

8-2018

Implementation of MD5 Framework for Privacy-Preserving Support for Mobile Healthcare

Sreevidya Pulugurta

St. Cloud State University, sspulugurta@stcloudstate.edu

Follow this and additional works at: https://repository.stcloudstate.edu/msia_etds

Recommended Citation

Pulugurta, Sreevidya, "Implementation of MD5 Framework for Privacy-Preserving Support for Mobile Healthcare" (2018).
Culminating Projects in Information Assurance. 66.
https://repository.stcloudstate.edu/msia_etds/66

This Starred Paper is brought to you for free and open access by the Department of Information Systems at theRepository at St. Cloud State. It has been accepted for inclusion in Culminating Projects in Information Assurance by an authorized administrator of theRepository at St. Cloud State. For more information, please contact rswexelbaum@stcloudstate.edu.

**Implementation of MD5 Framework for Privacy-Preserving Support
for Mobile Healthcare**

by

Sreevidya Soumya Pulugurta

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science in

Information Assurance

August, 2018

Starred Paper Committee:
Dennis Guster, Chairperson
Susantha Herath
Balasubramanian Kasi

Abstract

The improvement of science and technology has made life so easy and fast that smartphones and other touch-screen minicomputers have become the most trusted personal storage and communication devices for individuals. Comparable to the rich enhancement in wireless body sensor networks, it is valuable to the development of medical treatment to be exceptionally adaptable and become very flexible by means of smartphones through 2G and 3G system bearers. This has made treatment simple even to the common individual in the general public with less payable cash.

In this paper, we introduce privacy-preserving support for mobile healthcare using message digest where we have used an MD5 algorithm instead of AES, which can certainly achieve an efficient way and minimizes the memory consumed and the large amount of PHI data of the medical user (patient) is reduced to a fixed amount of size compared to AES which in parallel increases the speed of the data to be sent to TA without any delay which in-turn. This study implements a secure and privacy-preserving opportunistic computing framework (SPOC) for mobile-health care emergency. Utilizing smartphones and SPOC, assets like computing power and energy can be gathered to reliably to take care of intensive personal health information (PHI) of the medicinal client when he/she is in critical situation with minimal privacy disclosure.

With these, the healthcare authorities can treat the patients (restorative clients) remotely, where the patients live at home or at different spots they run. This sort of a treatment can be done under mHealth (Mobile-Healthcare). In malice of the fact that in them-medicinal services administration, there are numerous security and information protection issues to be succeed.

The main aim of this paper is to bring medical health to patients in remote locations by providing the basic triage of an emergency to increase the patient's body acceptance until they can reach a proper medical facility, in addition to providing emergency care in minimal payable cash.

Table of Contents

	Page
List of Figures	5
Chapter	
I. Introduction	7
Problem Statement	16
Objective	16
Significance of Study	16
Definition of Terms	17
II. Background and Literature Review	18
User-Centric Privacy Access Control for mHealth Emergency	19
Advanced Encryption Standard (AES)	21
Substitute Byte Transformation	22
ShiftRows Transformation	22
MixColumns Transformation	23
AddRoundKey Transformation	23
MD5	23
MOVEIt	28
III. System Analysis	30
Feasibility Study	30
Existing System	31
Proposed System	31

	4
Chapter	Page
System Requirements	33
IV. Software Environment	34
Features of .NET	34
The .NET Framework	38
MySQL	41
V. Implementation	43
Application Screenshots	44
VI. System Design	53
UML Diagrams	55
Use Case Diagram	55
Data Flow Diagram	57
VII. Testing	59
VIII. Conclusion	61
References	62
Appendix	64

List of Figures

Figure	Page
1. Pervasive Healthcare Monitoring Mobile Healthcare	11
2. Two-phase Access Control for mHealth	13
3. Basic Architecture of mHealth	14
4. AES Transformations	22
5. Message Digest Concept	26
6. Message Digest Operation	27
7. N-Tier Architecture	40
8. New Medical User Registration	44
9. User Details Being Saved	45
10. Validations on User Login	45
11. User Home Page	46
12. Patient Details on the Patient's Login Details	46
13. User Information Before Sensing	47
14. When the Sensor Starts Sensing	47
15. Once the Sensing is Completed and the Information is Gathered	48
16. Body Details Being Read by the Sensor	48
17. When the Data is Sent for Encryption	49
18. When the Encryption Process is Completed	49
19. When the Healthcare User Wants to Look at the Details	50
20. Healthcare User Login	50

Figure	Page
21. Data Accessed for Healthcare User	51
22. Data Encrypted for Use by the Healthcare Professional	51
23. Emergency Lookup of Information for Authorized Users	52
24. Use Case for Patient	56
25. Use Case for Admin	57
26. Use Flow Chart	58

Chapter I: Introduction

Information technology (IT) in medicine and health has recently been the focus of research and new innovations. The value of information technology has increased in the healthcare field, since it leads to the exponential improvement of quality of life and longevity of patients [1]. The mobile healthcare (mHealth) discipline is under constant development to allow better interaction between health professionals and patients via mobile device to support the medical decision of a patient's caretaker. Without a doubt, the mHealth framework can be performed as a critical use of pervasive processing to enhance medicinal service models, leading to more patients being reached. Similar to minute body sensor hubs planted into the human body, cell phones may also be utilized to pass on social insurance administrations to patients with chronic diseases like diabetes and heart attacks [2]. Mobile healthcare is not only able to help patients in critical situations, but also allows medical professionals to make diagnoses from across the globe easily. Mobile healthcare in this way helps keep track of patients anywhere and provides assistance remotely. This means that a patient is not required to be physically present in the doctor's facility or house [3].

Preferably, the client is equipped with a remote body sensor arrangement that is drawn up with the help of body sensor hubs. With cell phones, the patient also has the flexibility to leave their house or doctor's facilities and still get quality medical services checking by healthcare experts anytime and anywhere. For instance, every client's or patient's basic information like heartrate, glucose level, circulatory strain and other vital parameters are initially collected by the BSN. This information is accumulated with the help of cell phone by means of Bluetooth and is transmitted to a medical center by using 3G or 4G systems, regardless of the

patient's location [3]. Once the personal health information (PHI) of the patient is received, medical personnel can screen the wellbeing status of the patient and quickly react to the patient's situation with an emergency vehicle and medical professionals inside of a short time limit [2].

As mentioned here, mHealth gives a clear advantage to patients with chronic illnesses by providing them with quality medical services to check their vitals. Still, there are many issues regarding how we are notified of, apprehend, and educate others about difficulties found in mHealth, particularly amid a crisis. When all is said and done, a patient's PHI ought to be accounted for to the social insurance focus at regular intervals for typical remote checking. When the patient is in normal situation, the sensor nodes can send the PHI data readings to the healthcare center. If the patient's situation requires medical attention, the body sensor nodes will retrieve the readings from the patient's body in a smaller period of time. The readings will then transmit a large amount of data at regular time intervals of 5-10 seconds.

Be that as it may, the simple use of cell phones should not be the end goal of human services, but something more like helping patients or clients maintain their health conditions by constantly informing them of the best practices, in addition to helping maintain and achieve good health. Salina, for example, is a mobile healthcare application designed for Salina Regional Healthcare Centre. Salina allows easy access to the client's reports, medicinal usage, and some best practices via an app for their mobile devices. Through Salina, patients can instantly see their lab reports, prescriptions, health center services, and telephone directory of physician staff and hospital services. The use of a simple phone call may become deficient when a crisis occurs [3].

For a circumstance where the client does not have access to the mobile device and is in crisis is usually unforeseen, however it may occur with lower likelihood to happen. When the

number of mHealth patients increases, the albeit small likelihood still adds up to a huge estimation of medical clients who may play with this startling event of low vitality amid a crisis. Consequently, this circumstance cannot be ignored. Sharp processing, or more often coined as pervasive processing, has drawn much consideration worldwide as of late. The concept of pervasive processing is to embed computational capability into everyday objects to make an individual's task easy and more useful. In other words, the user will not have to be within reach of their computer or laptop but instead will use things they carry most with them at all times like mobile phones and watches. The Apple watch is such an example of pervasive processing.

Opportunistic computing has recently received much attention as a new pervasive computing paradigm and can be characterized by exploiting all available computing resources in an opportunistic environment to provide a platform for the distributed execution of a computing-intensive task [4], [5]. For example, once the execution of a task exceeds the energy and computing power available on a single node, other opportunistically contacted nodes can contribute to the execution of the original task by running a subset of task so that the original task can be reliably performed [4].

Obviously, opportunistic computing paradigm can be applied in an mHealth emergency to resolve the challenging reliability issue in the PHI process. In this paper, we have proposed to use the secure and privacy-preserving opportunistic computer (SPOC) framework. SPOC introduces a user-centric two-phase privacy access control to allow only those medical users who have similar symptoms to participate in opportunistic computing [6]. This framework helps provide user-centric healthcare by approved and certified medical professionals.

Module 1: Phase-I access control indicates that although a passerby has a smartphone with enough power, they are not welcomed to participate in opportunistic computing as a non-medical user since the opportunistic computing requires smartphones that are installed with the same medical application to co-operatively process the PHI [1]. If a passerby is not a medical user, the lack of necessary software does not make them an ideal helper. Therefore, the phase-I privacy access control is prerequisite.

Module 2: Phase-II access control only allows medical users who have similar symptoms to participate in opportunistic computing [7]. This is because medical users sharing symptoms are able to process the same type of PHI. When the emergency takes place at a location with high traffic, the threshold will be set high to minimize the privacy disclosure [4]. However, if the location has low traffic, the threshold should be low so a highly reliable PHI process and transmission can be first guaranteed.

Module 3: In the MD5 technique, data transferred from source to destination (here from mobile to TA) is first encrypted at the source side and then let it on track. To remove the original message or data, the decryption of that data will be continued at the destination. In this way the data is transferred from source to destination, and any attempt by a third party to compromise that particular data will be unsuccessful [7]. Another advantage of the MD5 technique is that it encrypts and decrypts the data very quickly and takes minimal time to send it from source to destination. The advantage of this is the communication between mobile and TA occurs in real time, meaning minimal delay will occur in data communication [8].

Additionally, this framework mainly establishes the disclosure of a patient's PHI. This proposal helps in computing opportunistic service with high reliability of information disclosure

to only authorized agents or medical personnel [1]. It is predicted that with the aid of the SPOC structure, every patient amid a medical crisis can carry through a client-driven protection access control, which permits only qualified partners to get an access to the client PHI information and the time spent to adjust high dependability of PHI and lessening PHI security disclosure during a crisis [2], [3]. Here this framework will help in privacy protection as when access is given to qualified partners, the patients' medical condition is in safer position where individuals with needed expertise will be at the patient's availability with adequate equipment to help them get to stable condition based on the PHI values during a crisis.

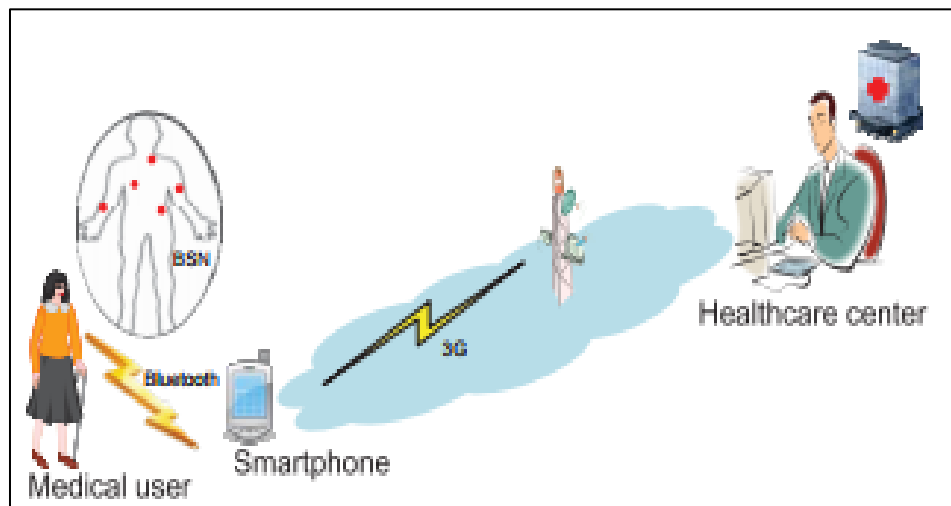


Figure 1: Pervasive Healthcare Monitoring Mobile Healthcare [9]

While there are many benefits of mobile healthcare, there are also numerous challenges in maintaining data privacy and load balancing when transferring data into safer hands. Every client of a mHealth service provider must abide by rules and regulations set forth by HIPPA, as any information relating to medical users are considered highly sensitive data requiring large security preservation. Involving various conditions and rules implied by HIPPA, user PHI is sensitive record of data which should not be corrupted in the process of transmission in the

emergency situations. Every individual having access to any patient's PHI must operate as a covered entity (CE). A CE must sign a BA agreement to safeguard the PHI.

CEs must ensure that the BA agreement includes safeguards regarding the use or disclosure of PHI to the BA, such as:

1. A description as to how the BA is permitted to use the PHI;
2. Prohibiting the BA from use or further disclosure of the PHI other than as permitted or required by the contract or by law;
3. Requiring the BA to use appropriate safeguards to prevent use or disclosure of the PHI beyond the scope of the contract [10]. BAs must then agree, in part, to:
 - Neither use nor disclose PHI, other than as permitted by the CE;
 - Use appropriate safeguards to prevent use or disclosure of PHI;
 - Mitigate harmful effects of use or disclosure of PHI;
 - Report to CEs use or disclosure of PHI not provided for by the BA-CE agreement;
 - Make internal practices, books and records related to the use and disclosure of PHI available to the Secretary of HHS upon request; and
 - Document disclosure of PHI.

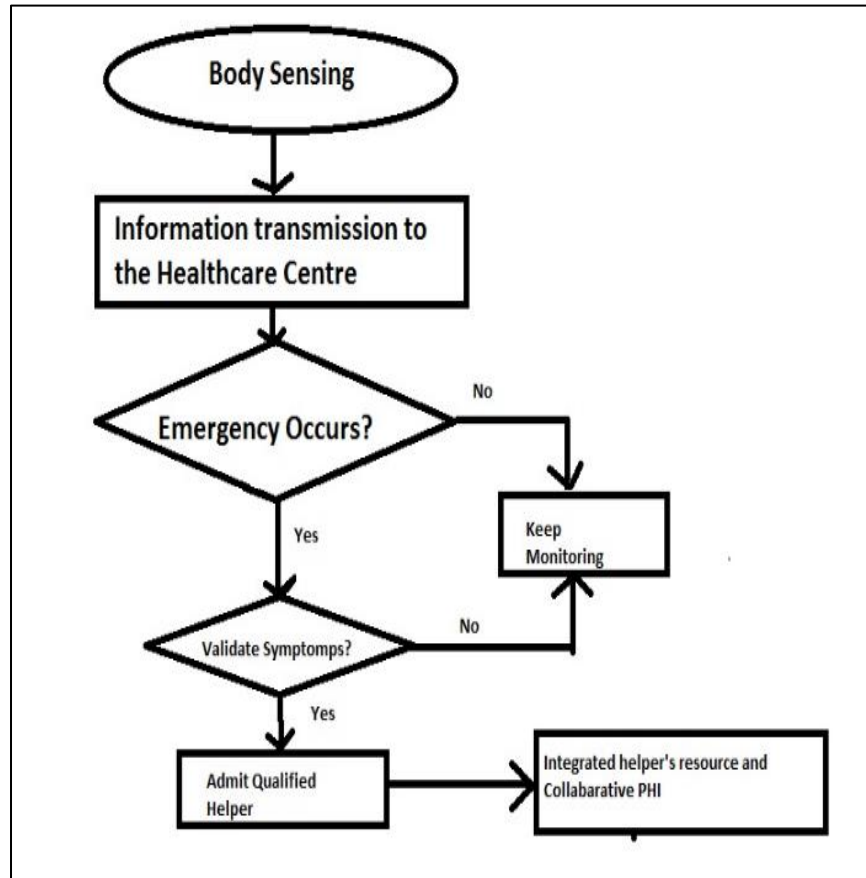


Figure 2: Two-phase Access Control for mHealth

The PHI is close to house data and extremely delicate to medicinal clients. Once the crude PHI is prepared in shrewd processing, the PHI would be exposed. So how can the high dependability of the PHI procedure be adjusted while minimizing the PHI security exposure amid the astute registering turns into a testing issue in an mHealth crisis?

In this paper, I outline a protected and secure saving processing system, known henceforth as SPOC, to help resolve the difficulties mentioned earlier. The proposers of this system suggest that with the aid of the SPOC structure, every medicinal client amid a crisis condition can complete a more client centric which permits access their PHI details [3].

Not all health providers are able to prescribe implants for every patient, however, as these might have other greater benefit for a patient's wellbeing. HIPPA (Health Insurance Portability and Accountability Act), ensures they have an agreement with the service provider as a Business Associate, as they will be providing services through covered entities as the PHI is disclosed. This essentially means that the provider will be dealing with sensitive material like financial, management and accreditation information.

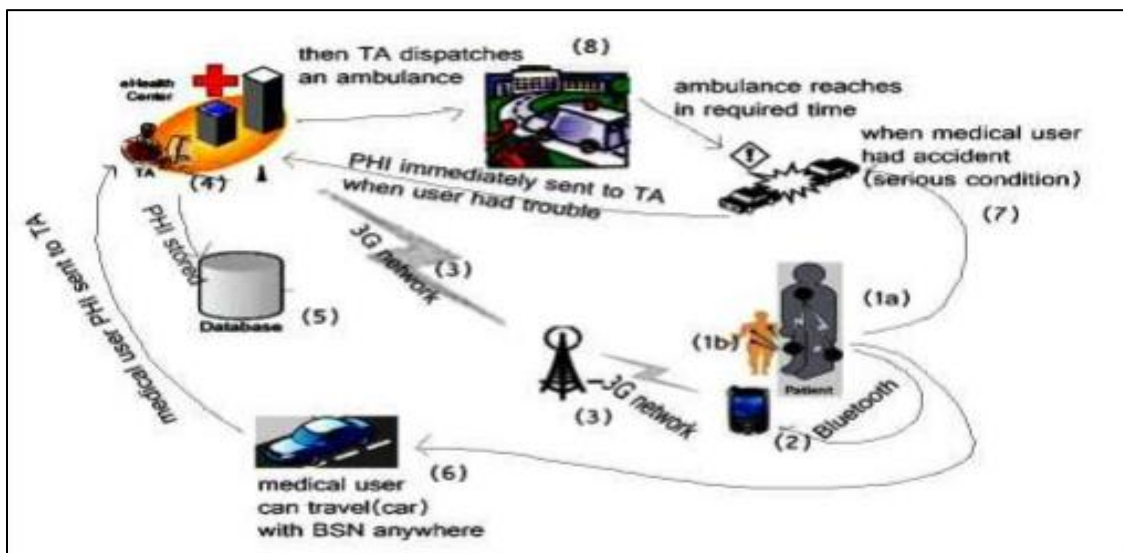


Figure 3: Basic Architecture of mHealth [3]

The average patient's smartphone is mainly used for normal functions such as phone calls, chatting, playing videos, listening to music and browsing the internet; this can lead to issues like a low or dead battery during an emergency. For example, if 10,000 emergency cases are considered, and common events such as general checkups or general prescription details using m-healthcare amount reaches 50, the probability lies at 0.5%. While this may be a low probability, the chance of this event still exists. However, this probability outstandingly indicates the actual reliability regarding mHealth system during an emergency.

Using BSN and the mobile devices or smartphones is important as we give authorized users access to assist patients in emergency situations. From the above figure it clearly depicts the following:

(1a) indicates that a patient when after registers for remote monitoring in TA he was assigned BSN into his body, and (1b) shows the same sensor nodes inserted to a patient. Likewise, several patients are registered and implanted sensor nodes (as seen in (6) and (7)) indicates that the patient can travel anywhere without difficulty. (2) shows the patient's readings gathered by the mobile device via Bluetooth as shown in. (3) and (4) show all the data of the user is sent through Bluetooth via secure 3G networks to a medical center, where the data is monitored by a trusted authority. (5) shows where the medical user's data is monitored and checked by the TA and securely stored in a database. (7) shows how medical representatives are notified of a user's change in health when met with a critical situation, or if a user's readings are no longer stable. The representative then dispatches an ambulance within a timely manner to ensure the user's safety and/or wellbeing, as shown in (8).

Expounded security investigation portrays that this proposed SPOC system can productively accomplish client-driven protection access control in mHealth crisis. In this report, we present privacy-preserving support for mobile healthcare utilizing message digest where we have utilized MD5 calculation rather than AES, which can absolutely accomplish efficiency, minimize the memory consumed, and ensure the huge amount of PHI about the therapeutic client (patient) is diminished to an altered standard of size as opposed to AES [2], [7]. This will increase the rate at which the information is sent to a designated trusted authority (TA), who can be entrusted with the medical staff focus and will help in presenting client's PHI to experts,

possibly sparing their lives in record time. Additionally, the calculation gives tight security in transporting the patients PHI to TA.

Execution assessments with broad reenactments clarify the message digests (MD) viability in terms of giving a highly dependable PHI procedure and transmission while diminishing the security exposure amid mobile healthcare crisis.

Problem Statement

According to the given mHealth scenario, there is a need to provide security and efficient storage to the user system. Here we propose the SPOC framework where the security for the user is enhanced. We provide user-centric privacy access control in computing emergency health care for registered clients. SPOC framework provides a platform where resources available on the mobile devices of other opportunistically contacted medical users can be gathered together to deal with the computing-intensive PHI process in an emergency situation. Also, this will validate the effectiveness of the proposed SPOC framework in mHealth emergency.

Objective

This paper's objective is to suggest the use of MD5 algorithms instead of AES for mobile healthcare using message digest, which can minimize the memory consumed and the large amount of PHI data of the medical user (patient) is reduced to a very small size compared to AES, which in turn increases the speed of the data to be sent to TA without any delay.

Significance of Study

The proposed application implements attribute-based encryption for high level data security, providing a secured network architecture between patient, service provider, and

healthcare system. It also allows both patients and the doctors to monitor their health condition, as well as any preliminary measure to be taken.

Definition of Terms

PHI – Personnel Health Information

SPOC – Secure Privacy and Opportunistic Communication

TA – Trusted Authority

MD – Message Digest

AES – Advanced Encryption System

PPSPC – privacy protective real computation protocol

HIPPA – Health Insurance Portability and Accountability Act

Chapter II: Background and Literature Review

The current frameworks present the pioneering figures worldview in remote sensor system to assume charge of the issue of putting away and executing an application that passes the memory assets accessible on a solitary sensor hub [3]. Particularly, the result depends on deploying the application code into various business modules. Every hub is added to the performance of the first application by running a subset of the application errands, giving financial backing of the neighboring hubs, and assessing the performance of administrative execution in shrewd figuring. This study is to complete a scientific model describing the administration summon procedure in the midst medical users and the healthcare professionals and deduce the ideal number of transcripts to be brought forth on experienced hubs, with the specific end goal to minimize the implementation time and streamline the computational and data transmission assets utilized [11]. The main goal is to minimize the computational time and reduce the time needed to reach the affected user.

The use of medicinal services is anticipated to rise to 15.9% by 2018. The disbursement of medical services for the global maturing populace has turned into a national business organization are imperative for determining how the artful processing worldwide work when assets accessible on distinctive hubs can be shrewdly assembled to give wealthier useful, they have not believed the potential security and protection issues existing in the sharp figuring world view [1].

According to Kim Jung Tae, from his paper about Secure Devices for Mobile device in health care, the medical industries have adapted to information innovation with mobile gadgets and remote correspondence [12]. The advent of mobile human services systems can profit

patients and healing facilities by providing superior patient care, in addition to diminishing regulatory and medicinal expenses for the both patients and clinics. Security issues show an intriguing examination subject in remote and unavoidable medicinal services systems [12]. As information innovation is produced, numerous associations like government organizations, open foundations, and partnerships have utilized information systems to improve the productivity of their work forms. In recent years, medicinal service associations all through the world have been receiving health information systems (HIS) in light of the remote system foundation [4]. As a piece of the remote system, a mobile operator has been utilized at an extensive scale in healing centers because of its extraordinary versatility. A few vulnerabilities and security prerequisites identified with mobile gadgets ought to be considered when executing mobile administrations in medical facilities [7]. Secure validation and conventions with a mobile operator for applying omnipresent sensor arranges in a medicinal services system condition is proposed for constant monitoring.

User-Centric Privacy Access Control for mHealth Emergency

Imagine a basic circumstance occurring to medicinal client in mHealth, henceforth known as U_0 . Let U_j represent the individual wellbeing data of medicinal client two. If U_0 falls unconscious within a matter of seconds, the social insurance delegates will handle and control that U_0 's circumstance. At this time, an emergency vehicle and a therapeutic or health care agent will be sent to the U_0 's location. Typically, the vehicle will be dispatched within the time span of 15 to 20 minutes [7]. In the meantime, the client will require continuous monitoring on the client's PHI.

However, U0's mHealth device may not be adaptable to participate in the high-concentrated PHI procedure and transmission. Entrepreneurial registering is dispatched, as shown in Figure 3, and the specific client-driven security access control is prepared to decrease the PHI protection presentation in sharp processing.

Stage I: The goal of this stage is to get control and find if another patient is in crisis. Let U0 and Uj represent the individual wellbeing data of two patients. To gain control, U0's mHealth device first chooses an arbitrary number. When quiet Uj passes by the crisis place, U0 sends their client data (CData). Here CData can be any information like U0 PHI or the individual health information critical to the crisis. In the wake of accepting CData, Uj will handle the following steps:

- Uses patient's access control key to register data;
- Compute Authentication Auth, where timestamp is the current timestamp, and returns Authktimestamp to U0. At the point when client U0 gets Authktimestamp at time timestamp0, they first check the legitimacy of the time between timestamp0 and timestamp so as to avoid repetitive attacks. When Authorized timestamp is allowed, U0 acknowledges the Auth0. On the off chance that it holds, Uj is validated as a patient and permits the stage I control [2].

Stage II: Once Uj passes stage I, U0 and Uj will continue preparing the stage II access control to see whether they both had adjacent signs to uncover PHI. U0 first characterizes a normal limit for the quantity of regular signs. At that point, they figure a protection saving course for both U0 and Uj. Since the PPSPC convention guarantees neither, U0 nor Uj will uncover

their own social insurance data to each other amid the calculation it can productively accomplish security safeguarding access control.

Advanced Encryption Standard (AES)

NIST recommended AES as the latest encryption standard in 2001, replacing the DES algorithm. The AES algorithm supports many combinations of data like 128, 192, 256 bits, etc. Depending on the length of the key used in encryption or decryption techniques, the DES algorithm is termed as AES128, AES 192, and AES256. AES has 10, 12, or 14 rounds for 128, 192, or 256 bits respectively. These rounds take plain text as input and generate the final cipher text as output. In AES, the 128-bit data block is partitioned into four separate operational blocks where each block represents an array of bytes. These arrays are arranged in a matrix form of 4 rows and 4 columns known as a state [9].

In both encryption and decryption techniques, the first operation that takes place is an AddRoundKey. In the next step, the output undergoes nine more steps where each round involves four different transformations of data. These transformations include SubBytes, ShiftRows, MixColumns, and AddRoundKey. No MixColumns transformation takes place in the final tenth round [13]. The below figure depicts the complete transformation and encryption process for AES. The decryption process is similar to the AES encryption process, with the difference being the sequence of steps is reversed and inverse functions like Inverse SubBytes, Inverse ShiftRows, and Inverse AddRoundKey are used. All other steps remain the same [14].

Each round of AES undergoes the below mentioned transformations:

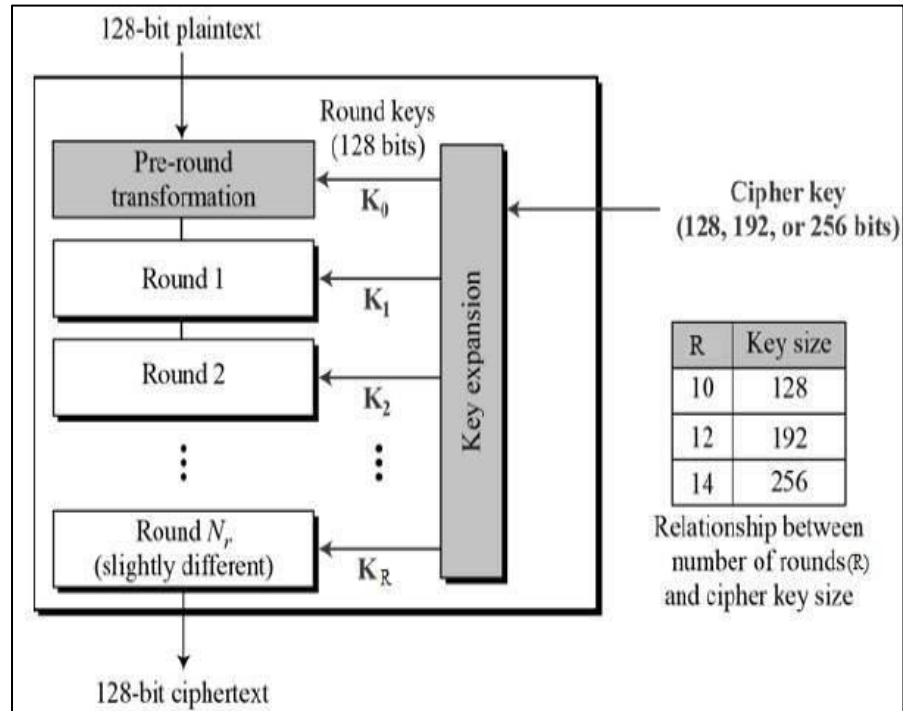


Figure 4: AES Transformations

Substitute Byte Transformation

AES consists of 128-bit blocks of data, where each block consists of 16 bytes. In this SubByte transformation step, each 8-bit byte of a data block is converted to another block with the help of an 8-bit substitution box that is termed as Rijndael Sbox [15]. This box is used as a lookup table to get the replacement of bytes for transformation.

ShiftRows Transformation

ShiftRow transformation is a simple step where bytes present in the last 3 rows of the state are cyclically shifted based on the location or index of the row. For instance, if one byte in the second row is shifted left in a cyclic manner in second row, two and three bytes in third and fourth rows are similarly shifted left in a cyclic manner respectively [16].

MixColumns Transformation

MixColumns transformation is similar to the logic of a matrix multiplication. In this step, a matrix multiplication is applied to each column in the state [2]. A constant matrix is taken as reference and is multiplied to each column. The uniqueness of this transformation is that the bytes of the state are taken as polynomials instead of integers or numbers.

AddRoundKey Transformation

AddRoundKey is equivalent to applying a bitwise XOR operation to the 128 bits of a state and 128 bits of the round key. Consequently, this transformation applies in its own inverse. Classical encryptions are also known as asymmetric-key algorithms. These classical algorithms are a class of algorithms of cryptography that are loosely related, often identical, and cryptographic for both decryption and encryption. The encryption key may appear to be identical to the decryption key, but there is a simple transformation to go between the two keys. The keys in practice represent a shared secret between two or more parties that can be used to maintain a private information link.

MD5

A hash function is an algorithm that takes a random block of data and returns a fixed-size bit string, the cryptographic hash value, such that any accidental or intentional change to the data will change the hash value. The data to be encoded is often called the "message," and the hash value is sometimes called the message digest or fingerprint.

A hash function is usually considered secure only after the following conditions are satisfied:

- It is impossible to find a message that corresponds to a given hash code. This is sometimes referred to as the one-way property of a hash function [3].
- It is impossible to find two different messages that generate the same hash code value. This is also referred to as the strong collision resistance property of a hash function.
- It is impossible to alter a message without changing the hash.

The concept of MD5 is similar to the hash function. The MD5 algorithm takes an input of variable length and produces a message digest that is 128 bits long, known as the output. The message digest is called the “hash” or “fingerprint” of the input [3]. However, it has been proven that a 128-bit output can find a security hazard in the algorithm. This leads one to reason that a 128-bit hash output may not offer a sufficient amount of security protection within the near future. To provide higher security protection, extending the hash output accepts the same input format as that of MD5, and produces a 512-bit hash [3]. Since MD5 is still the most broadly used hash algorithm, improving the current implementation in the future to a MD5 512-bit hash is much easier within an integrated structural design.

The process of MD5 contains the following steps: preparing the input by adding bits and appending data length, initializing MD5 buffer, processing message in 16-word block, and output [2], [9].

Step 1. Padding bits. Pad the message with the 1 bit first. Then pad the message with as many 0 bits until the resulting bit length equals $448 \bmod 512$. The bit length of the original message is now a 64-bit integer. The total bit length of the padded message is $512N$ for a

positive integer N . The padded message is partitioned into N successive 512-bit blocks M_1, M_2, \dots, M_N .

Step 2. The Initializing MD5 buffer. A buffer (4 registers) is used to hold the intermediate and final result of the hash function that are each 32 bits long known as chaining variable. The buffers (A, B, C, and D) are defined as:

A= 08 ab 32 ef

B= 98 ba dc f4

C= fe dc ba 98

D= 76 54 32 10

Step 3. Processing the blocks. The contents of the four buffers (A, B, C and D) are now mixed with the words of the input, using the four auxiliary functions (F, G, H and I). There are four rounds, each of which involves 16 basic operations. The auxiliary function F is applied for four buffers (A, B, C and D) using message word M_i and Key K_i . The item “ $\lll s$ ” denotes the binary shift by s bits. Then each auxiliary function takes an input of 32 bits and outputs 32 bits.

The logical operators AND, OR, XOR and NOT are applied to the input bits.

$F(X, Y, Z) = XY \vee \text{not}(X) Z$

$G(X, Y, Z) = XZ \vee Y \text{ not } (Z)$

$H(X, Y, Z) = X \text{ xor } Y \text{ xor } Z$

$I(X, Y, Z) = Y \text{ xor } (X \vee \text{not } (Z))$

Step 5. Output: After all rounds have been performed, the buffers A, B, C and D contain the MD5 digest of the original input.

MD5 is a widely used cryptographic function with a 128-bit hash value that has been employed in a wide variety of security applications and is also commonly used to check the integrity of files. An MD5 hash is typically expressed as a 32-digit hexadecimal number and is one of the most common hash function [2], [8]. Its basic principle is to process the input information in groups divided by 512 bits, with each group divided into 16 sub-groups with 32 bits. After a series of processing, the algorithm output composed by 4 groups with 32 bits and cascade this 4 groups will generate a hash value with 128 bits. In the process of MD5 algorithm, fill information first to make its length 64 bits less than any multiples of 512. The filling method is to attach a 1 and millions of 0s before adding information length and filling indicated by binary system with 64 bits. These two steps are to make the information length an integer multiple of 512 and ensure the difference after different information filling.

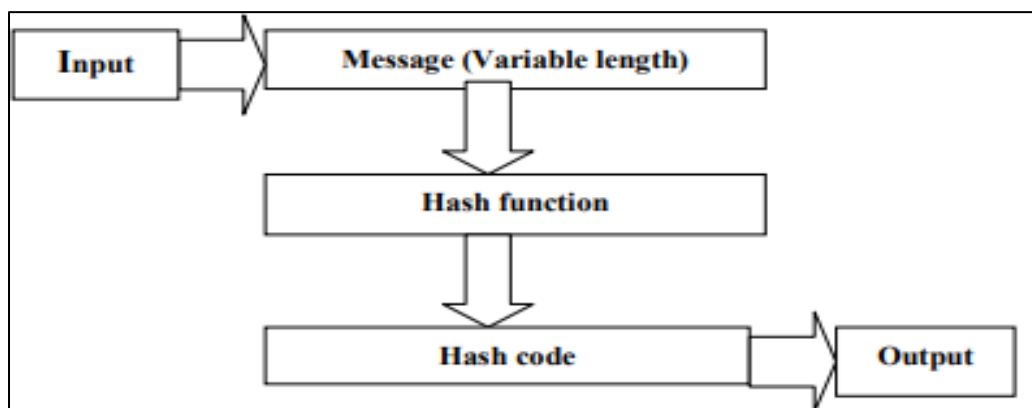


Figure 5: Message Digest Concept

The concept of message digest is that the input is transformed into the message at an acceptable length to apply the hash function on the input. Once the hash function is applied, the hash code is applied. The generated value after applying the hashing is sent as an output.

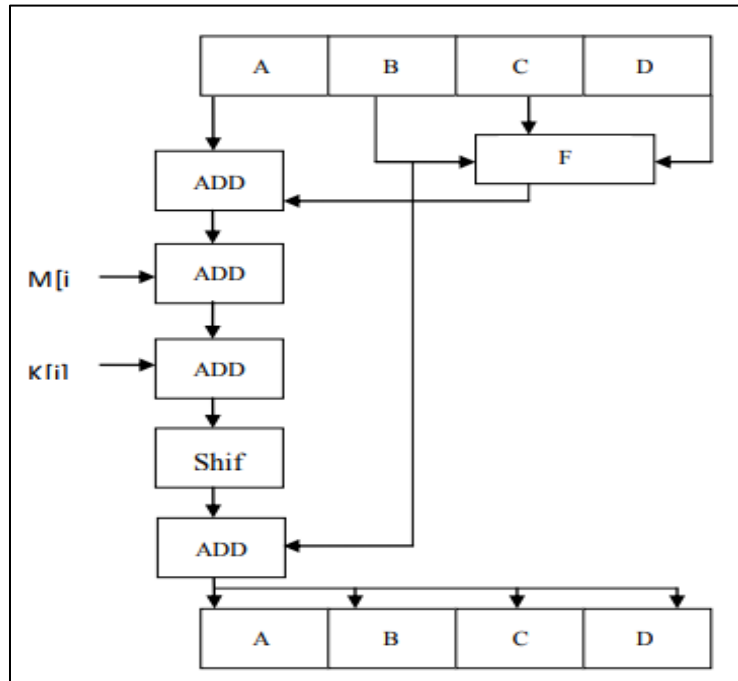


Figure 6: Message Digest Operation

There are currently three versions of message digest: MD2, MD4, MD5. There are major differences between the three versions. Each have a different method of optimization. Here are major differences described below:

MD2 –

- This algorithm was optimized for 8-bit computers.
- This was developed in 1989.
- This was fast, simple, and used a considerably less amount of CPU.
- In this version, a 16-bit checksum is appended to the length of the message.

MD4 –

- This algorithm was developed in 1990.
- In this the enhancement of the algorithm, the 128-bit carries a message.

- In this version, 3 rounds of 16 steps was designed [4].
- At its conception, this version originally claimed to be collision resistant.

MD5-

- This was proposed in the year 1992.
- In this version, a variety of both 64- and 128-bit hash code is generated.
- In this, 4 rounds of 16 steps computing is performed.
- In this, collision to certificates might be possible [17].

MOVEIt

MOVEIt is a third-party control service and administrative function. This service will need a locally installed copy of its API to able to function. This can be as MOVEIt Transfer Java class while using in Java programming or MOVEIt Transfer API COM API in case of .NET applications.

While using it in Java applications it will require the Sun Java v.1.4.2 or higher (v.1.5 preferred) version. This class provides a precompiled, standard FTP-like command-line client interface for use by mainframe JCL, Unix/Linux shell scripts, and local operating system schedulers like Cron [15].

The API COM component typically helps the developers with file transfers while using languages like ASP.NET, ASP, Access, C++, .NET, VBScript, or any Microsoft windows compatible software's. This API helps the developers by providing a precompiled interface which acts more like FTP command-line interface, which can be used by FTP/FTPS scripts and batch files, as well as by the Windows Event Scheduler [15].

MOVEIt generally provides the MOVEIt Transfer, MOVEIt Cloud, and MOVEIt automation. While using the secure transfer interface this will help extend the transfer capabilities for administrators to maintain control over the user access controls and permissions to data and files. Any developer who is interested in using or accessing this interface via its API interface will have to be authorized and authenticated with a proper username and other factors like passwords or an HTTP certificate. All communications between the API interface and its API clients are protected by the 256-bit SSL encrypted, firewall-friendly HTTPS protocol, which uses only a single firewall port (443) [13,] [16]. During the process of file transfer between the API interface and the API client, the communication is secured by providing a 256-bit SSL encryption and HTTPS protocol using port 443. The file transfer using this interface provides more efficiency for the communication as it employs the following policies- integrity checks, automatic transfer retry, non-repudiation, and guaranteed delivery mechanism. MOVEIt also uses SSL certificates and SSH key where the developers can use the COM API during the programming process for management of the certificates. With the help of this interface it gives the developers the opportunity to automatically imported, added, removed, and listed to user accounts using the SSH keys and SSL certificates. The MOVEIt API interface is a separately priced and licensed option that permits unlimited use of the interface and the MOVEIt API Java class, the COM component and their command-line interfaces [15].

This is modified version of md5sum.vbs to prevent files with the same MD5 hash which will be generated via the same task from being transferred to destinations.

Chapter III: System Analysis

Feasibility Study

Feasibility study of the project is performed to derive a business proposal to determine the cost estimates. This study is performed and discussed with business to investigate various logistics required. For feasibility analysis, some basic comprehension of the real necessities for the system is required. The feasibility study can only be completed by performing the complete analysis of economic feasibility, technical feasibility and social feasibility of the project.

Economic feasibility. This study is performed to check the monetary effect that the system will have on performing any implementation in this direction. The organization will measure the amount of innovative work to be put on the project. Based on the study results, the organization will try to decide if the project can be a cost effective one. The organization will also take a study on the logistics of resources and software frameworks needed for the implementation.

Technical feasibility. This study is performed to check the technical feasibility; that is, the technical necessities of the system. Organization will look into the technical aspects needed for the organization like software platforms, project hosting, and implementation. Scope for future enhancement is also taken into consideration.

Social feasibility. This part of study is to check the level of acknowledgment of the system by the end user of the application. The organization must understand that the end user should not feel undermined by the system but should rather acknowledge it as a need. User acceptance criteria is one of the deciding factors for this study.

Existing System

Here in system initialization, the algorithm used is AES. This is necessary, as the algorithm cannot transmit the PHI data of a medical user who is in a critical situation. At this particular time, the sensor nodes will be too busy obtaining readings from the body nodes. This activity produces a huge amount of data within a small period of time, which the AES algorithm will not be ready to encrypt as it takes a lot of time to be sent and occupies a large amount of memory [9]. If in critical condition, the user data is to be sent without any delay so that they can help the user in time to save or protect his/her life. Because of this, the MD5 (message digest) algorithm is used to encrypt the large size of data to a fixed amount of size. Without any delay, the data is transmitted to the healthcare center and the patient's life is protected.

Proposed System

Our proposed SPOC framework aims at security and privacy issues, developing a user-centric privacy access control of opportunistic computing during an mHealth emergency. In this paper, a new secure and privacy preserving opportunistic computing framework, called SPOC, is proposed to address this challenge [17]. With the proposed SPOC framework, each medical user in an emergency can achieve the user-centric privacy access control to allow only those who are qualified to participate in the opportunistic computing to balance the high-reliability of PHI process and minimizing PHI privacy disclosure in an mHealth emergency. Specifically, the main contributions of this paper are threefold.

First, the SPOC is a secure and privacy-preserving opportunistic computing framework for mHealth emergency. With SPOC, the resources available on other opportunistically contacted medical users' smart phones can be gathered together to deal with the computing

intensive PHI process in an emergency situation [12]. Since the PHI will be disclosed during the process in opportunistic computing, SPOC introduces a user-centric two-phase privacy access control to only allow medical users who have access privileges to participate in opportunistic computing. This minimizes the PHI privacy disclosure.

Second, to validate the effectiveness of the proposed SPOC framework in an mHealth emergency, we also develop a custom simulator built in Java and .NET. Extensive simulation results show that the proposed SPOC framework can help medical users to balance the high-reliability of PHI process and minimize the PHI privacy disclosure in an mHealth emergency. The following module is used send the messages from human health monitoring smart phone to opportunistic user smart phone and in turn to server via wireless router:

- Step 1: Connect the smart phones and server to a wireless router.
- Step 2: Obtain the IP address of the smart phones and server.
- Step 3: Check whether all IP addresses are in the same network.
- Step 4: Register the opportunistic IP address and server IP address.
- Step 5: Run the package file.
- Step 6: Now the health professional monitoring the user's smart phone will receive PHI data from the file.
- Step 7: After receiving the file, the PHI data is sent to the server through the opportunistic user's smart phone.

In this project we have implemented a new COM -API to transfer the DATA into the trusted authority. The API in implementation is the MOVEIt API, which is provided by the MOVEIt third party control service. Also, we have decided to go this API provided by MOVEIt

with the MD5 architecture as the output size will for this algorithm will be constant, i.e., 128 bits irrespective input. Using MD5, the input cannot be reversible as it is a hashing algorithm.

Encryption is usually more expensive than hashing, as the time consumed is variably high.

System Requirements

Hardware requirements.

System : Pentium IV 2.4 GHz.

Hard Disk : 500 GB.

Monitor : Any (1).

Mouse : Any (1).

Keyboard : Any (1).

Ram : 2 GB.

Software requirements.

Operating system : Windows 7/Linux.

Coding Language : C#, .NET

IDE : Visual Studio

Data Base : MySQL DB

Browser : Chrome/ IE/ Firefox

Chapter IV: Software Environment

Features of .NET

Microsoft .NET is an arrangement of Microsoft programming advancements for quickly building and incorporating XML Web administrations, Microsoft Windows-based applications, and Web arrangements. The .NET Framework is a dialect-unbiased stage for composing programs that are also able to interoperate. There is no dialect hindrance with .NET: there are various languages accessible to the designer, including Managed C++, C#, Visual Basic and Java Script. The .NET system gives the command to segments to cooperate consistently, regardless of whether locally or remotely on various stages. It institutionalizes basic information sorts and interchanges conventions with the goal that segments made in various languages can interoperate without much of a stretch [13]. ".NET" is likewise the aggregate name given to different programming segments based upon the .NET stage. These will be the two items (Visual Studio.NET and Windows.NET Server, for example) and administrations (like Passport, .NET My Services, etc.).

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.

The CLR is portrayed as the "execution engine" of .NET. It creates the world inside which programs run. The most critical highlights are converting a low-level constructing agent-style dialect, called Intermediate Language (IL), into code local to the stage being executed on; memory administration, outstandingly including refuse accumulation; checking and authorizing security limitations on the running code; and loading and executing programs, with variant

control and other such features. The following highlights of the .NET structure are additionally notable.

Managed code. Managed code is code that directs .NET and contains certain additional information—"metadata"—to portray itself. While both managed and unmanaged code can keep running in the runtime, just the code contains the data that permits the CLR to ensure, for example, safe execution and interoperability.

Managed data. While managed code is being used, managed data enters the picture. CLR gives memory distribution and performs garbage accumulation. A few .NET languages utilize managed data as a matter of course, for example, C#, Visual Basic.NET and JScript.NET, while others, to be specific C++, do not use it. Targeting CLR, depending on the dialect you're using, forces certain requirements on the highlights accessible. Likewise, with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications—data that does not get garbage collected and will be taken care by unmanaged code.

Common type system. The CLR utilizes something many refer to as the Common Type System (CTS) to entirely authorize sort security. This guarantees all classes are perfect with each other, by portraying sorts regularly. CTS characterize how it sorts function inside the runtime, which empowers sorts in a single dialect to interoperate with sorts in other dialects, including cross-dialect exemption dealing. Additionally, the runtime also guarantees that code does not try to get to memory that hasn't been designated to it, demonstrating that sorts are just utilized as a part of proper ways.

Common language specification. The CLR gives work to assist dialect or language interoperability. To guarantee managed code that can be completely utilized by designers using

any programming dialect can be created, an arrangement of dialect highlights and standards for utilizing them called the Common Language Specification (CLS) has been characterized. Parts that take after these standards or disclose just CLS highlights are viewed as CLS-agreeable.

The class library. .NET gives a solitary established chain of importance of classes, containing more than 7000 sorts. The base of the namespace is called System; this contains basic sorts like Byte, Double, Boolean, String, and Object. All items mentioned come from System and in addition objects are under esteem sorts. Value types or sorts can be distributed on the stack, which can give valuable adaptability. There are additionally productive methods for changing over esteem sorts to protest sorts if, and when, it is essential.

The arrangement of classes is quite exhaustive: giving collection, record, screen, and system I/O, threading, et cetera, and also XML and database network. The class library is subdivided into various sets or namespaces, each giving particular ranges of usefulness, with conditions between the namespaces kept to a base.

Languages supported by .NET. The multi-dialect ability of the .NET Framework and Visual Studio .NET empowers engineers to utilize their current programming aptitudes to assemble a wide range of uses and XML Web administrations. The .NET system underpins new forms of Microsoft's old top picks Visual Basic and C++ (as VB.NET and Managed C++), yet there are various new options added to the family.

Visual Basic .NET has been refreshed to incorporate numerous better-than-ever dialect includes that make it an effective protest arranged programming dialect. These highlights incorporate legacy, interfaces, and over-burdening, among others. Visual Basic also now bolsters the care of organized special cases, custom qualities, and further reinforces multi-threading.

Microsoft Visual J# .NET gives the most straightforward change to engineers using Java while working in the universe of XML Web Services, and drastically enhances the interoperability of Java-dialect programs with existing programs written in an assortment of other programming languages.

Dynamic State has made Visual Perl and Visual Python, which empower .NET-mindful applications to be worked in either Perl or Python. The two items can be incorporated into the Visual Studio .NET condition. Visual Perl incorporates support for Active State's Perl Dev Kit.

C#.NET is additionally agreeable with CLS (Common Language Specification). CLS is set of guidelines and advances that are strengthened by the CLR (Common Language Runtime). CLR is the runtime condition given by the .NET Framework dealing with the execution of the code, while also influencing the advancement to process less demanding code by giving administrations.

C#.NET is a CLS-agreeable dialect. Any items, classes, or segments that were made in C#.NET can be used as a part of some other CLS-agreeable dialect. Furthermore, we can utilize items, classes, and segments made in different CLS-consistent languages in C#.NET. The use of CLS guarantees finished interoperability among applications, paying little attention to the languages used to make the application.

Constructors and destructors. Constructors are utilized to instate objects, while destructors are utilized to annihilate them. Destructors are used to discharge the resources designated to the question. In C#.NET, the sub finish strategy is accessible. The sub conclude system is utilized to finish the errands that must be performed when a question is pulverized. The

sub conclude method is called consequently when a protest is demolished. Furthermore, the sub finish strategy can be called just from the class it has a place with, or from inferred classes.

Overloading. Over-loading is another element in C#. Over-burdening empowers us to characterize various systems with a similar name, where every method has an alternate arrangement of contentions. Other than utilizing over-burdening for strategies, we can use it for constructors and properties in a class.

Multithreading. C#.NET plays with the concept of multithreading. An application that incorporated multithreading can deal with different issues and load issues through the program. It can also use multithreading to diminish the time taken by an application to react to its computation.

Structured exception handling. C#.NET strengthens organized execution, which empowers us to distinguish and remove mistakes at runtime. In C#.NET, we have to utilize Try, Catch, and Finally blocks to make special case handlers. Using these blocks, exemption handlers can be used to enhance the execution of our application.

The .NET Framework

The .NET Framework is a unique processing stage that streamlines application onto the Internet.

Objectives of .NET framework.

1. To give a steady protest situated programming condition, whether question codes are put away and executed locally on the Internet or executed remotely.
2. To give a code-execution condition limiting programming organization and ensuring safe execution of the code.

3. Take out execution issues. There are distinctive sorts of utilization; for example, Windows-based applications and Web-based applications.

This system was developed in .NET framework using C#, ASP.NET, ADO.NET, SQL Server, and N-Tier architecture to ensure the availability, integrity, and confidentiality of the data. Applications built on N-Tier architecture can easily implement the concepts of distributed application design and architecture. The N-Tier applications provide strategic benefits to enterprise solutions.

- Improved security because of data abstraction,
- System availability,
- Makes the application management easy,
- Code maintenance is not complex.

Simply stated, an N-Tier application helps us distribute the overall functionality into various tiers or layers:

- Presentation Layer
- Business Rules Layer
- Data Access Layer
- Database/Data Store

Each layer can be developed independently of the other if it adheres to the standards and communicates with the other layers, as per the specifications.

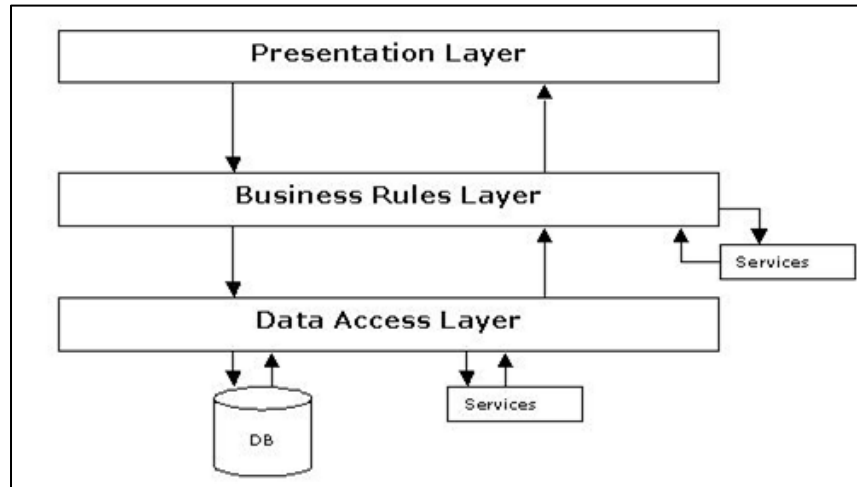


Figure 7: N-Tier Architecture

ADO.NET. ADO.NET uses a multilayered architecture that revolves around a few key concepts, such as connection, command, and dataset objects. However, the ADO.NET architecture is quite different from the classic ADO [18].

ADO.NET Data Providers. A data provider is a set of ADO.NET classes that allows you to access a specific database, execute SQL commands, and retrieve data. Essentially, a data provider is a bridge between your application and a data source [18].

The classes that make up a data provider include the following:

- **Connection:** object used to establish a connection to a data source.
- **Command:** used to execute SQL commands and stored procedures.
- **Data Reader:** used to read the data from the data source, and forwards access to the data retrieved from a query.
- **Data Adapter:** used for two tasks: to fill a dataset (a disconnected collection of tables and relationships) with information extracted from a data source, and to apply changes to a data source, according to the modifications made in a dataset.

ADO.NET does not include generic data provider objects. Instead, it includes different data providers specifically designed for different types of data sources. Each data provider has a specific implementation of the connection, command, data reader, and data adapter classes optimized for a specific relational database management system (RDBMS). This application was designed and developed on SQL Server database, and therefore a connection class named SQL Connection was used.

Visual studio. Visual studio is an integrated development environment provided by Microsoft. This is a platform for developers to write, view, edit, and debug code. It can also be used to publish apps written in various programming languages. Visual Studio supports 16 different programming languages, including C, C++, Visual Basics, .NET, JavaScript, XML, HTML, CSS, and many more. Visual Studio uses Microsoft development tools like Windows API, Windows Forms, Windows Silverlight and Windows Store. It also includes code editor and code refactoring. Visual Studio accepts other tools as plug-ins to enhance the functionality, in addition to including support for version controllers like sub version control and Git.

MySQL

MySQL is an open source RDBMS owned by the Oracle Corporation and is known for its reliability, performance, and ease of use. It presently stands as a primary option to support most of the web-based applications, and is also being provided with Oracle MySQL cloud service to help industries build low-cost database systems for their applications. The current application uses MySQL workbench to connect the application to the database where all the keys and data will be stored. MySQL workbench is an additional tool, which enables MySQL with a user-friendly GUI to access, query, and retrieve data from database tables rather than using the

command prompt. It also helps database administrators to visually create, design, model, and manage databases. MySQL server can be installed by downloading the installer provided by the MySQL community. It is open source software and can be customized while installing. The user needs to set a username and password while installing so that the server can be accessed only through their personal credentials. While trying to access user data from SQL, the data administrators try to perform various tasks. SQL tasks can be categorized based following types used to query the data from SQL Server Database:

- Data Definition Language commands (DDL)
- Data Manipulation Language commands (DML)
- Transaction Control Language commands (TCL)
- Data control Language commands (DCL) were used to query the data from SQL Server Database.

Chapter V: Implementation

The main roles in the implementation process of this application are Actuary or Patient-Medical User Login, Emergency provider or Health Care Login, and Database Admin. Each user has their specified roles where every login has different privileges.

Medical user login. The end user usually enrolls into the system by registering themselves. Once the user is registered, all their information is noted. The user's health record is also noted from their personal medical records. Once the user is successfully added to the database, they will be able to connect through their mobile device or from an external source. Once generated, the information is stored in the home gateway. The healthcare personnel can access the information only if the user agrees to share the information.

Healthcare login. This is the login provided to the healthcare personnel from the healthcare provider. This is only given to the users who have been given certain access privileges to the user's data. Also, the user will have to give the key generated by encryption to access any data during the process of an emergency. The data of a medical user that is the patient is only transmitted to the healthcare personnel if the medical user authorizes it.

Body sensor. The sensor will begin a scan if the user enters an emergency situation. Once the sensor notices any kind of emergency, the sensor will notify the Home Gateway about the abnormality. Once the scan has started, the authorized healthcare personnel and the medical user are alerted about the situation for precautionary measures.

The details of the body sensor are communicated with a router to the end user. While this process is being communicated, the data in the system database is applied with the hash function and is stored. Once the details are stored, they are transferred to the emergency caretaker using a

secure transfer function where the message is sent to the authorized individual. The data can only be accessed by the personnel who is given the key generated after applying the hash. This hash is generated by a unique ID in the situation of emergency.

Application Screenshots

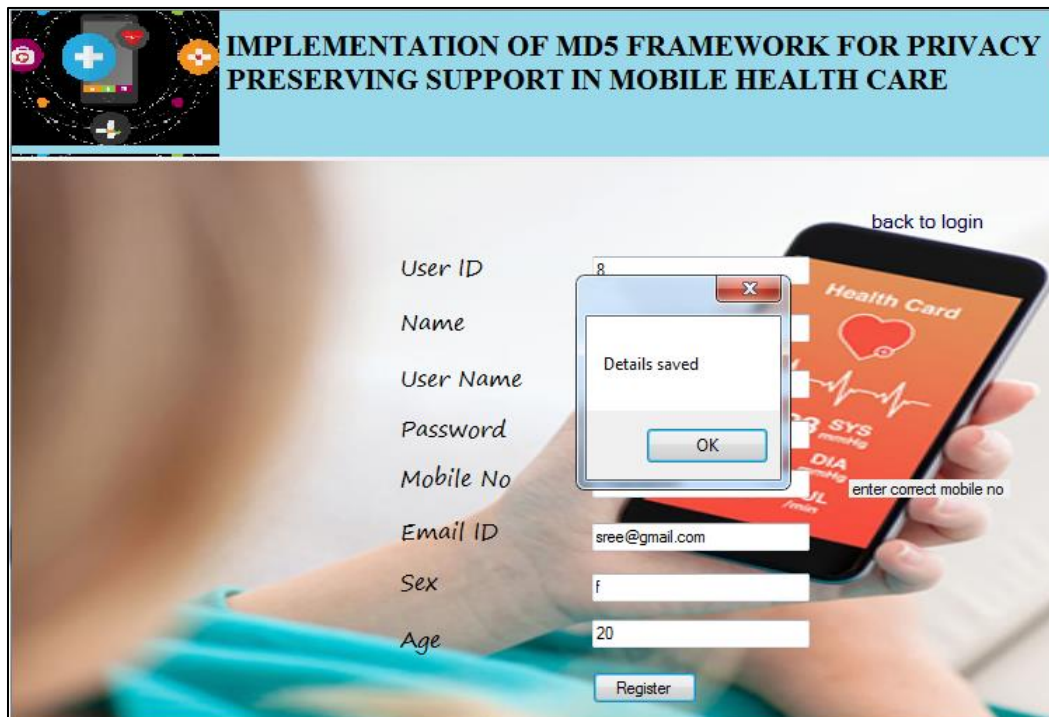


Figure 8: New Medical User Registration

Initially, the user is entered into the system by registering their personal details into the application. All the user (patient) information will be stored into the database from the registration page.

IMPLEMENTATION OF MD5 FRAMEWORK FOR PRIVACY PRESERVING SUPPORT IN MOBILE HEALTH CARE

back to login

User ID

Name

User Name

Password

Mobile No enter correct mobile no

Email ID

Sex

Age

Figure 9: User Details Being Saved

All the required validations on the registration process have been included. The user will have to provide the mandatory fields to finish the registration process. All this is specified in the screenshot below, where valid contact details need to be provided.

IMPLEMENTATION OF MD5 FRAMEWORK FOR PRIVACY PRESERVING SUPPORT IN MOBILE HEALTH CARE

Medical User Login

Username

Password

Figure 10: Validations on User Login

Once, the registration process is completed, the user confirms that the login has been successful by validating the user login. Also, here the user will have a chance to change any of the user settings online after accessing the application with the user credentials.



Figure 11: User Home Page

The above screenshot depicts the home page the user will have access to once they log into the application. The user is able to check the menu options at any point in time, and can also check their physical health details from the application.

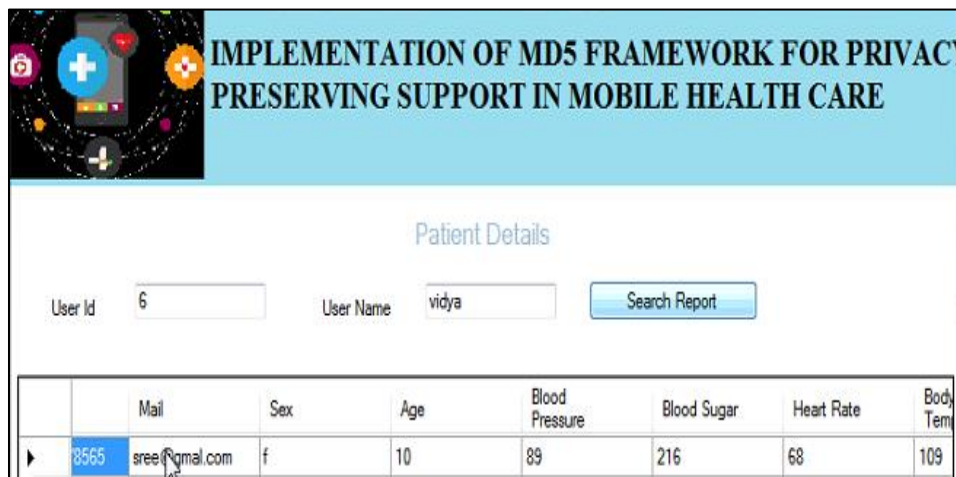


Figure 12: Patient Details on the Patient's Login Details

The patient details tab will allow the user to view physical health details like blood pressure, sugar levels, heart beat rate, body temperatures. This can be customized based on the user's profiles on the health parameters where the user will be required to have a constant monitoring. Fields in the patient's record, like age, will be dynamically calculated based on the information given by the user while registering.

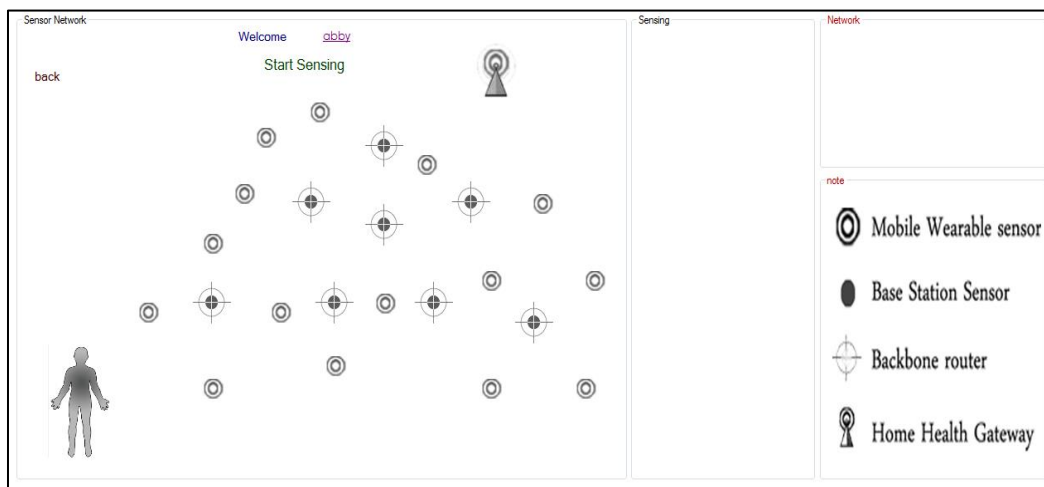


Figure 13: User Information Before Sensing

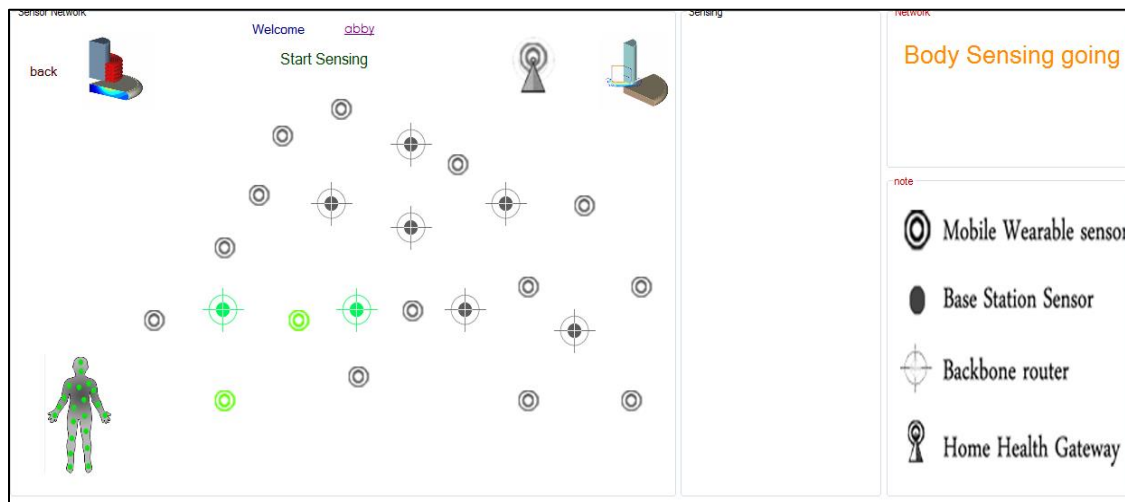


Figure 14: When the Sensor Starts Sensing

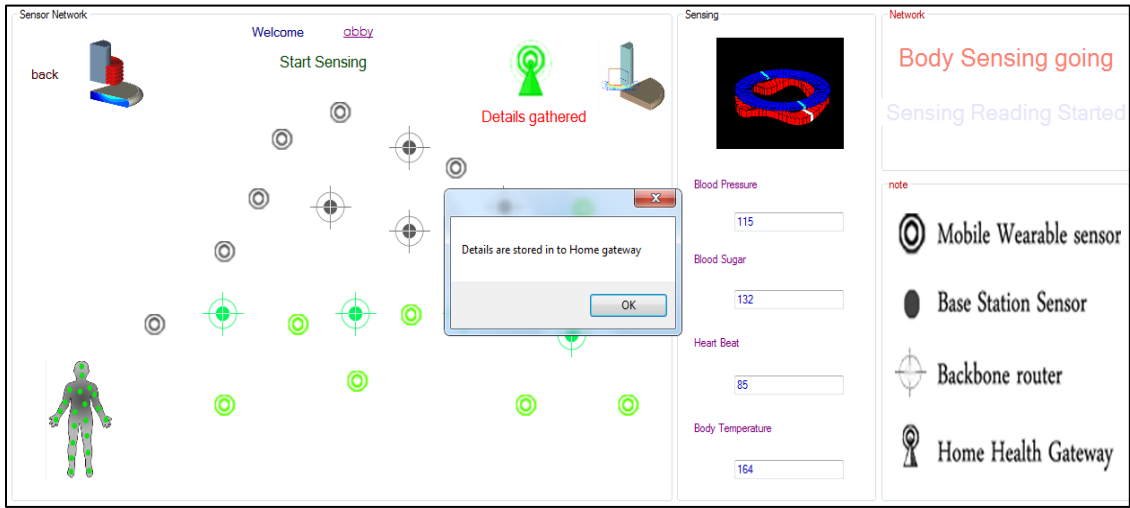


Figure 15: Once the Sensing is Completed and the Information is Gathered

The information of the user will be in a constant state of sensing using the body sensor units. Once the scan is completed, the data is gathered and transmitted to the healthcare unit. The transmission can only be possible if the user is using a wireless mobile phone with access to 4G or higher networks.

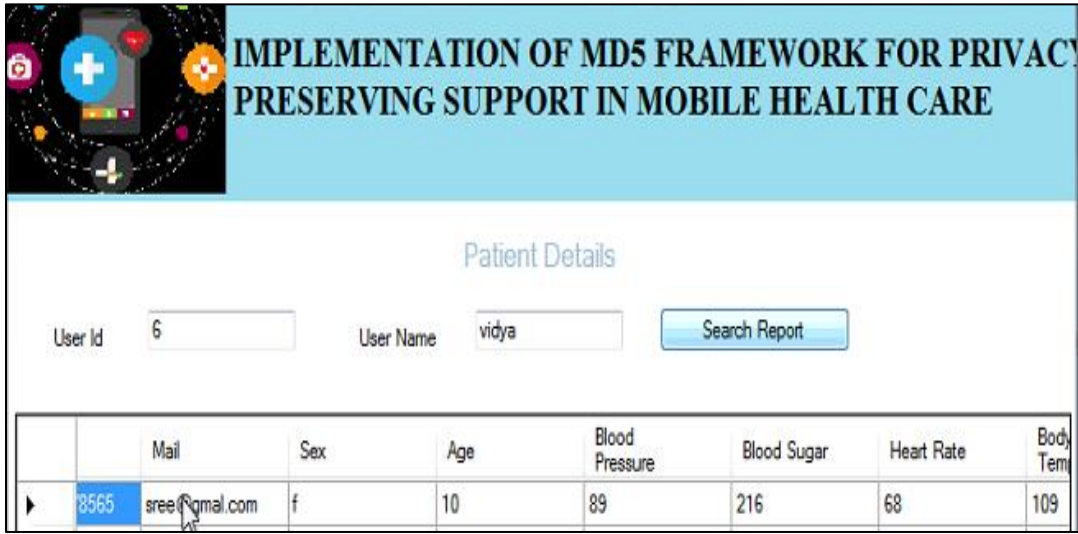


Figure 16: Body Details Being Read by the Sensor

The data will be transmitted to the Health Care unit for constant monitoring and record management of the user. These records will be utilized for treating and diagnosing the user or patient.

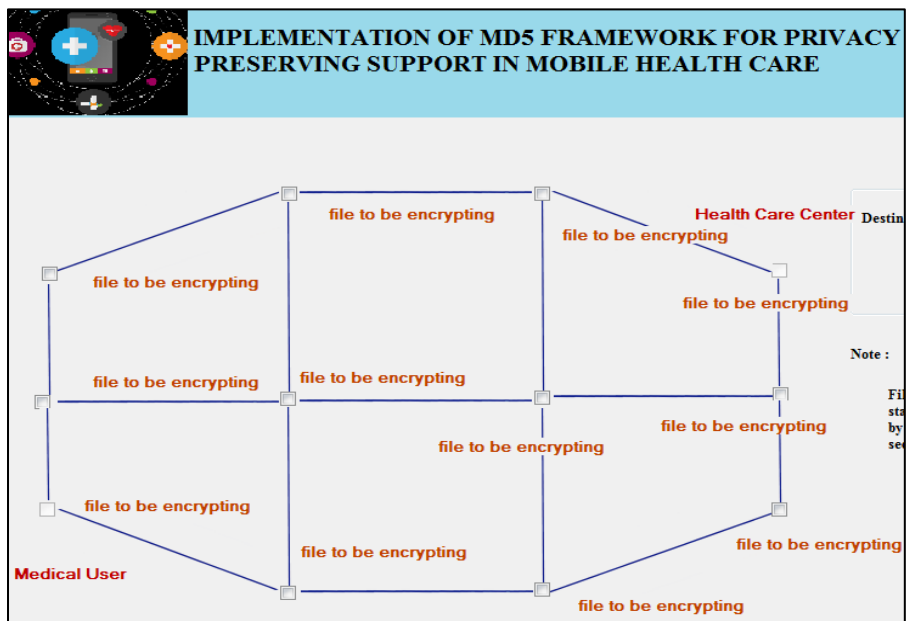


Figure 17: When the Data is Sent for Encryption

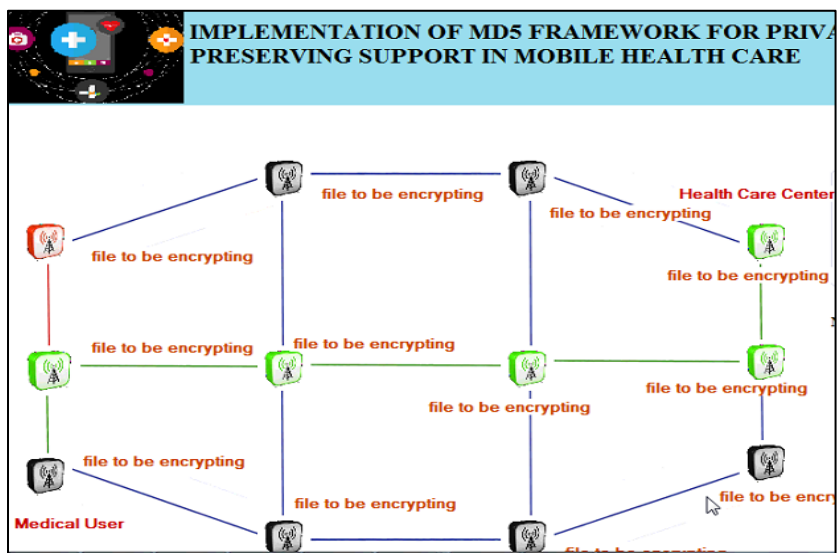


Figure 18: When the Encryption Process is Completed

The data is then stored using the SQL databases in the healthcare unit. The data is sent for encryption to maintain data security and integrity. Only by applying the encryption policies is it allowed and agreed by HIPPA to hold the user's information.

The screenshot shows a web application window titled "Emergency". The main header area is light blue and contains the text "IMPLEMENTATION OF MD5 FRAMEWORK FOR PRIVACY PRESERVING SUPPORT IN MOBILE HEALTH CARE". Below the header, there is a section titled "Patient Details" with two input fields: "User Id" and "User Name". To the right of these fields is a button labeled "Search Report".

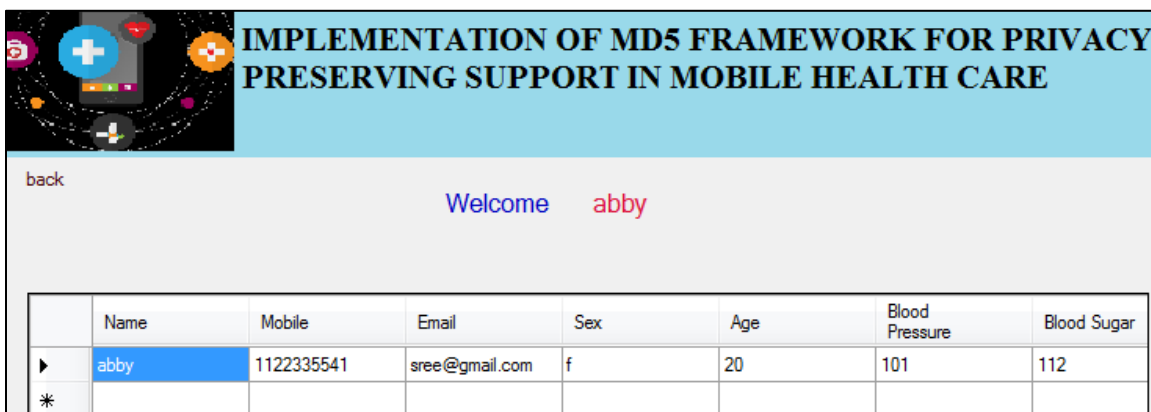
Figure 19: When the Healthcare User Wants to Look at the Details

The screenshot shows a web application window titled "Health care User Login". The main header area is light blue and contains the text "IMPLEMENTATION OF MD5 FRAMEWORK FOR PRIVACY PRESERVING SUPPORT IN MOBILE HEALTH CARE". Below the header, there is a section titled "Health care User Login" with two input fields: "Medical ID" and "Medical Key". To the right of these fields is a button labeled "login". On the left side of the login section, there is an illustration of a stack of papers with checkmarks and a folder labeled "MOBILE HEALTH CARE DATA".

Figure 20: Healthcare User Login

When a healthcare personnel wants to access a user's information to provide medication, the healthcare user should be given rights to access the information. The healthcare user will be given credentials to view this information. All access privileges for healthcare users will be

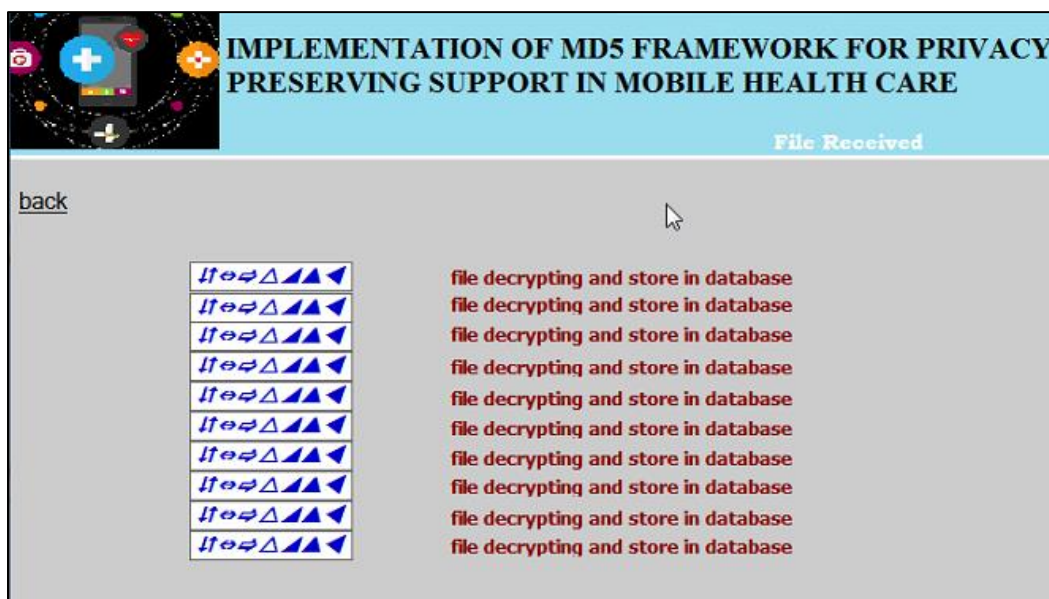
managed by LDAP authentication. Using this authentication, the users are categorized based on various access groups and will be synced into the application via active directory containers.



The screenshot shows a mobile application interface with a header titled "IMPLEMENTATION OF MD5 FRAMEWORK FOR PRIVACY PRESERVING SUPPORT IN MOBILE HEALTH CARE". Below the header, there is a "back" button and a welcome message "Welcome abby". A table displays user data for 'abby'.

	Name	Mobile	Email	Sex	Age	Blood Pressure	Blood Sugar
▶	abby	1122335541	sree@gmail.com	f	20	101	112
*							

Figure 21: Data Accessed for Healthcare User



The screenshot shows a mobile application interface with a header titled "IMPLEMENTATION OF MD5 FRAMEWORK FOR PRIVACY PRESERVING SUPPORT IN MOBILE HEALTH CARE". Below the header, there is a "back" button and a "File Received" notification. A list of encrypted files is displayed, each with a blue icon and a red text description.

File Icon	Description
!@#Δ▲▲▲	file decrypting and store in database
!@#Δ▲▲▲	file decrypting and store in database
!@#Δ▲▲▲	file decrypting and store in database
!@#Δ▲▲▲	file decrypting and store in database
!@#Δ▲▲▲	file decrypting and store in database
!@#Δ▲▲▲	file decrypting and store in database
!@#Δ▲▲▲	file decrypting and store in database
!@#Δ▲▲▲	file decrypting and store in database
!@#Δ▲▲▲	file decrypting and store in database
!@#Δ▲▲▲	file decrypting and store in database

Figure 22: Data Encrypted for Use by the Healthcare Professional

For the healthcare personnel to access any particular patient's information, the user will have to submit a request to obtain decrypted information within a short span of time.



The screenshot shows a web application window titled "Emergency". The header area is light blue and contains the text "IMPLEMENTATION OF MD5 FRAMEWORK FOR PRIVACY PRESERVING SUPPORT IN MOBILE HEALTH CARE" in bold black letters. To the left of the header is a graphic with a blue cross and other medical symbols. Below the header, the text "Patient Details" is centered. The main content area contains two input fields: "User Id" and "User Name", each with a text box. To the right of the "User Name" field is a button labeled "Search Report".

Figure 23: Emergency Lookup of Information for Authorized Users

In emergency situations, the healthcare user will be provided with access rights to view the data. These instances usually occur when the healthcare user is not already part of the authorization group. These credentials will be communicated by the internal mailing systems.

Chapter VI: System Design

Developers or software engineers initially design the requirements while in the process of development. There are members from the business perspective also involved in this process. The whole sit together to transform the entire design into an architecture for the project. This will be the blueprint for all the events and activities required during the project.

Once the team has the set of requirements in hand, developers may begin the process of creating software. Based on the organization requirement and logistics, each project will be started in a specific development model. The classic approach to software development is a methodology known as waterfall approach. This model was developed by Winston Royce during the 1970s approach to software development as a linear process. It follows a rigid series of steps that begin with developing system requirements, moving on to developing software requirements, then producing a preliminary design from those requirements, which is used as a basis for detailed design. Once that design is completed, developers begin the coding and debugging process where they create the software.

When they finish coding, the software is tested rigorously. If the code meets all acceptance criteria, it is moved to operations and maintenance mode. This approach allows for movement back to a previous step, but only by one phase at a time. For instance, if there are any tests that have not been passed during the testing phase, the total project will be moved back to the coding phase. This method has largely been used in the software industry but has created some issues where too much time has been consumed moving back and forth in the developmental phases.

Recently, more organizations around the world have started using the Agile approach for software development. This approach values moving rapidly to the creation of the software and is quite popular. The creators of the Agile approach authored a document called the *Agile Manifesto* that discusses their approach in detail [19], [20]. There are four core concepts to Agile: valuing individuals and interactions over processes and tools, working software over comprehensive documentation, satisfaction over contract negotiation, and responding to change over a fixed plan. There are twelve principles agile developers follow, with the highest priority being the satisfaction of customers through early and continuous delivery of valuable software [19]. They welcome changing requirements even late in development.

This methodology helps deliver information efficiently and effectively to a development team through a conversation in person. Frequently, the whole team decides jointly on where the developers will deliver working software, from couple of weeks to a couple of month's cycle with a preference to a shorter timescale. In an organization using Agile, people from business and development must work together daily throughout the project, with working software being the primary focus for progress. During the entire process, both development and testing work hand in hand for the end working software.

In the Agile process, continuous attention to technical excellence and good design enhances agility. Simplicity is essential. In this methodology, constant monitoring to the end users' needs is prioritized, with a higher inclination towards customer's requirements [20]. Every organization has its own strategy and business needs on which the development methodology is chosen.

UML Diagrams

In order to visualize and document the flow or artifacts of systems, a language called UML is used. UML stands for Unified Modelling Language [21]. Pictorial representation of a flow of steps helps in acquiring knowledge about the process more easily. UML has been developed in order to organize the traditional developed concepts, such as object-oriented concepts.

Using UML diagrams any complex level project can be explained to the developers. The major forms of UML diagrams can be categorized into two types:

1. Static representations, such as class diagrams
2. Dynamic representations, such as activity diagrams, use case diagrams, and sequence diagrams.

Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a sort of behavioral graph characterized by and made from a Use-case investigation. Figure 24 depicts the use case diagram for the project. It explains the role of the medical user or the patient in the project. The user first registers to the system and fills in the required details. The user will have to login to verify the PHI. Once the user is on the site, they can check the PHI using the sensor details.

We also have a use case diagram for the admin in the project (see Figure 25). This demonstrates where the admin can access information securely during emergency situations.

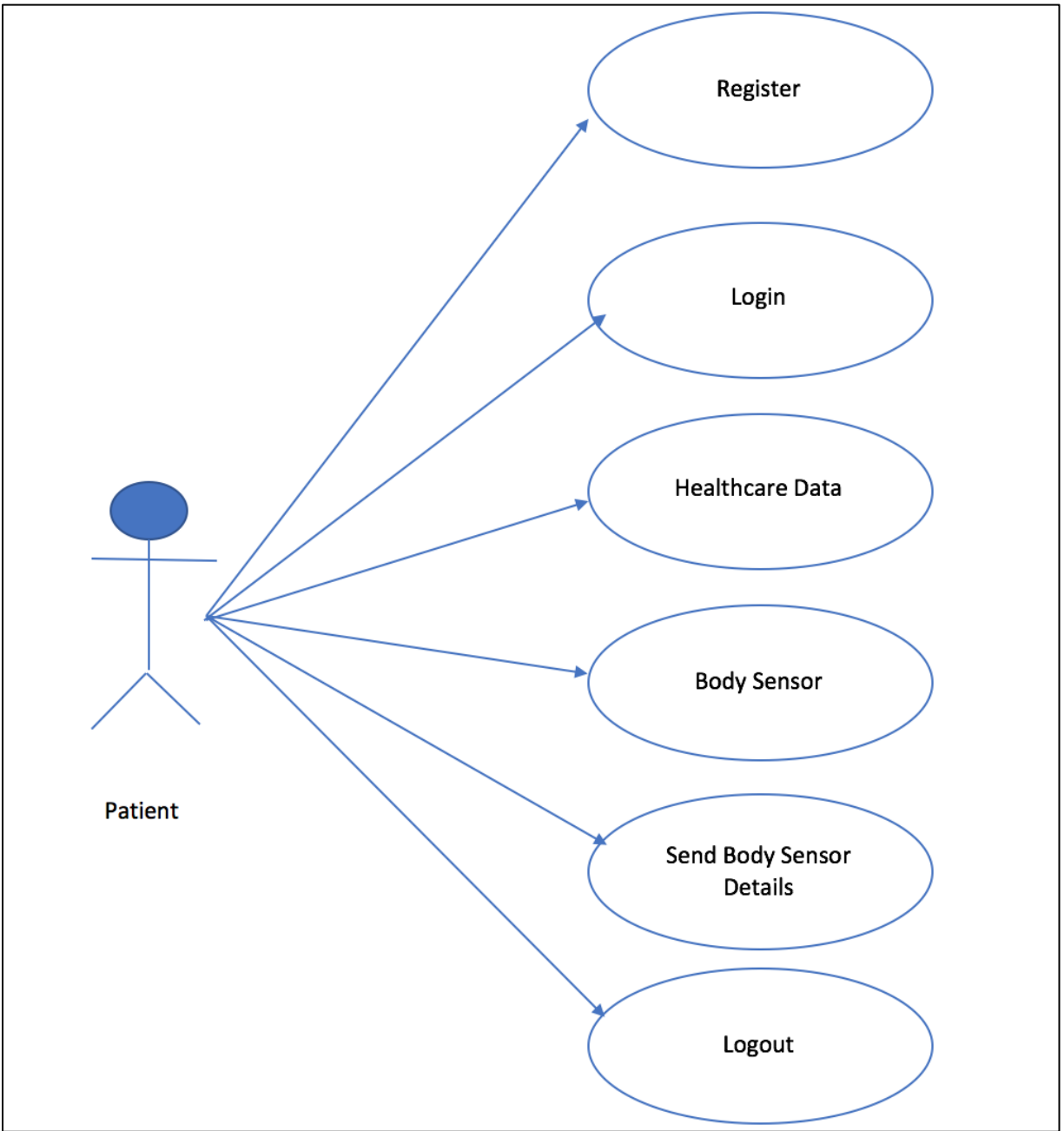


Figure 24: Use Case for Patient

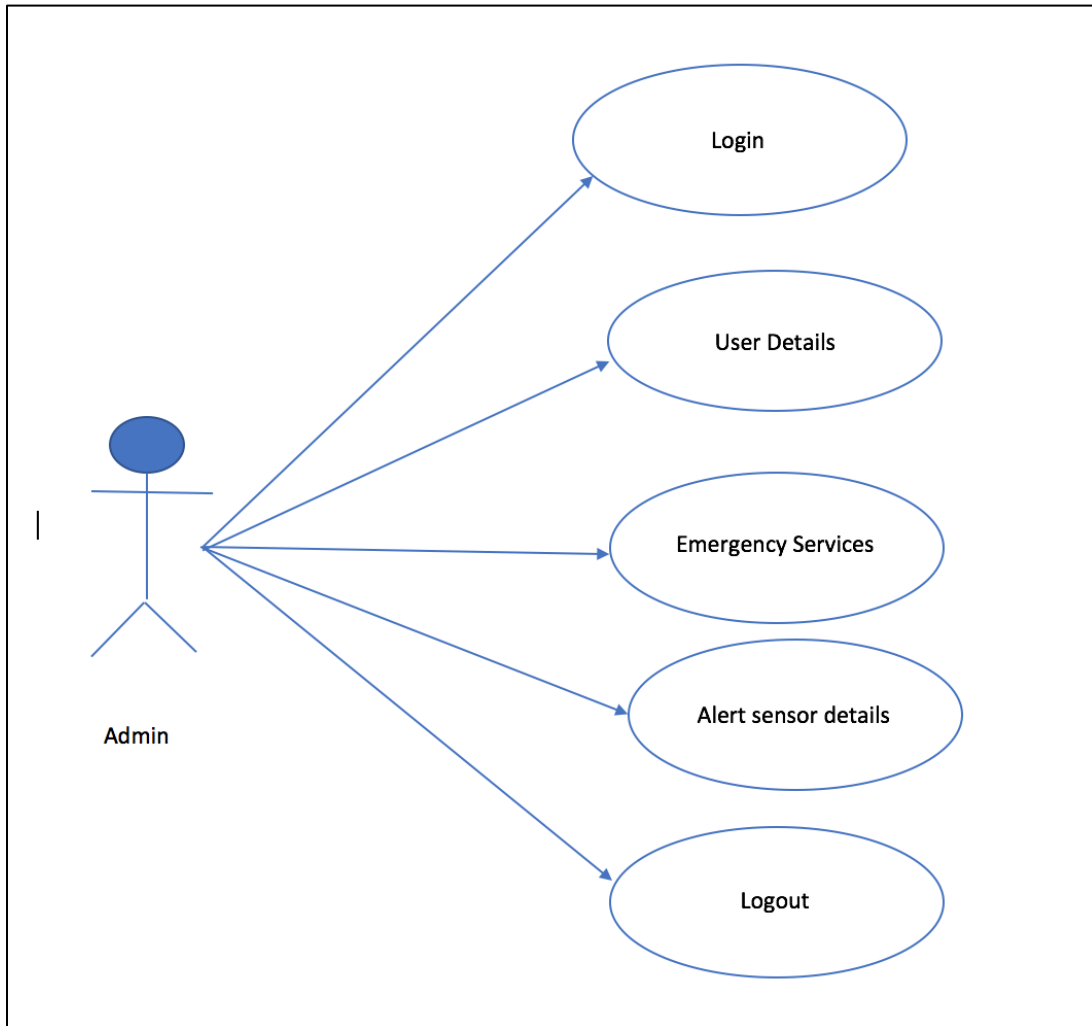






Figure 25: Use Case for Admin

Data Flow Diagram

Data flow diagram is largely used for gathering high quality requirements, and is a graphical representation of the data flow in the application. This helps in gaining a clear picture of the flow of the application end to end. This also helps developers and testers to gain a broader view of the project. Data flow diagrams have been helpful to maintain higher customer satisfaction.

Basic components of the data flow diagram are:

1. External entities represented by: 
2. Data flow represented by: 
3. Data process is represented by: 
4. Multiple actions are represented by: 

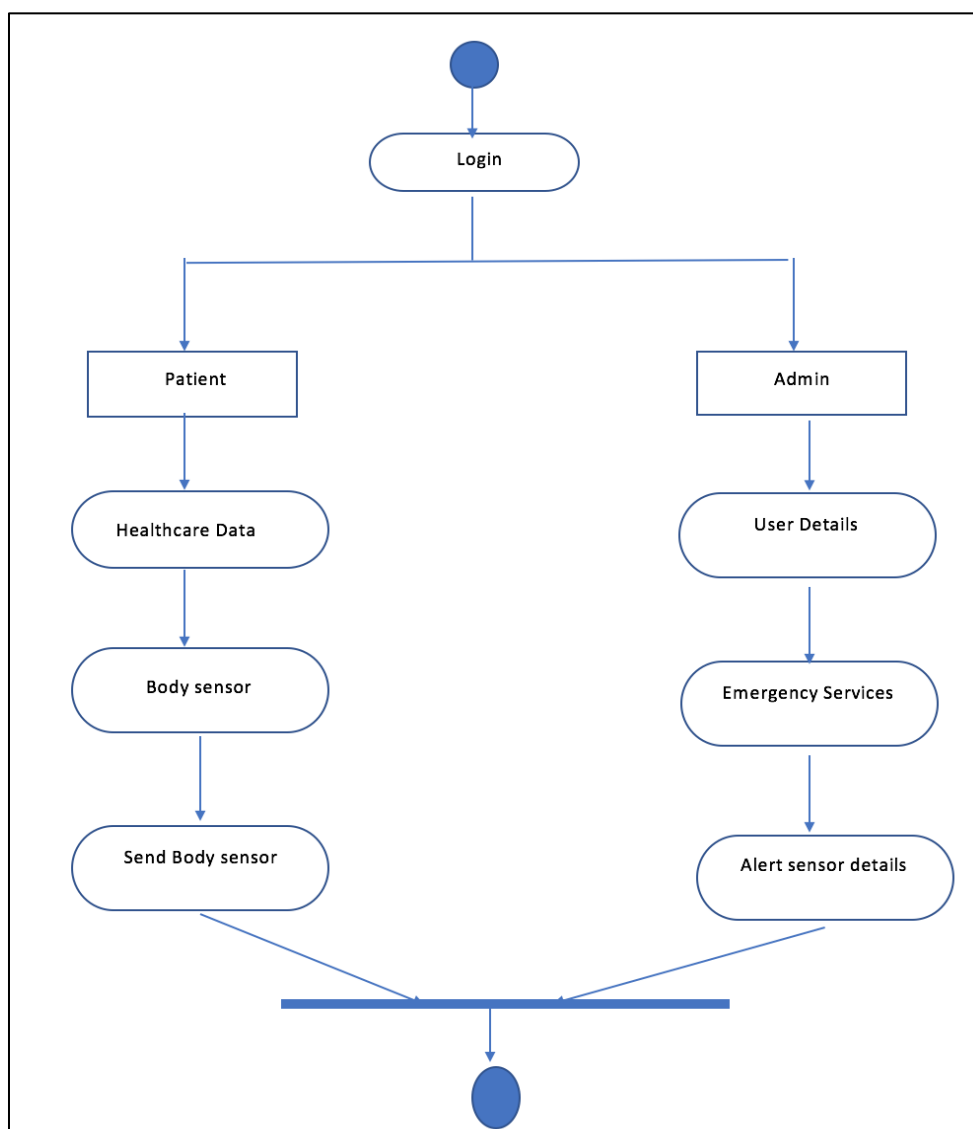


Figure 26: User Flow Chart

Chapter VII: Testing

The purpose of testing is to identify errors during the process of checking the application to execute as per design. The tests are designed to check the functionality of components in a finished product.

Testing is a part of the development process. Both developers and testers together work in the development process and test the functionality of the project to ensure that all requirements are met. Apart from the normal manual testing, however, a majority of the methods work better during development rather than in final testing of the application. Some of the .NET Integrated Development Environments (IDEs) available have some form of debugging and testing tools built into them. Visual Studio, for example, is such an IDE used for the development of web applications with integrated testing and debugging tools.

The testing of this program will consist of manual testing. The application will be tested for its level of accessibility to ensure all the end user's requirements are met. User testing was done to comprehend the application's usability and accessibility, in addition to obtain a better understanding of how users will use the application and where they may find problems rather than relying on the development and manual testing. This testing will include getting a host of different users to complete simple tasks on the application.

Testing also includes applying certain loads and verifying that the time to process the data is met as per the requirement. In software development, there are many different approaches to testing an application. Various types include acceptance testing, black box testing, white box testing, compatibility testing, regression testing, and performance testing. One of the most popular of these is unit testing, which plays an important role in test-driven development.

In testing all the objectives, the following tests have been performed:

1. Tested the UI login screens for successful login.
2. Tested for registering of the user successfully into the system.
3. Tested the user entering the basic health information.
4. Tested for user health information sensing using the sensor into the application.
5. Tested for the admin to login and log out.
6. Tested for data transmission to the health center in a secured manner.
7. Tested the specific functions performing the sharing of information to healthcare personnel in case of emergency.

Chapter VIII: Conclusion

In the proposed paper, the ideas of various secure data preserving techniques like SPOC are discussed with details of their favors and inconveniences. A detailed schedule of requirement gathering, layout of product requirement, software outline has been made in constructing the application. The correlations involved have been implemented into the application and all apparent issues have been fixed. The implemented application has been thoroughly tested for defects. The current scope of study is confined only to one single server for a single database.

Future scope. The future scope for this study could be trying to implement a similar architecture across cloud-based environments and more accessible across various domains. There has been limitation with different app types, which can be achieved in the future scope.

References

- [1] A. Pereira, F. Marins, B. Rodriguez, and F. Portela, "Improving quality of medical service with mobile health software," in *The 5th International Conference on Current and Future Trends and Communication Technologies in Healthcare*, 2015.
- [2] R. Lu, X. Lin, and X. Shen, "SPOC: A secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency," *Parallel and Distributed Systems, IEEE Transactions on* 24.3, 2013, pp. 614-624.
- [3] G. Vaibhav and Prof. A. Bakshi, "A review paper on secure support for mobile healthcare using message digest," *International Journal of Research in Science & Engineering*.
- [4] R. Rivest, "The MD4 message digest algorithm," RFC 1320, MIT and RSA Data Security, Inc., April 1992.
- [5] Y. Ren, R. W. Yonglin, N. Pazzi, and A. Boukerche, "Monitoring patients via a secure and mobile healthcare system," *Wireless Communications, IEEE*, vol. 17, no. 1, 2010, pp. 59-65.
- [6] P. S. Rani and S. Ramakrishna, "Mobile-healthcare emergency based on opportunistic computing framework incorporating secure and privacy preserving mechanism," *Communications in Computer and Information Security in Computing Communications*, 2013.
- [7] A. Passarella, et al., "Performance evaluation of service execution in opportunistic computing," in *Proceedings of the 13th ACM international conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, ACM, 2010.
- [8] M. Banikazemi, V. M. Mohammad, and D. K. Panda, "Efficient collective communication on heterogeneous networks of workstations," 1998. *Proceedings of International Conference on Parallel Processing*, IEEE, 1998.
- [9] N. Yankelovich, et al., "Meeting central: making distributed meetings more effective," *Proceedings of the 2004 ACM conference on Computer supported cooperative Work*, ACM, 2004.
- [10] C. Doukas, T. Pliakas, and I. Maglogiannis, "Mobile healthcare information management utilizing Cloud Computing and Android OS," *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. IEEE, 2010.
- [11] S. Dähne, et al., "Space: A novel framework for linking up the amplitude of neuronal oscillations to behaviorally relevant parameters," *NeuroImage*, vol. 86, pp. 111-122, 2014.

- [12] J.-H. Wu, S.-C. Wang, and L.-M. Lin, "Mobile computing acceptance factors in the healthcare industry: A structural equation model," *International Journal of Medical*, vol. 76, no. 1, 2007, pp. 66-77.
- [13] A. K. Agarwa and W. Wang, "Measuring performance impact of security protocols in wireless local area networks," in *2nd International Conference on Broadband Networks, IEEE*, October 2005, pp. 581-590.
- [14] "ADO.NET implementation," from *Microsoft library*, [https://msdn.microsoft.com/enus/library/ms178371\(v=vs.100\).aspx](https://msdn.microsoft.com/enus/library/ms178371(v=vs.100).aspx).
- [15] "MOVEIt Transfer API, ipswitch development, ts/DS-MI-Transfer-API," <https://www.ipswitch.com/Ipswitch/media/Ipswitch/Documents/Resources/Data%20Sheet>
- [16] "AES algorithm diagrammatic representation" retrieved from online, https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.html.
- [17] J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Information Security*. Springer Berlin Heidelberg, 2009, pp. 347-362.
- [18] ".NET implementation and development from ASP developer online source," <https://www.developer.com/net/asp/top-10-features-in-asp.NET-4.5.html>.
- [19] "AgileManifesto" is online, <http://agilemanifesto.org/>.
- [20] V. Szalvay, "An introduction to Agile Software Development," Danube Technologies, Inc., 2004.
- [21] Tutorials Point. (n.d.). UML Tutorial. Retrieved May, 2017 from online "http://www.tutorialspoint.com/uml/"

Appendix

MoveIT Code Snippet-

```

Const FOR_READING = 1
Const FOR_WRITING = 2
Const FOR_APPENDING = 8

Do While 1=1

    Dim MD5Filename, tmp
    Dim fso,fo,foname, xDB
    Dim RetMsg, RetCode
    Dim TaskID, MySQL, MyConnStr, LastLogStamp, AccessSet, bOK
    Dim Title, Body, CRLF, TempCopy
    CRLF = chr(10) & chr(13)
    ' MD5Filename = MIMacro(trim(MIGetTaskParam("MD5Sum_Filename")))
    ' IF MIGetDebugLevel() >= 60 THEN MILogMsg "MD5Filename=" & MD5Filename
    RetCode = 0
    RetMsg = ""
    TaskID = MIMacro("[TaskID]")

    ' MyConnStr is the ODBC connection string of the SOURCE DATABASE.
    MyConnStr = MIGetCentralInfo("ConnStr")

    if MIGetTaskInfo("NSources") = "0" then
        RetMsg = "ERROR A 'MD5Unique' process cannot be the first step in a
task!"
        RetCode = 581
        Exit Do
    end if

    if MIGetTaskInfo("ProcessIsPerFile") = "True" then
        tmp = CalcMD5Checksum()
        IF MIGetDebugLevel() >= 50 THEN MILogMsg "MD5 hash of " &
MIGetOriginalFilename & " is " & tmp

        IF tmp="" then
            RetMsg = "ERROR Could not execute MD5 command on this file!!!"
            RetCode = 589
            Exit Do
        END IF

        ' Check to see if this task has processed a file with this MD5 already
        MySQL = "SELECT LogStamp FROM Stats WHERE TaskID='" & TaskID & "' AND
Action='process' " & _
        " AND Message='MD5=' & tmp & "' ORDER BY LogStamp LIMIT 1; "
        IF MIGetDebugLevel() >= 70 THEN MILogMsg "Lookup Query: " & MySQL
        IF MIGetDebugLevel() >= 70 THEN MILogMsg "Starting Lookup..."
    
```

```

' Try to open database
Set xDB = New siLockWrap
If NOT xDB.OpenConn(MyConnStr) Then
    RetMsg = "Failed to open source database"
    RetCode = 100
    Exit Do
End If
' Execute query
xDB.DoReadQuery MySQL, AccessSet
LastLogStamp = "NO MATCH"
If Not AccessSet Is Nothing Then
    If Not AccessSet.EOF Then
        LastLogStamp = AccessSet("LogStamp")
    End If
End If
IF MIGetDebugLevel() >= 70 THEN MILogMsg "Lookup Finished"

if LastLogStamp = "NO MATCH" then
    ' Just write the MD5 into the file activity record
    RetCode = 0
    RetMsg = "MD5=" & tmp
else
    ' Ooops - found match - IGNORE THIS FILE and complain
    TempCopy = "C:\temp\md5unique_" & MIMacro("[TaskID]") & "_" & tmp & "_"
    & MIMacro("[Rnd:8]") & ".dup"
    RetMsg = "IGNORED file - matched previous file examined on (" &
CStr(LastLogStamp) & ") - MD5=" & tmp & " - COPIED_TO=" & TempCopy
    set fso = CreateObject("Scripting.FileSystemObject")
    fso.CopyFile fso.GetFile(MICacheFilename).ShortPath, TempCopy
    MIIgnoreThisFile True
    RetCode = 5000
    Title = "MOVEit Central Detected Dup in (" & MIMacro("[TaskName]") &
")"
    Body = "MOVEit Central detected a duplicate file posting in the '" &
MIMacro("[TaskName]") & _
        "' task at " & MIMacro("[yyyy]-[mm]-[dd] [hh]:[tt]:[ss]") & "." &
CRLF & CRLF & _
        "This file matched another file posted at " & CStr(LastLogStamp) & "
with an MD5 hash of " & tmp & "." & CRLF & CRLF & _
        "This file was ignored and not sent to any destinations. However a
copy of this file was saved as '" & TempCopy & "' for further analysis."
    bOK = MIEmailOperator(Title, Body, 2)
    if NOT bOK then
        IF MIGetDebugLevel() >= 20 THEN MILogMsg "Could NOT notify MOVEit
Central operator (using MIEmailOperator)"
        end if
    end if
else
    ' Cannot run PER TASK

```

```

        RetMsg = "ERROR You MUST run this script PER-FILE not per-task (or
'once after all downloads')!!!"
        RetCode = 584
    end if

    Exit Do
Loop

MISetErrorDescription RetMsg
MISetErrorCode RetCode

'-----
' Function: CalcMD5Checksum
'
' Calculates the MD5 Checksum for the file by calling md5sum.exe and
' parsing the output of this utility.
'-----
Function CalcMD5Checksum
    Dim inputFilename, outputFilename
    Dim cmd
    Dim ret
    Dim fso, fi, buf, i

    Set fso = CreateObject("Scripting.FileSystemObject")

    ' The input file is the one we want to calculate the hash on
    inputFilename = fso.GetFile(MICacheFilename).ShortPath

    ' The output file is a temp file we can redirect the output to
    outputFilename = MINewCacheFilename

    ' Build the command to run
    cmd = "md5sum.exe " & inputFilename & " > "" & outputFilename & """"

    ' Run the command
    If MIGetDebugLevel() >= 60 then MILogMsg "Running command '" & cmd & "'"
    ret = MIRunCommand(cmd)

    If ret <> 0 Then
        If MIGetDebugLevel() >= 20 then MILogMsg "ERROR #" & ret & " while
running command '" & cmd & "'"
        MISetErrorCode 1000 + ret
        MISetErrorDescription "md5sum.exe returned error " & ret
        CalcMD5Checksum = ""
        Exit Function
    End If

    ' Read the output file to get the checksum
    Set fi = fso.OpenTextFile(outputFilename, FOR_READING)

```

```

buf = fi.ReadLine
fi.Close
Set fi = Nothing
Set fso = Nothing

' Log the output from md5sum.exe
MILogMsg buf

' Find where the 2 spaces followed by the input filename begins
i = Instr(buf, " " & inputFilename)

' Return everything up to the space
CalcMD5Checksum = Left(buf, i-1)
End Function

'=== class siLockWrap =====
' This class uses ADO to access a database.
Class siLockWrap
    Dim MyADODConnect ' Connection used by all database methods
    Dim DSN            ' Cached DSN from OpenConn (for debugging?)
    Dim XPEDDict      ' Scratch dictionary object
    Dim DateDelim     ' Character used to delimit dates in SQL
    Dim ObjName       ' Name of class, used in debug log
    Dim TableName     ' Name of data table, used to return start stamp

    Dim adUseClient
    Dim adOpenStatic
    Dim adLockReadOnly

    Private Sub Class_Initialize
        ObjName = "siLockWrap."
        adUseClient = 3
        adOpenStatic = 3
        adLockReadOnly = 1
    End Sub

    Private Sub Class_Terminate
        Set MyADODConnect = Nothing
    End Sub

    '--- Sub OpenConn -----
    ' Open the cached database connection.
    ' Call this function soon after Dim-ing the object.
    '
    ' Entry: ConnStr is the database connection string.
    '
    ' Exit: Returns True if successful, False otherwise.
    Public Function OpenConn(ConnStr)
        On Error Resume Next
        OpenConn = False
        Set MyADODConnect = CreateObject("ADODB.Connection")

```

```

        MyADODConnect.Open ConnStr
        If Err.Number = 0 Then
            OpenConn = True
        End If
    End Function

'--- Function DoReadQuery -----
' Do a SQL query and return a recordset.
'
' Entry:  MySQL   is a SQL query.
'
' Exit:   Returns error code (SIL_SEV_SUCCESS is success)
'        RecSet  is a recordset, If we could create one.
Public Function DoReadQuery(MySQL, RecSet)
    Dim errcode, FuncName
    FuncName = ObjName & "DoReadQuery:"
    Set RecSet = CreateObject("ADODB.Recordset")
    ' Disconnected recordsets don't work with GROUP and WHERE,
    ' as with MakePageFolders
    RecSet.CursorLocation = adUseClient ' For disconnected recordset
    RecSet.Open MySQL, MyADODConnect, adOpenStatic, adLockReadOnly
    If Err.Number <> 0 Then
        Exit Function
    End If
    If Err.Number <> 0 Then
        Exit Function
    End If
    If RecSet.EOF Then
        Exit Function
    End If
    Set RecSet.ActiveConnection = Nothing ' Disconnects the RS
    DoReadQuery = errcode
End Function

'--- Function DoUpdateQuery -----
' Perform a SQL update query.
'
' Entry:   MySQL is an UPDATE SQL statement.
'
' Exit:    Returns SIL_SEV_*** error code.
Public Function DoUpdateQuery(MySQL)
    Dim errcode, FuncName
    On Error Resume Next
    errcode = False
    FuncName = ObjName & "DoUpdateQuery: "
    Dim MyADODRecSet
    Set MyADODRecSet = CreateObject("ADODB.Recordset" ' Execute the query
    MyADODRecSet.Open MySQL, MyADODConnect
    ' If Err.Number <> 0 Then
    If Err.Number = 0 Then
        errcode = True
    End If
    Set MyADODRecSet = Nothing

```

```
    DoUpdateQuery = errcode  
End Function
```
