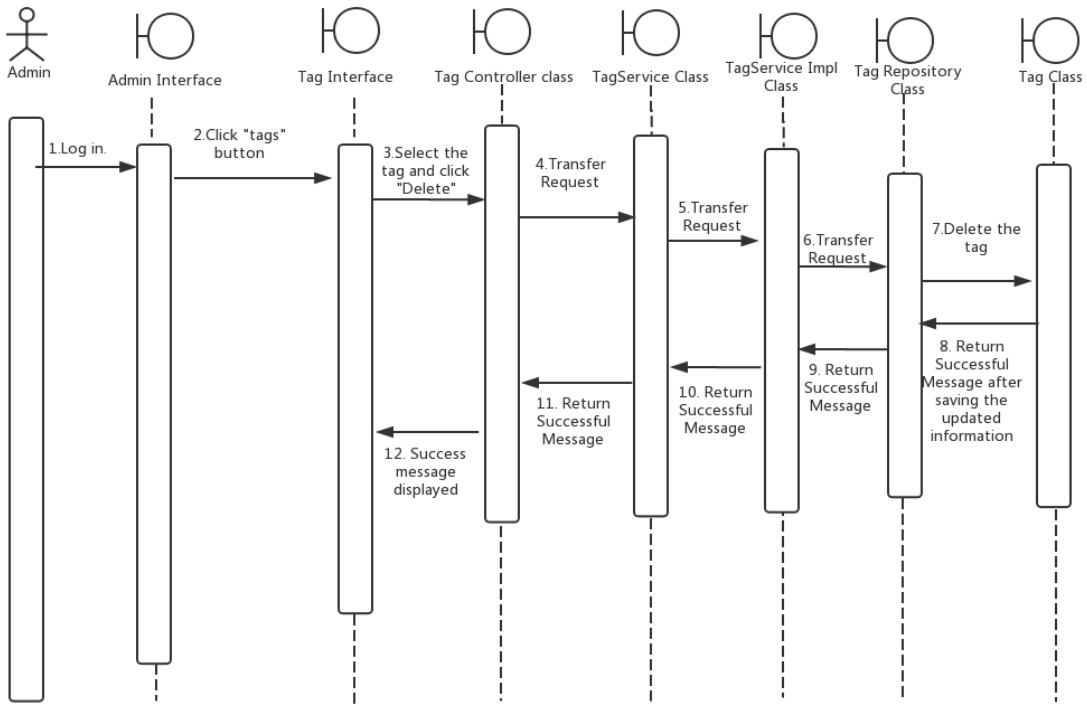

Figure 36: Delete a Tag Sequence Diagram for the Blog System

General User Modulo and Use Case Realization

Comment a blog. This function allows a general user to comment a published blog after entering his or her name and email address. This function involves obtaining the blog information from the database and updating the comment section of the blog in the database according to the user's comment. Also involves displaying the updated blog on the screen. The flowchart Comment a Blog is shown in Figure 37.

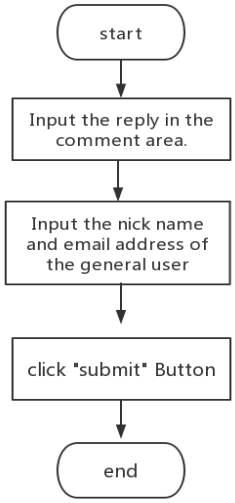

Figure 37: Comment a Blog Flow Chart for the Blog System

The Sequence diagram for the realization of the *Delete a Tag* use case is shown in Figure 38.

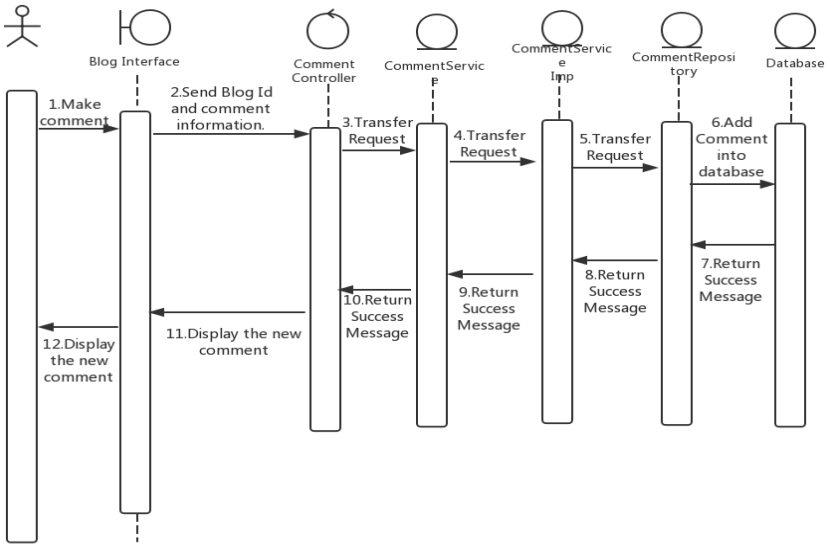

Figure 38: Comment a Blog Sequence Diagram for the Blog System

View blog archive. This function allows a general user to view the Blog archive. This function involves obtaining the blog information from the database and display the information

on the interface. The flow chart View Blog Archive is shown in Figure 39.

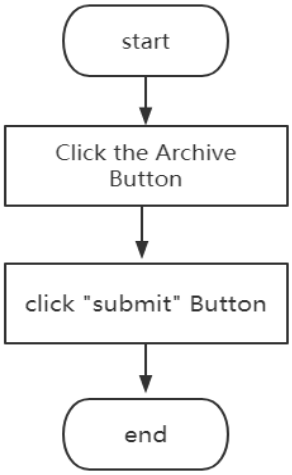

Figure 39: View Blog Archive Flow Chart for the Blog System

The Sequence diagram for the realization of the *View Blog Archive* use case is shown in Figure 40.

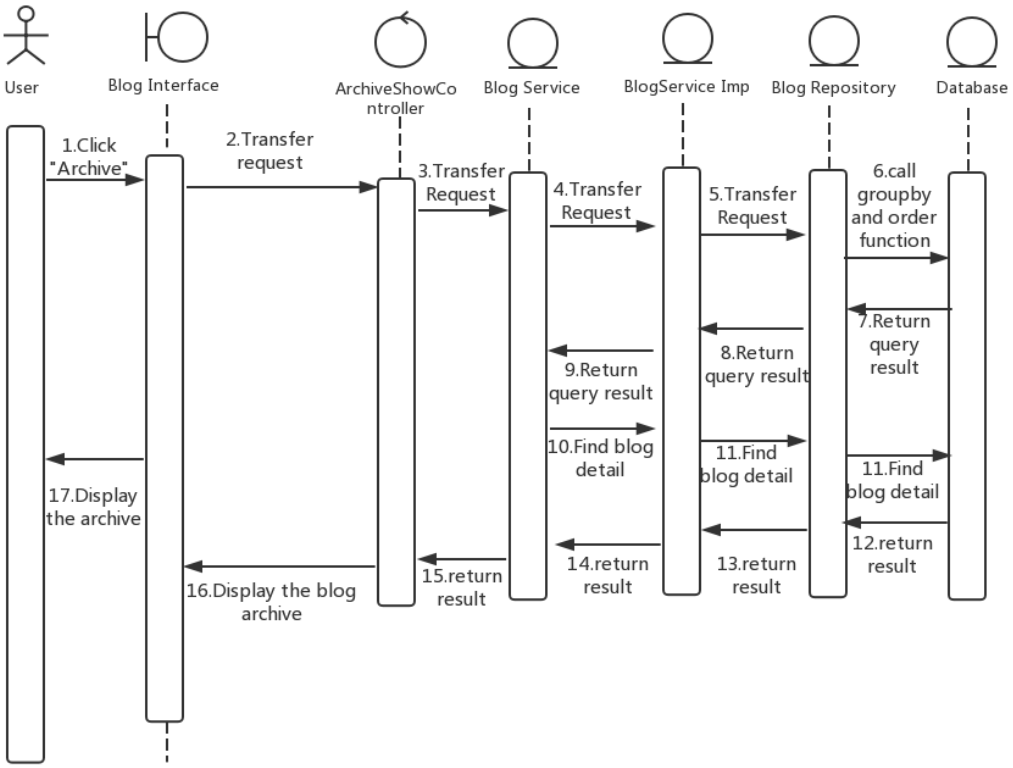

Figure 40: View Blog Archive Sequence Diagram for the Blog System

Search a blog. This function allows a general user to search a blog using keywords. This function involves obtaining the blog information from the database and display the information on the interface. The flow chart Search a Blog is shown in Figure 41.

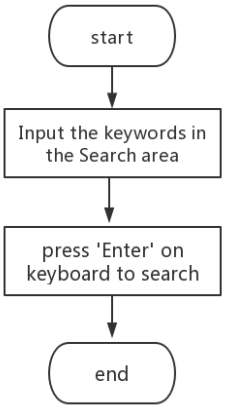

Figure 41: Search for a Blog Flow Chart for the Blog System

The Sequence diagram for the realization of the *Search for a Blog* use case is shown in Figure 42.

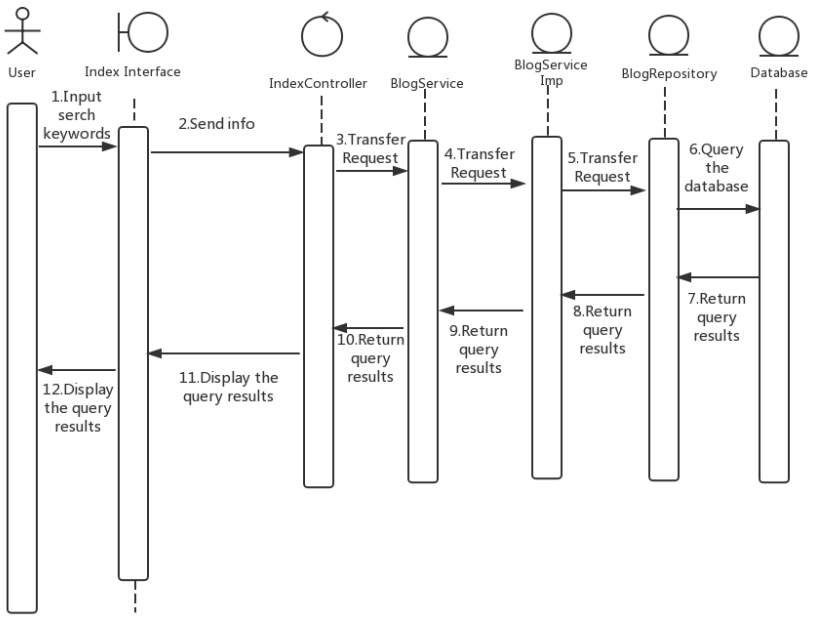

Figure 42: Search for a Blog Sequence Diagram for the Blog System

3.4 Blog System Screen Shots

The screenshots of the different functionalities of the Personal Blog System are presented below:

Login Screen

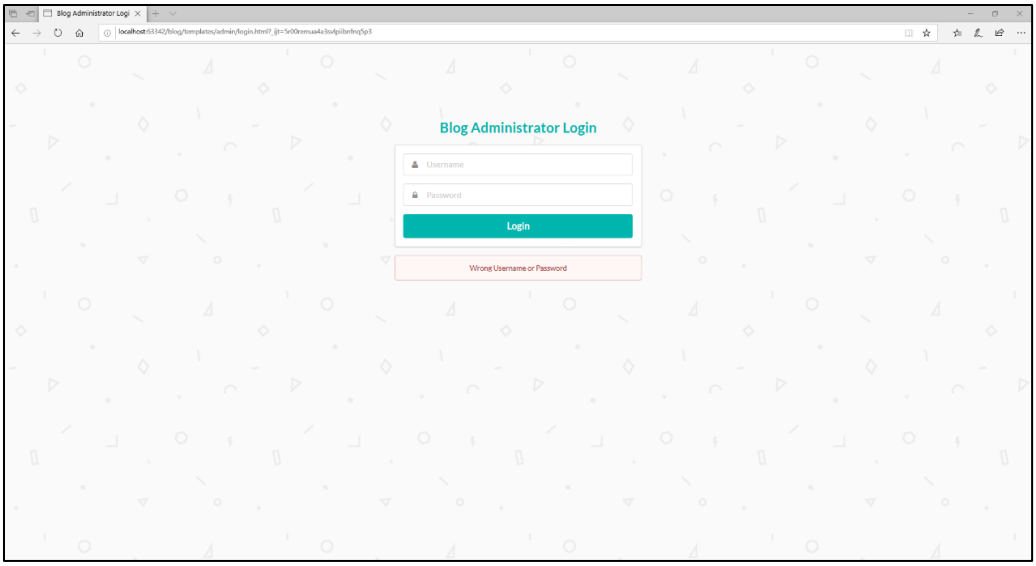

Figure 43: Login Screen of the Blog System

Admin Home

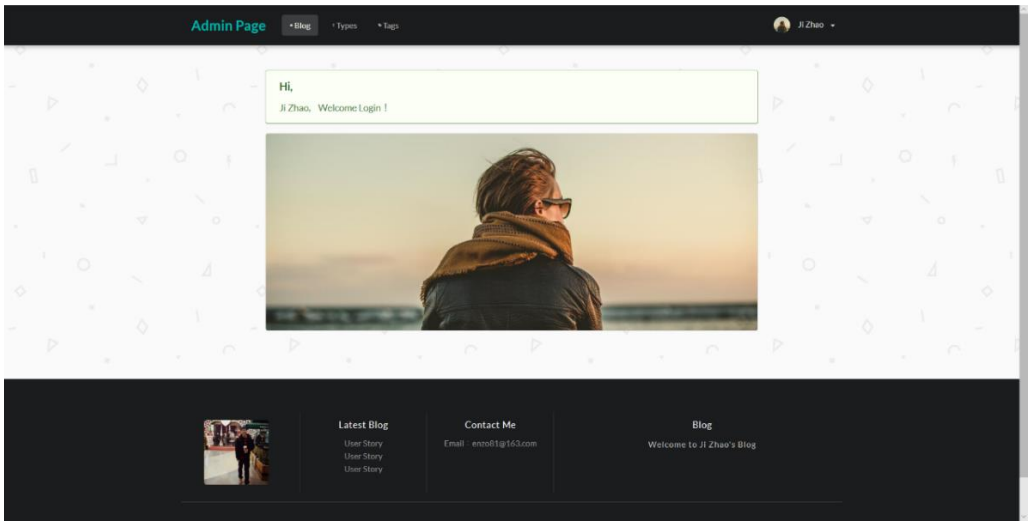

Figure 44: Admin Home Screen of the Blog System

Manage Blog

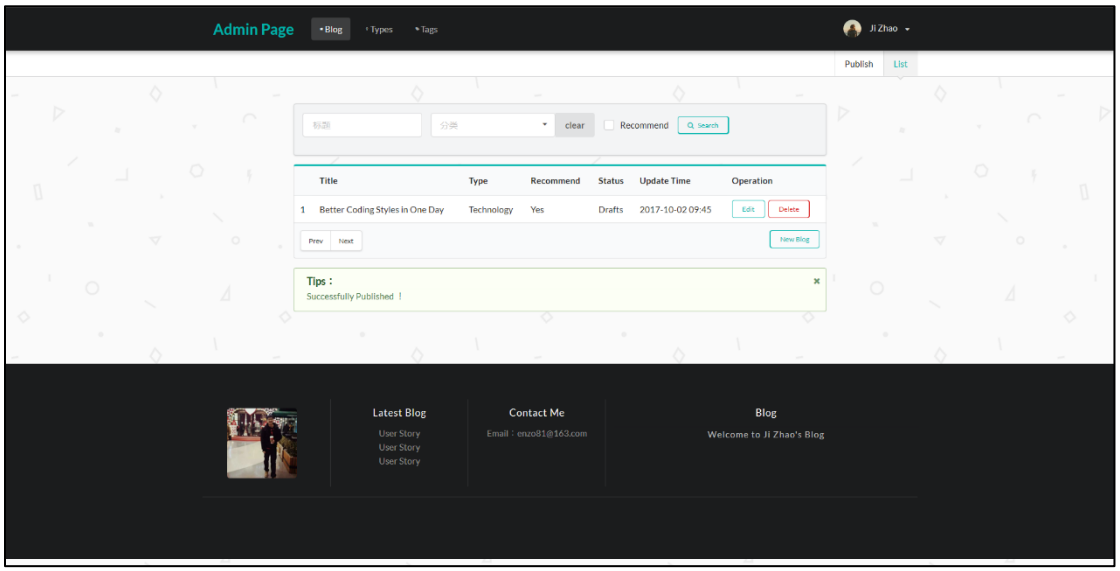

Figure 45: Manage Blog Screen of the Blog System

New Blog

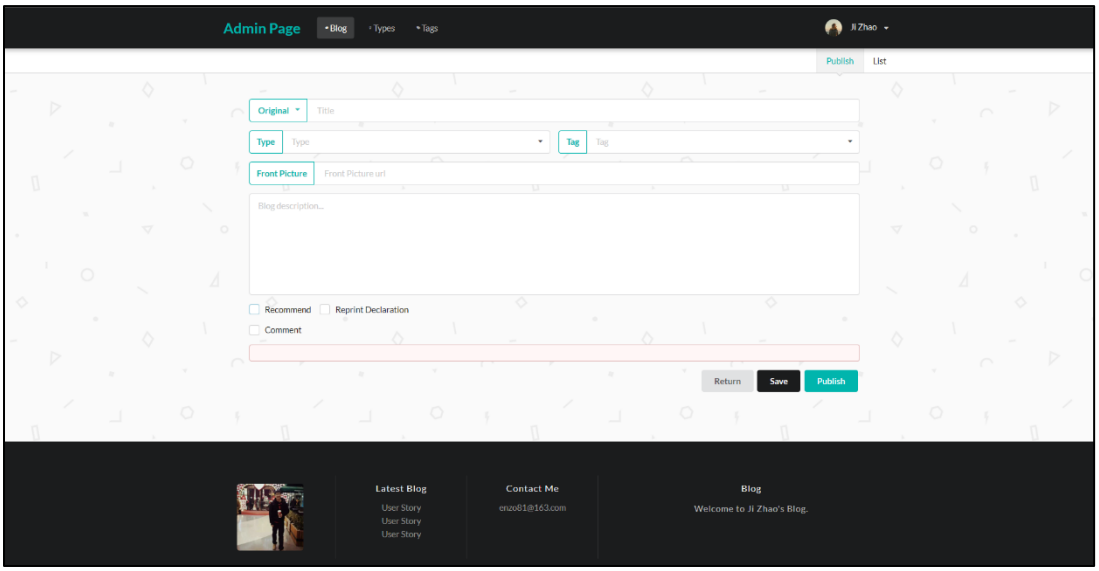

Figure 46: New Blog Screen of the Blog System

Manage Tag

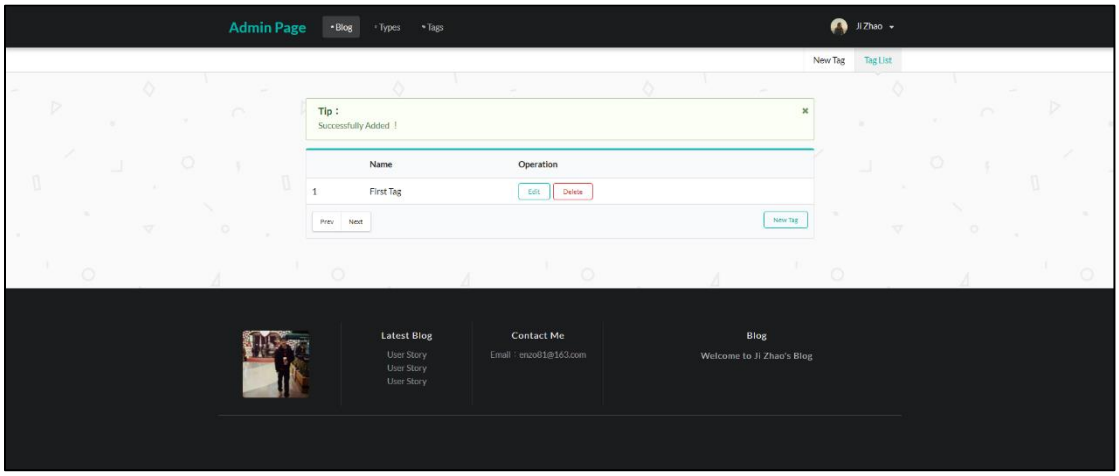

Figure 47: Manage Tag Screen of the Blog System

New Tag

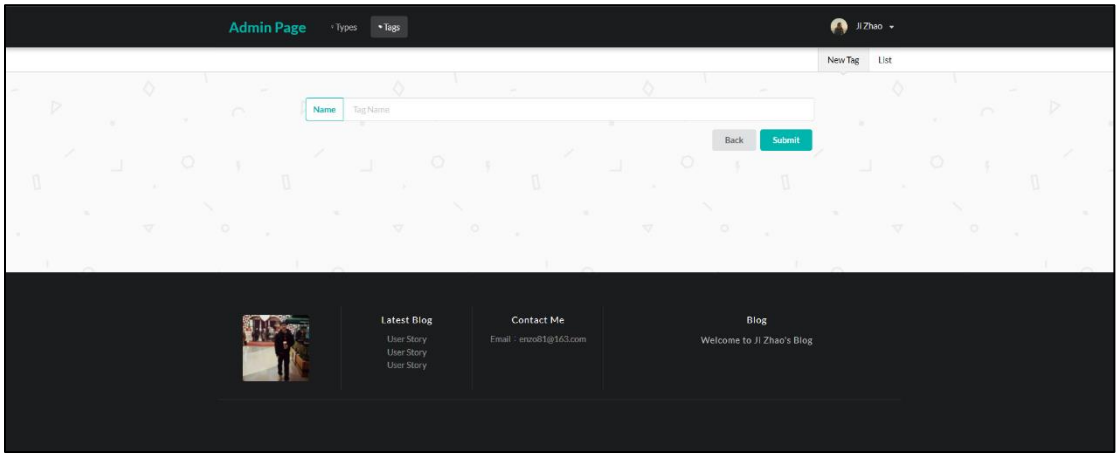

Figure 48: New Tag Screen of the Blog System

Blog Home

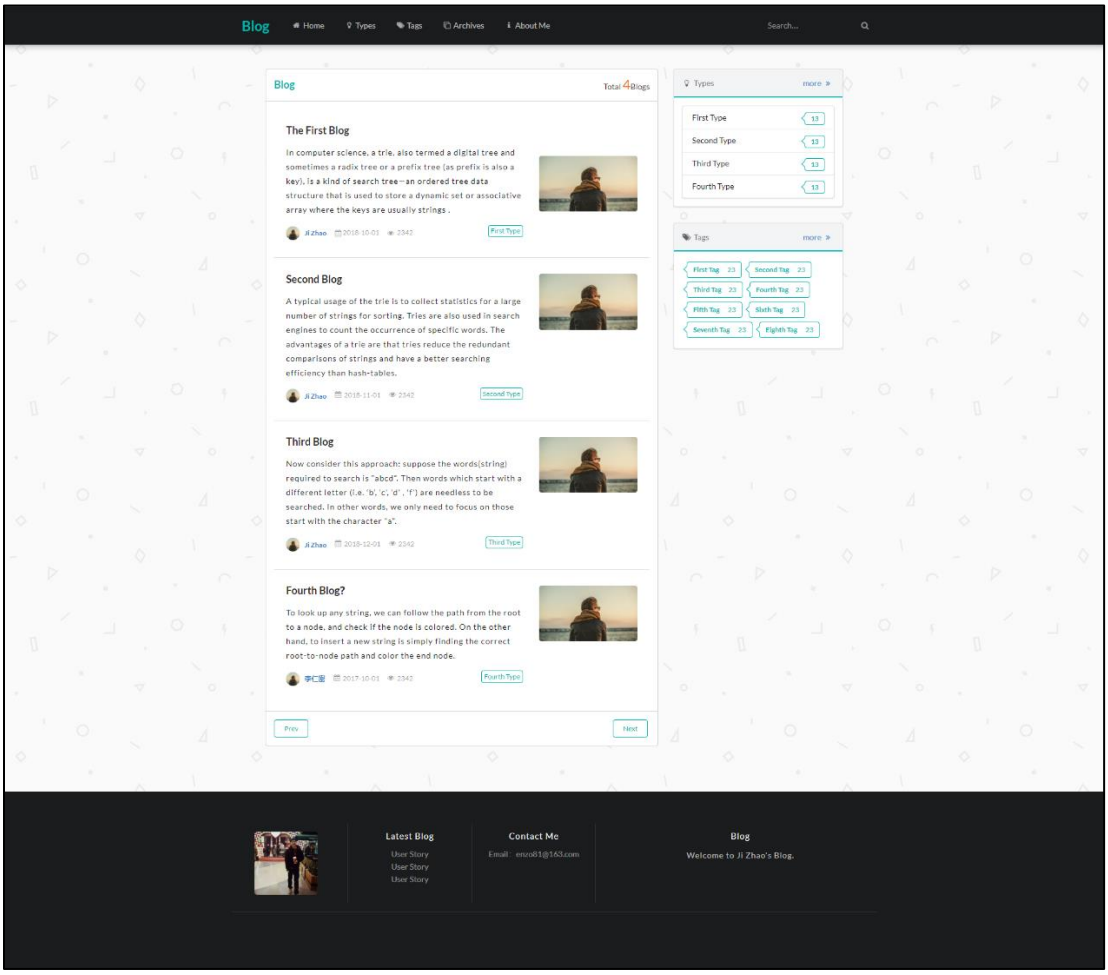

Figure 49: Blog Home Page Screen of the Blog System

Blog Details

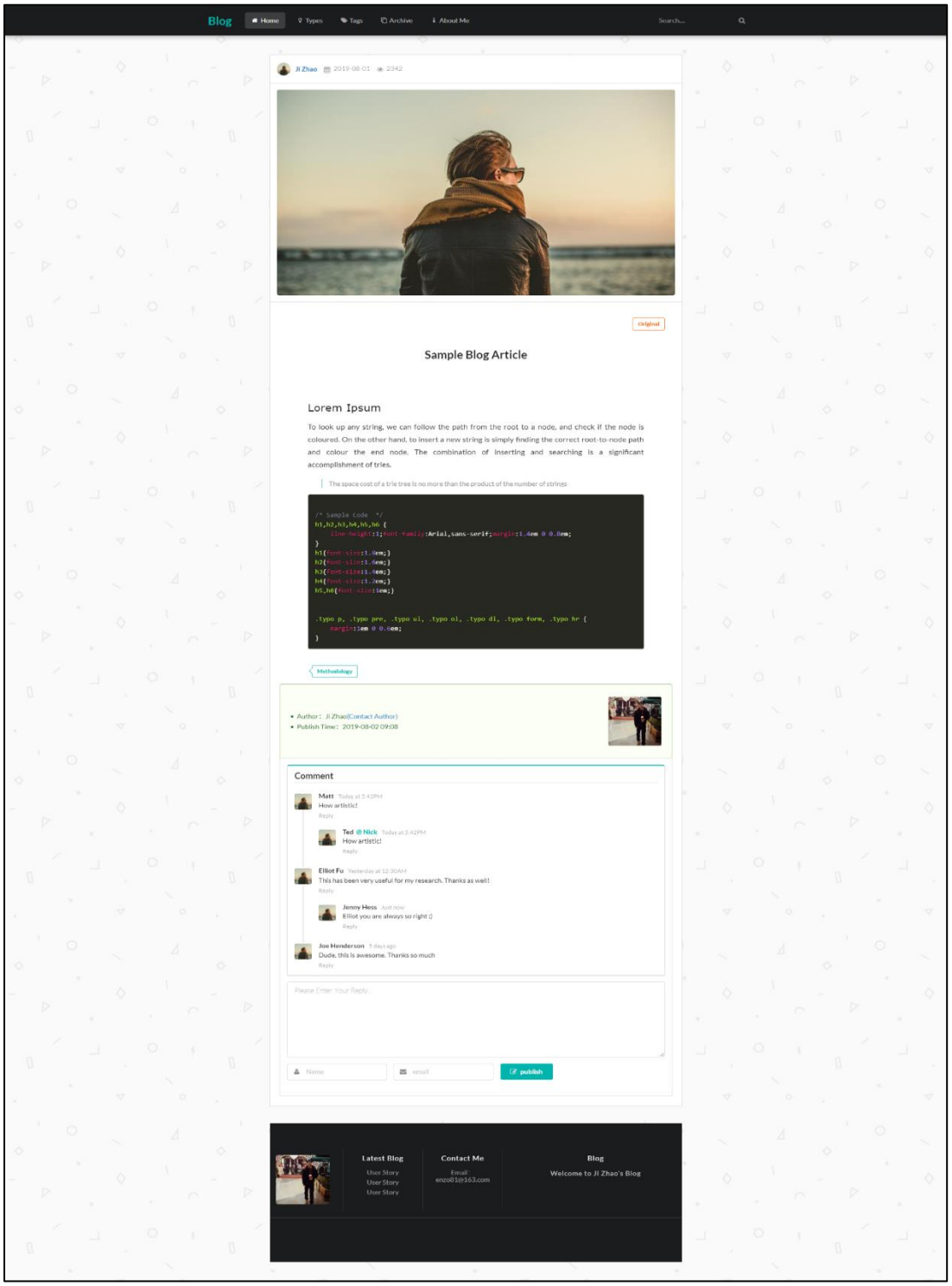

Figure 50: Blog Details Screen of the Blog System

Tags

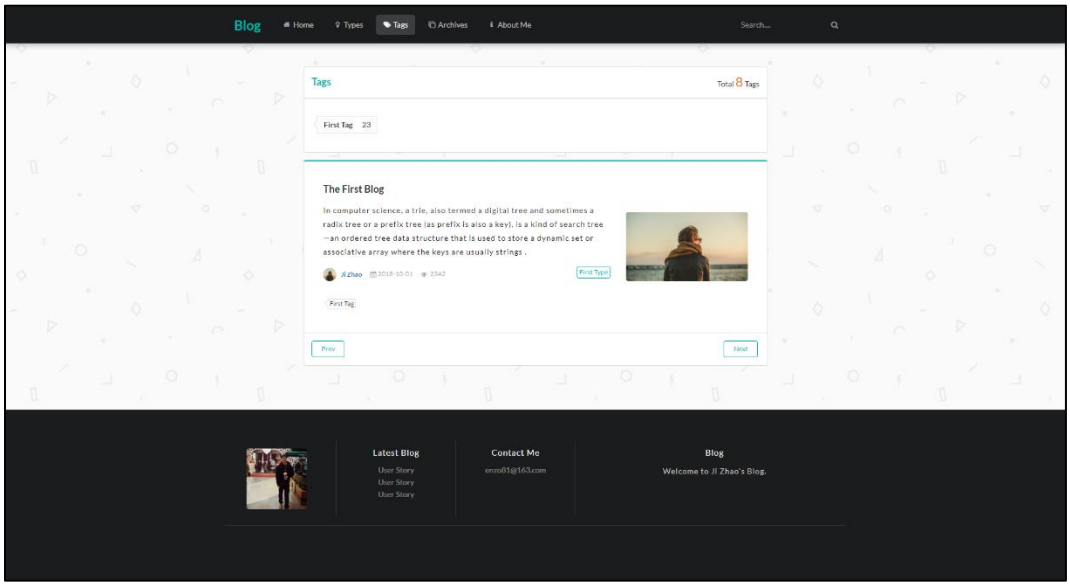

Figure 51: Tags Screen of the Blog System

About Me

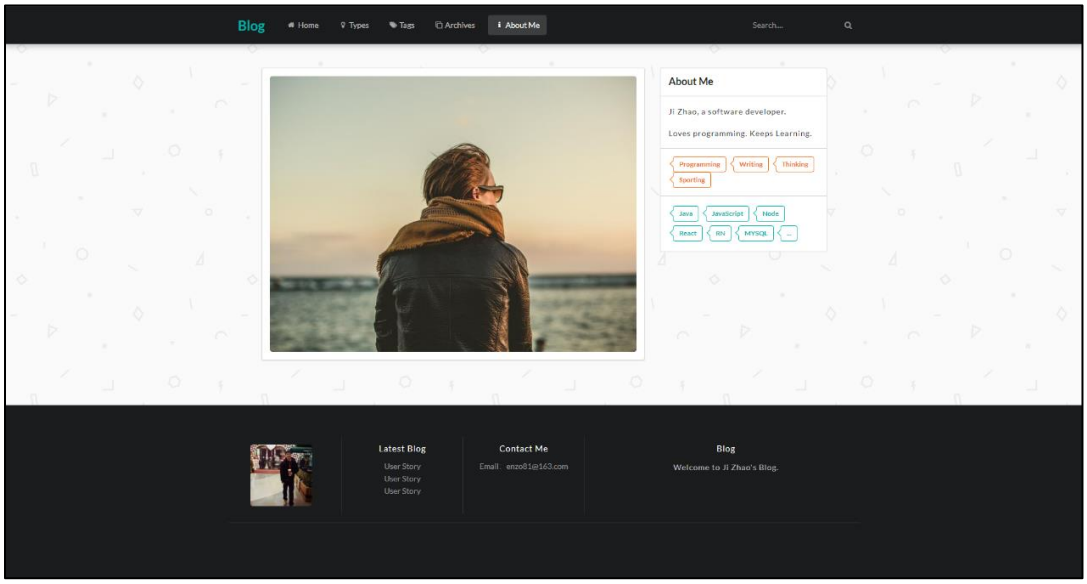

Figure 52: About Me Screen of the Blog System

Chapter 4: Conclusion

Spring and Spring Boot are powerful tools in developing web applications. With the integrated Object Relational Mapping tools such as Hibernate web applications are developed not only faster but easier. Applications developed with Spring Boot and Hibernate are robust, loosely coupled and easy to use.

The objective of this project is to learn, understand the working of Spring Boot and Hibernate and implement a ‘Personal Blog’ integrating these two frameworks. The blog can be used to write and post articles, pictures and codes by the administrator. The blog also allows general users to read and comment on the passages.

The Personal Blog System can be further enhanced by implementing some additional features such as supporting video files in the blog, share a blog to another user, and send notifications to administrator when a general user leaves comment.

References

[1] "JPA Introduction," https://www.tutorialspoint.com/jpa/jpa_introduction.htm.

[2] "Spring Boot," http://spring.io/projects/spring-boot.

[3] "Spring Boot Introduction," http://www.tutorialspoint.com/spring_boot/spring_boot_ introduction.html.

[4] C Bauer and G. King, "Hibernate in Action," Manning, 2005.

[5] "Introduction to Spring Framework," http://www.docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/overview.html.

[6] "What exactly is RESTful API," http:// stackoverflow.com/questions/671118/what-exactly-is-restful-programming.

[7] "Hibernate Overview–Tutorialspoint," https://www.tutorialspoint.com/hibernate/orm_ overview.html.

Bibliography

C. Walls and R. Breidenbach. *Spring in Action* [M]. Greenwich: Manning Publications, 2008.

R. Johnson, J. Hoeller, A. Arendsen, T. Risberg, and C. Sampaleanu. *Professional J Development with the Spring Framework.* Indianapolis, Wiley Publishing, Inc., 2005.

Hibernate, https://hibernate.org/.