**APPENDIX 1**

**TELEMEDICINE: EMPOWERING EMERGENCY DERMATOLOGY CARE WITH A CONSULT MOBILE APPLICATION**

**A PROJECT REPORT**

***Submitted By***

**VARSHINI D**

**MOHANA PRIYA S**

**AKASH R**

**SHARMILA T**

***in partial fulfillment for the award of the degree***

***of***

***BACHELOR OF ENGINEERING***

***IN***

COMPUTER SCIENCE AND ENGINEERING

**KARPAGA VINAYAGA COLLEGE OF ENGINEERING AND TECHNOLOGY**

**ANNA UNIVERSITY:: CHENNAI 600 025**

MAY 2025

**APPENDIX 2**

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **“TELEMEDICINE : EMPOWERING EMERGENCY DERMATOLOGY CARE WITH A CONSULT MOBILE APPLICATION”** is the Bonafide work of **“VARSHINI. V, MOHANA PRIYA. S, SHARMILA. T, AKASH. R”** who carried out the project work under my supervision.

**SIGNATURE**

(Dr. Jayapradha)

**HEAD OF THE DEPARTMENT**

Computer Science and Engineering

GST Road, Chinna Kolambakkam,

Palayanoor PO, Madurantagam Taluk,

Chengalpattu, Tamil Nadu 603 308.

**SIGNATURE**

(xxxxxxxxxxx)

**SUPERVISOR**

Assistant Professor

Computer Science and Engineering

GST Road, Chinna Kolambakkam,

Palayanoor PO, Madurantagam Taluk,

Chengalpattu, Tamil Nadu 603 308.

**TELEMEDICINE: EMPOWERING EMERGENCY DERMATOLOGY CARE WITH A CONSULT MOBILE APPLICATION**

**Abstract**:

In these task, the modules of client and specialist in these application, the client register and login in to the application. the client login with legitimate verification, in these application the client catch picture of skin and transfer to specialist, the specialist register and login in to the application, in client side the specialist list view and the client select the specialist to send pictures with his information. The specialist get and add the aftereffect of his sickness. The specialist send the installment add up to the client, the client once pay the sum to the specialist the specialist confirm and send the remedy report in these application. at long last the client get the record to download

**APPENDIX 3**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVATIONS** | **iii**  **vi**  **vii**  **x** |
| **1.** | **INTRODUCTION**   * 1. INTRODUCTION   2. EXISTING SYSTEM   3. PROPOSED SYSETM   1.3.1 OBJECTIVES  1.3.2 SCOPE   * 1. LITERATURE SURVEY   2. MERITS   3. DEMERITS | **11**  11  12  12  12  13  14  16  17 |
| **2.** | **PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHEDOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.2.3 MODULE DIAGRAM  2.3 TECHNIQUE OR ALGORITHM | **18**  18  18  18  18  20  24 |
| **3.** | **REQUIREMENTS**  3.1 GENERAL  3.2 HARDWARE REQUIREMENTS  3.3 SOFTWARE REQUIREMENTS | **25**  25  25  25 |
| **4.** | **SYSTEM DESIGN**  4.1 GENERAL  4.1.1 USECASE DIAGRAM  4.1.2 STATE DIAGRAM  4.1.3 ACTIVITY DIAGRAM  4.1.4 CLASS DIAGRAM  4.1.5 SEQUENCE DIAGRAM  4.1.6 DATAFLOW DIAGRAM  4.1.7 ER – DIAGRAM  4.1.8 SYSTEM ARCHITECTURE | **26**  26  26  27  28  29  30  31  32  33 |
| **5.** | **SOFTWARE SPECIFICATION**  5.1 GENERAL  5.2 JAVA  5.2.1 THE JAVA FRAMEWORK  5.2.2 OBJECTIVES OF JAVA  5.2.3 JAVA SERVER PAGES  5.2.4 EVOLUTION OF ANDROID APPLICATION  5.2.5 BENEFITS OF JSP  5.3 SERVLETS  5.4 JAVA SERVLETS  5.5 CONCLUSION | **35**  35  35  35  36  37  38  40  41  41  42 |
| **6.** | **IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION | **43**  43  43 |
| **7.** | **SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS | **52**  52  52 |
| **8.** | **SOFTWARE TESTING**  8.1 FEASIBILITY STUDY  8.1.1 ECONOMICAL FEASIBILITY  8.1.2 TECHNICAL FEASIBILITY  8.1.3 OPERATIONAL FEASIBILITY  8.2 SYSTEM TESTING  8.2.1 VARIOUS LEVEL OF TESTING  8.2.1.1 WHITE BOX TESTING  8.2.1.2 BLACK BOX TESTING  8.2.1.3 UNIT TESTING  8.2.1.4 FUNCTIONAL TESTING  8.2.1.5 PERFORMANCE TESTING  8.2.1.6 INTEGRATION TESTING  8.2.1.7 VALIDATION TESTING  8.2.1.8 SYSTEM TESTING  8.2.1.9 OUTPUT TESTING  8.2.1.10 USER ACCEPTANCE TESTING | **54**  54  54  55  55  55  55  56  56  57  57  58  58  59  60  60  61 |
| **9.** | **APPLICATION AND FUTURE ENHANCEMENT**  9.1 GENERAL  9.2 APPLICATIONS  9.3 FUTURE ENHANCEMENTS | **62**  62  62  63 |
| **10.** | **CONCLUSION**  10.1 CONCLUSION  10.2 REFERNCES | **64**  64  64 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.2 | Activity Diagram for Anonymous database Management |  |
| 4.3 | Use case Diagram for Anonymous database Management |  |
| 4.4 | Data flow diagram for Anonymous database Management |  |
| 4.5 | Sequence diagram for Anonymous database Management |  |
| 4.6 | Collaboration diagram for Anonymous database Management. |  |
| 4.7 | Class diagram for Anonymous database Management. |  |
| 4.8 | Architecture Diagram for Anonymous database Management |  |
| 4.9 | Object Diagram |  |
| 4.10 | State Diagram |  |
| 4.11 | Component Diagram |  |
| 4.12 | E-R Diagram |  |

**LIST OF SYMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *+ public*  *-private*  *# protected*  *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation* | Represents a collection of similar entities grouped together. |
| 2. | Association | Class B  Class A  name | Associations represent static relationships between classes. Roles represent the way the two classes see each other. |
| 3. | Actor |  | It aggregates several classes into a single class. |
| 4. | Aggregation | Class B  Class A  Class B  Class A | Interaction between the system and external environment. |
| 5. | Relation  (uses) | **uses** | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is like another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State State | State of the process. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final State |  | Final state of the object |
| 11. | Control Flow |  | Represents various control flow between the states. |
| 12. | Decision Box |  | Represents decision making process from a constraint. |
| 13. | Use case |  | Interaction between the system and external environment. |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process / Sate |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard, sensors, etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message |  | Represents the message exchanged. |

**LIST OF ABBREVATIONS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | DB | Data Base |
| 2. | SMC | Secure Multiparty Computation |
| 3. | SC | Service Centre |
| 4. | DBC | Data Base Confidentiality |
| 5. | JVM | Java Virtual Machine |
| 6. | JSP | Java Server Page |

**CHAPTER 1:**

**INTRODUCTION**

* 1. **INTRODUCTION:**

In these endeavor, the modules of client and master in these application, the client register and login in to the application. the client login with legitimate check, in these application the client get picture of skin and move to prepared proficient, the master register and login in to the application, in client side the master overview view and the client select the master to send pictures with his information. The master get and add the possible result of his burden. The master send the piece add up to the client, the client once pay the total to the master the master assert and send the fix report in these application. At last the client get the record to download.

* 1. **EXISTING SYSTEM:**

Although many soft robotic skins have been introduced, their use has been hindered due to practical limitations, such as difficulties in manufacturing, poor accessibility, and cost inefficiency. To solve this, we present a low-cost, easy-to-build soft robotic skin utilizing air-pressure sensors and 3D-printed pads.

**Techniques**:

PHRI METHOD

* 1. **PROPOSED SYSTEM:**

The pick expert in these application. the expert view in these application. transfer illness name and solution archive in these application they add how much charges to client. the client view the expenses and pay the sum in these application, the expert affirm the installment, they after installment, the solution view to the client, the past records view and organize in these application.

**Techniques:**

Realtime database, authentication -firebase, firebase API.

**1.3.1 OBJECTIVES:**

**Enhance Data Security and Integrity**: Utilize blockchain technology to create a secure and immutable ledger for recording patient data, ensuring that information remains accurate and protected from unauthorized alterations or tampering.

**Streamline Specialist and Department Integration**: Develop a system where various healthcare professionals, including client specialists, radiologists, surgeons, and pharmacists, can efficiently register, log in, and access patient data through a structured endorsement process by administrators.

**Maintain Confidentiality**: Implement a unique encrypted identifier system for patient data to prevent misuse and ensure that sensitive information is only accessible by authorized personnel within different departments.

**Facilitate Efficient Data Handling**: Enable seamless communication between healthcare departments by forwarding unique encrypted identifiers, ensuring that patient information is correctly and securely processed across different stages of care.

**Ensure Compliance and Reliability**: Create a reliable and compliant system that adheres to healthcare data protection standards, fostering trust among patients and healthcare providers by ensuring the confidentiality and integrity of patient information.

**1.3.2 SCOPE:**

**User Registration and Login**:

**Client Specialist**: Register and log in, with endorsement by administrator support specialists.

**Radiologists, Surgeons, and Pharmacists**: Register and log in, with endorsement by administrators.

**Administrators**: Manage the registration and endorsement process for all specialists.

**Patient Interaction**:

**Patient Registration and Login**: Patients will register and log in to the web application.

**Specialist Interaction**: Specialists will provide patients with a unique encrypted identifier after consultation.

**Data Forwarding and Processing**:

**Unique Encrypted Identifier**: Specialists will issue a unique encrypted number to patients, which is then forwarded to relevant departments (surgery, radiology, pharmacy).

**Data Confidentiality**: Ensure that the unique identifier prevents any misuse of patient details.

**Blockchain Integration**:

**Immutable Ledger**: Use blockchain to record and verify patient data, enhancing data integrity and security.

**Encryption and Access Control**: Employ encryption techniques to secure patient data and control access based on role and endorsement.

**System Management**:

**Administrator Oversight**: Administrators will oversee the registration, endorsement, and login processes for all users, ensuring compliance and security within the system.

* 1. **LITERATURE SURVEY:**

# **TITLE:** Achieving Secure, Verifiable, and Efficient Boolean Keyword Searchable Encryption for Cloud Data Warehouse

**AUTHOR**: Thanaruk Theeramunkong, Somchart Fugkeaw

**YEAR:** · January 2024

**DESCRIPTION:**

Cloud data warehouse (CDW) platforms have been offered by many cloud service providers to provide abundant storage and unlimited accessibility service to business users. Sensitive data warehouse (DW) data consisting of dimension and fact data is typically encrypted before it is outsourced to the cloud. However, the query over encrypted DW is not practically supported by any analytical query tools. The Searchable Encryption (SE) technique is palpable for supporting the keyword searches over the encrypted data. Although many SE schemes have introduced their own unique searching methods based on indexing structure on top of searchable encryption techniques, there are no schemes that support Boolean expression queries essential for the search conditions over the DW schema. In this paper, we propose a secure and verifiable searchable encryption scheme with the support of Boolean expressions for CDW. The technical construct of the proposed scheme is based on the combination of Partial Homomorphic Encryption (PHE), B+Tree and Inverted Index, and bitmapping functions to enable privacy-preserving SE with efficient search performance suitable for encrypted DW. To enhance the scalability without requiring a third party to support the verification of search results, we employed blockchain and smart contracts to automate authentication, search index retention, and trapdoor generation. For the evaluation, we conducted comparative experiments to show that our scheme is more proficient and effective than related works.

# **TITLE:** MaxD K-means: A clustering algorithm for Auto-generation of centroids and distance of data points in clusters

**AUTHOR**: Tutut Herawan, Abul Beg

**YEAR:** · January 2012

**DESCRIPTION:**

K-Means is one of the unsupervised learning and partitioning clustering algorithms. It is extremely popular and widely used for its simplicity and fastness. The main drawback of this algorithm is that user should specify the number of cluster in advance. As an iterative clustering strategy, K-Means algorithm is overly sensitive to the initial starting conditions. In this paper, we propose a clustering technique called MaxD K-Means clustering algorithm. MaxD K-Means algorithm auto generates initial k (the desired number of cluster) without asking for input from the user. MaxD K-means also used a novel strategy of setting the initial centroids. The experiment of the Max-D means has been conducted using synthetic data, which is taken from the Llyod’s K-Means experiments. The results from the new algorithm show that the number of iteration improves tremendously, and the number of iterations is reduced by confirming an improvement rate is up to 78%.

# **TITLE:** Privacy-Preserving Patient-Centric Clinical Decision Support System on Naive Bayesian Classification

**AUTHOR:** Rongxing Lu, Ximeng Liu, Jianfeng Ma

**YEAR:**January 2015

**DESCRIPTION:**

Clinical decision support system (CDSS), which uses advanced data mining techniques to help clinician make proper decisions, has received considerable attention recently. The advantages of CDSS include not only improving diagnosis accuracy but also reducing diagnosis time. Specifically, with enormous amounts of clinical data generated every day, na¨ ıve Bayesian classification can be utilized to excavate valuable information to improve CDSS. Although CDSS is quite promising, the flourish of CDSS still faces many challenges including information security and privacy concerns. In this paper, we propose a new privacy preserving patient-centric clinical decision support system, called PPCD, which helps clinician complementary to diagnose the risk of patients’ disease in a privacy-preserving way. In PPCD, the past patients’ historical data are stored in cloud and can be used to train the na¨ ıve Bayesian classifier without leaking any individual patient medical data, and then the trained classifier can be applied to compute the disease risk for new coming patients and also allow these patients to retrieve the top-k disease names according to their own preferences. Specifically, to protect the privacy of past patients’ historical data, a new cryptographic tool called additive homomorphic proxy aggregation scheme is designed. Moreover, to leverage the leakage of na¨ ıve Bayesian classifier, we introduce a privacy-preserving top-k disease names retrieval protocol in PPCD. Detailed privacy analysis ensures that patient’s information is private and will not be leaked out during the disease diagnosis phase. In addition, performance evaluation via extensive simulation also demonstrates that our PPCD can efficiently calculate patient’s disease risk with high accuracy in a privacy-preserving way.

* 1. **MERITS:**

**Merits:**

* **Privacy Protection**: Ensuring the confidentiality and security of sensitive patient information.
* **Edge Computing**: Processing data locally on devices within the healthcare institution rather than relying on centralized cloud servers.
* **Blockchain Technology**: A decentralized and secure ledger system for recording and managing data access and transactions.
* **Data Sovereignty**: Giving patients control over their medical data, allowing them to manage access permissions.
* **Security**: Measures taken to protect data from unauthorized access, modification, or disclosure.

**DEMERITS:**

It requires high energy utilization contrasted with other agreement conventions. Information exchange is unreliable.

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

This project focuses on developing a secure healthcare management system using blockchain technology. It ensures encrypted transmission and storage of sensitive patient data. The system enhances privacy, integrity, and accessibility across healthcare stakeholders.

**2.2 METHODOLOGIES:**

Methodologies is the process of analysing the principles or procedure of a Progressive Anonymous Database management system.

**2.2.1 MODULES NAME:**

* USER
* DOCTORS
* SURGEON
* RADIOLOGIST
* PHARMACIST
* ADMIN

**2.2.2 MODULE EXPLANATION:**

1. **USER REGISTER AND LOGIN:**

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database and treated in a coherent and reliable way independent of other transactions. A transaction represents any change in database user will transfer the amount to provider. They registered with data’s and then login.

1. **ADMIN LOGIN AND APPROVE:**

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database and treated in a coherent and reliable way independent of other transactions. The admin login into the application, they verify the user application and approve to check it manually in these application.

1. **DOCTORS** **REGISTER AND LOGIN:**

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database and treated in a coherent and reliable way independent of other transactions. A transaction represents any change in database user will transfer the amount to provider. They registered with datas and then login.

1. **SURGEON** **REGISTER AND LOGIN:**

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database and treated in a coherent and reliable way independent of other transactions. A transaction represents any change in database user will transfer the amount to provider. They registered with data and then login.

1. **RADIOLOGIST** **REGISTER AND LOGIN:**

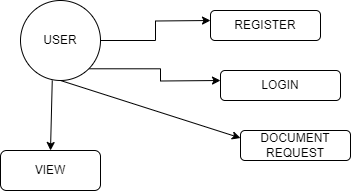
This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database and treated in a coherent and reliable way independent of other transactions. A transaction represents any change in database user will transfer the amount to provider. They registered with data and then login.

1. **PHARMACIST** **REGISTER AND LOGIN:**

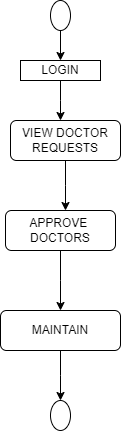
This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database and treated in a coherent and reliable way independent of other transactions. A transaction represents any change in database user will transfer the amount to provider. They registered with data and then login.

**2.2.3 MODULE DIAGRAM:**

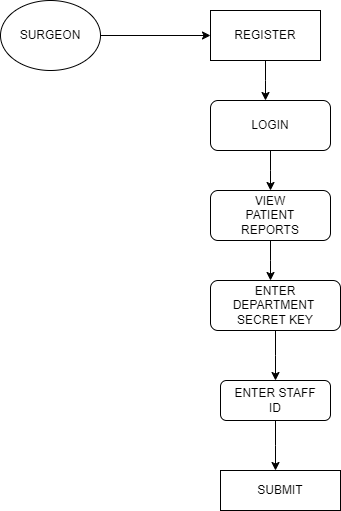
**USER:**

****

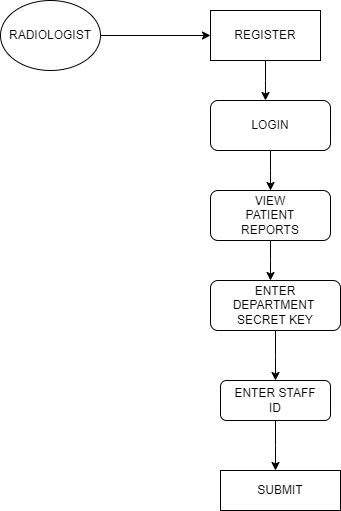
**ADMIN**

****

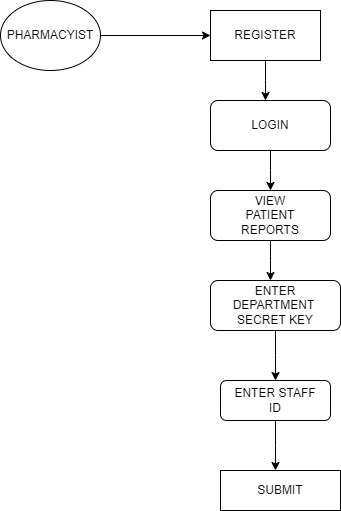
**SURGEON**

****

**RADIOLOGIST**

****

**PHARMACIST:**

****

**2.3 TECHNIQUES OR ALGORITHM:**

The system utilizes **blockchain technology** to create a decentralized and secure environment for storing and transmitting patient health data. Blockchain ensures data immutability, transparency, and traceability, eliminating the risk of unauthorized tampering.

To protect sensitive information, **Advanced Encryption Standard (AES)** is used for encrypting patient records before storing or transmitting them across the network. This guarantees confidentiality during data exchange between stakeholders.

**Smart contracts** are employed to automate and control access to patient data. These self-executing contracts enforce predefined rules, ensuring that only authorized users—such as doctors or hospitals—can access or update specific data, based on patient consent. This approach strengthens data privacy and reduces the need for centralized authorities.

**CHAPTER 3**

**REQUIREMENTS**

**3.1 GENERAL:**

To successfully design, develop, and implement the project, certain foundational tools, technologies, and software environments are essential. These general requirements outline the minimum setup necessary for ensuring the project’s smooth execution and functionality. Without meeting these prerequisites, the development process cannot proceed efficiently or may result in system incompatibility, functionality issues, or performance bottlenecks.

The general requirements are divided into three major categories:

**3.2 HARDWARE REQUIREMENTS**

* PROCESSOR : DUAL CORE 2 DUOS
* RAM : 8 GB DD RAM
* HARD DISK : 250 GB

**3.3 SOFTWARE REQUIREMENTS**

* FRONT END : HTML, CSS, JAVASCRIPT, BOOTSTRAP
* BACK END : JAVA, SPRINGBOOT
* OPERATING SYSTEM : WINDOWS 10
* IDE : SPRING TOOL SUITE
* DATABASE : MYSQL

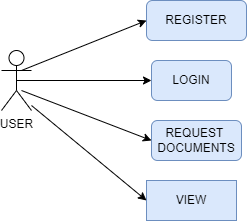
**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 GENERAL:**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

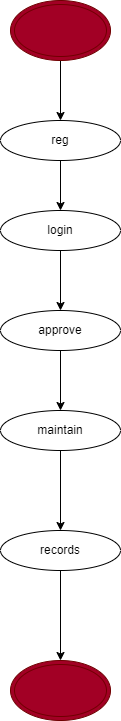
**4.1.1 USECASE DIAGRAM:**

****

**EXPLANATION:**

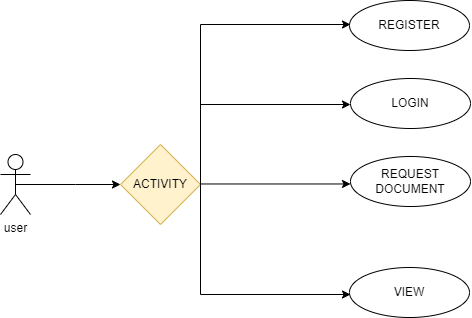
This use case diagram illustrates the primary interactions between a **User** and the system. The user can perform four main functions: **Register** to create an account, **Login** to access the system, **Request Documents** as a service, and **view** the available or requested information. Each use case represents a specific functionality offered by the system. The diagram clearly defines the system’s scope from the user’s perspective and helps in understanding the requirements for user-level interactions. It serves as a general specification for designing user-related features in the application.

**4.1.2 STATE DIAGRAM:**



**EXPLANATION:**

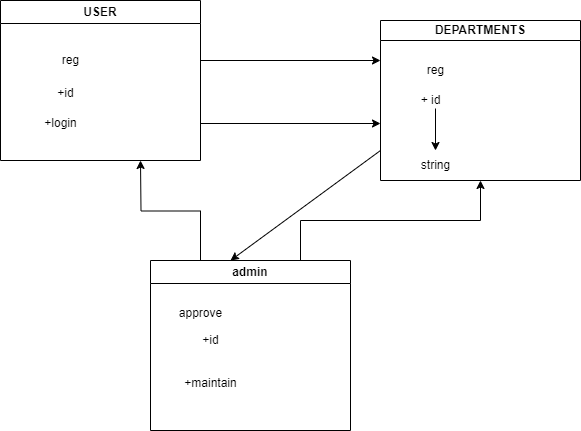
This state diagram illustrates the general workflow of a user within the system. The process begins at the initial state, which marks the start of user interaction. The first step is **registration (reg)**, where the user provides necessary details to create an account. Upon successful registration, the user moves to the **login** state to authenticate and gain access to the system. Once logged in, the user's request or profile undergoes an **approval** process, typically managed by an administrator or an automated rule engine. After approval, the system transitions to the **maintain** state, where the user can update or manage their data or services. Finally, the flow reaches the **records** state, where all user actions and data are securely stored or archived. The diagram concludes at the final state, indicating the end of the process. Each state is essential and must be completed in sequence to ensure proper functioning and data integrity within the system.

**4.1.3 ACTIVITY DIAGRAM:**

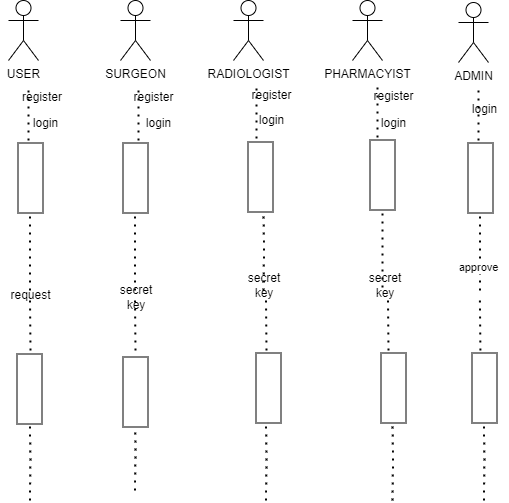
**EXPLANATION:**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc.

**4.1.4 CLASS DIAGRAM:**

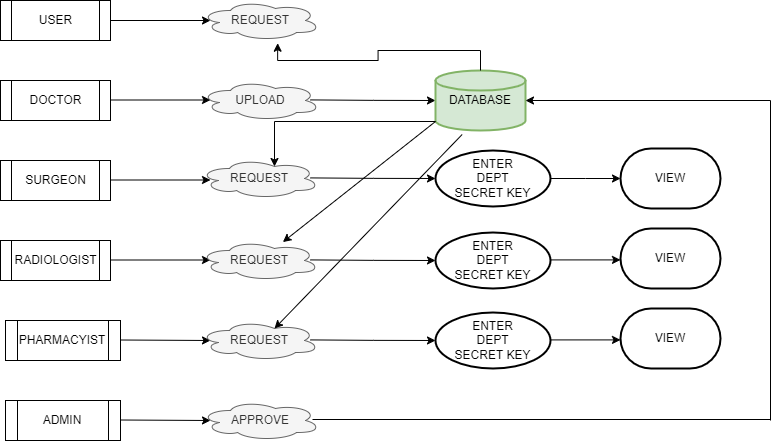
**EXPLANATION:**

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. The classes in a class diagram represent both the main objects and or interactions in the application and the objects.

**4.1.5 SEQUENCE DIAGRAM:**

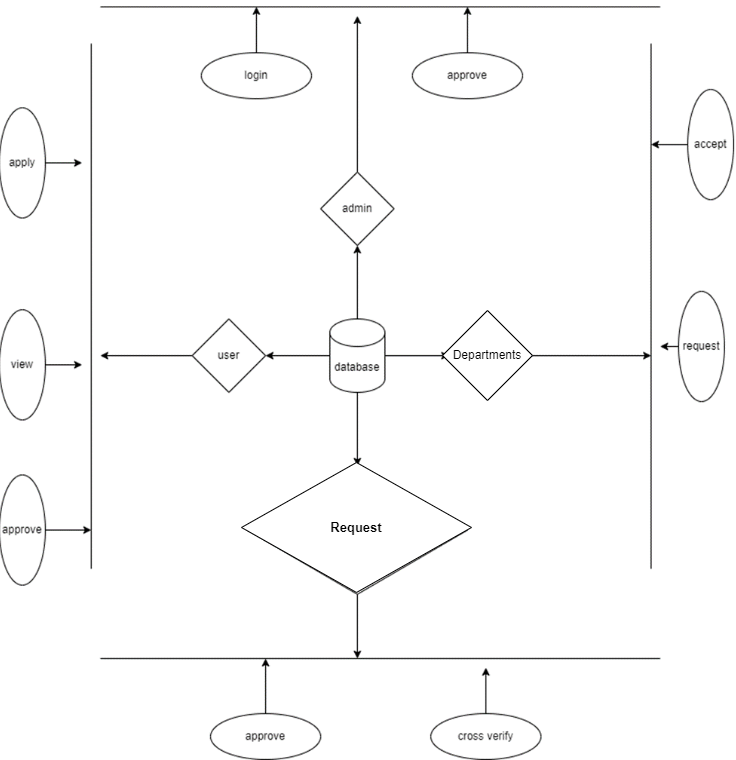
**EXPLANATION:**

In our sequence diagram specifying processes operate with one another and in order. In our sequence diagram first propose a for this in our component diagram first propose a data in this proposed method we are using Hash-Solomon Code Algorithm to encrypt the data.

**4.1.6 DATAFLOW DIAGRAM:**

**EXPLANATION:**

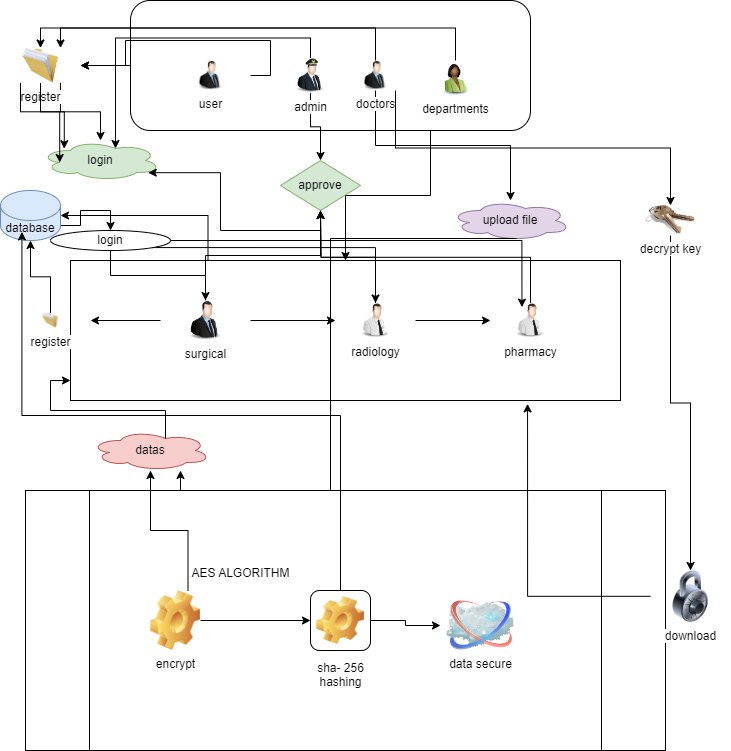
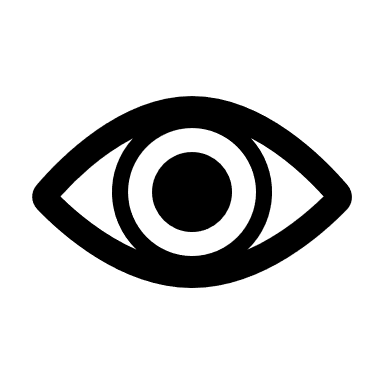
The Data Flow Diagram (DFD) of the Secure Healthcare Management System illustrates how encrypted patient data flows securely between various components. Patients register and input their medical data, which is encrypted using AES and sent to the blockchain for secure storage. Doctors or authorized healthcare providers can request access to this data through a smart contract mechanism, which verifies permissions before granting access. Admins manage user roles and system policies. The system ensures all data exchanges occur over secure channels, with blockchain maintaining an immutable record of transactions, thus enhancing trust, data integrity, and privacy across the healthcare network.

**4.1.7 E-R DIAGRAM:**

**EXPLANATION:**

An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure. For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes.

**4.1.8 SYSTEM ARCHITECTURE:**

****

view

**EXPLANATION:**

The systems architect establishes the basic structure of the system, we propose a Hash code Solomon algorithm, and we can put a small part of data in local machine and fog server to protect the privacy. Moreover, based on computational intelligence, this algorithm can compute the distribution proportion stored in cloud, fog, and local machine, respectively. Through the theoretical safety analysis and experimental evaluation, the feasibility of our scheme has been validated, which is really a powerful supplement to existing cloud storage scheme.

**CHAPTER 5**

**SOFTWARE SPECIFICATION**

**5.1 GENERAL:**

This chapter is about the software language and the tools used in the development of the project. The platform used here is JAVA. The Primary languages are JAVA, J2EE and J2ME. In this project J2EE is chosen for implementation.

**5.2 FEATURES OF JAVA:**

**5.2.1 THE JAVA FRAMEWORK:**

**Java** is a programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century and is widely used from application software to web applications the java framework is a new platform independent that simplifies application development internet. Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacentres, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

**5.2.2 OBJECTIVES OF JAVA:**

To see places of Java in Action in our daily life, explore java.com.

**WHY SOFTWARE DEVELOPERS CHOOSE JAVA:**

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

* Write software on one platform and run it on any other platform.
* Create programs to run within a Web browser and Web services.
* Develop server-side applications for online forums, stores, polls, HTML forms processing, and more.
* Combine applications or services using the Java language to create highly customized applications or services.
* Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and any other device with a digital heartbeat.

**Some Ways Software Developers Learn Java:**

Today, many colleges and universities offer courses in programming for the Java platform. In addition, developers can also enhance their Java programming skills by reading Sun's java.sun.com Web site, subscribing to Java technology-focused newsletters, using the Java Tutorial and the New to Java Programming Centre, and signing up for Web, virtual, or instructor-led courses.

**Object Oriented:**

To be an Object-Oriented language, any language must follow at least the four characteristics.

1. Inheritance: It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding addition a feature as needed.

2. Encapsulation: It is the mechanism of combining the information and providing the abstraction.

3. Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.

4. Dynamic binding: Sometimes we do not have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

**5.2.3 JAVA SERVER PAGES – An Overview:**

Java Server Pages or JSP for short is Sun's solution for developing dynamic web sites. JSP provide excellent server-side scripting support for creating database driven web applications. JSP enable the developers to directly insert java code into jsp file, this makes the development process quite simple, and its maintenance also becomes extremely easy.

JSP pages are efficient, it loads into the web server’s memory on receiving the request very first time and the subsequent calls are served within a very short period.

In today's environment most web sites servers’ dynamic pages based on user request. Database is very convenient way to store the data of users and other things. JDBC provide excellent database connectivity in heterogeneous database environment. Using JSP and JDBC its very cc easy to develop database driven web application.

Java is known for its characteristic of "write once, run anywhere." JSP pages are platform Java Server Pages.

Java Server Pages (JSP) technology is the Java platform technology for delivering dynamic content to web clients in a portable, secure, and well-defined way. The Java Server Pages specification extends the Java Servlet API to provide web application developers with a robust framework for creating dynamic web content on the server using HTML, and XML templates, and Java code, which is secure, fast, and independent of server platforms.

JSP has been built on top of the Servlet API and utilizes Servlet semantics. JSP has become the preferred request handler and response mechanism. Although JSP technology is going to be a powerful successor to basic Servlets, they have an evolutionary relationship and can be used in a cooperative and complementary manner.

Servlets are powerful and sometimes they are a bit cumbersome when it comes to generating complex HTML. Most servlets contain a little code that handles application logic and a lot more code that handles output formatting. This can make it difficult to separate and reuse portions of the code when a different output format is needed. For these reasons, web application developers turn towards JSP as their preferred servlet environment.

**5.2.4 EVOLUTION OF WEB APPLICATIONS:**

Over the last few years, web server applications have evolved from static to dynamic applications. This evolution became necessary due to some deficiencies in earlier web site design. For example, to put more of business processes on the web, whether in business-to-consumer (B2C) or business-to-business (B2B) markets, conventional web site design technologies are not enough. The key issues, every developer faces when developing web applications, are:

1. Scalability - a successful site will have more users and as the number of users is increasing Fastly, the web applications must scale correspondingly.

2. Integration of data and business logic - the web is just another way to conduct business, and so it should be able to use the same middle-tier and data-access code.

3. Manageability - web sites just keep getting bigger and we need some viable mechanism to manage the ever-increasing content and its interaction with business systems.

4. Personalization - adding a personal touch to the [web page](http://www.roseindia.net/jsp/javaserverpagestutorial.shtml) becomes an essential factor to keep our customer coming back again. Knowing their preferences, allowing them to configure the information they view, remembering their past transactions or frequent search keywords are all important in providing feedback and interaction from what is otherwise a one-sided conversation.

Apart from these general needs for a business-oriented web site, the necessity for modern technologies to create robust, dynamic, and compact server-side web applications has been realized. The main characteristics of today's dynamic web server applications are as follows:

1. Serve HTML and XML, and stream data to the web client.

2. Separate presentation, logic, and data.

3. Interface to databases, other Java applications, CORBA, directory, and mail services.

4. Make use of application server middleware to provide transactional support.

5. Track client sessions.

**5.2.5 BENEFITS OF JSP:**

One of the main reasons why the Java Server Pages technology has evolved into what it is today, and it is still evolving is the overwhelming technical need to simplify application design by separating dynamic content from static template display data. Another benefit of utilizing JSP is that it allows to more cleanly separate the roles of web application/HTML designer from a software developer. The JSP technology is blessed with a few exciting benefits, which are chronicled as follows:

1. The JSP technology is platform independent, in its dynamic web pages, its web servers, and its underlying server components. That is, JSP pages perform perfectly without any hassle on any platform, run on any web server, and web-enabled application server. The JSP pages can be accessed from any web server.

2. The JSP technology emphasizes the use of reusable components. These components can be combined or manipulated towards developing more purposeful components and page design. This reduces development time apart from the at development time, JSPs are quite different from Servlets, however, they are precompiled into Servlets at run time and executed by a JSP engine which is installed on a Web-enabled application server such as BEA WebLogic and IBM WebSphere.

**5.3 SERVLETS:**

Earlier in client- server computing, each application had its own client program, and it worked as a user interface and need to be installed on each user's personal computer. Most web applications use HTML/XHTML that are mostly supported by all the browsers and web pages are displayed to the client as static documents.

A web page can merely display static content, and it also lets the user navigate through the content, but a web application provides a more interactive experience.

Any computer running Servlets or JSP needs to have a container. A container is nothing but a piece of software responsible for loading, executing, and unloading the Servlets and JSP. While servlets can be used to extend the functionality of any Java- enabled server.

They are mostly used to extend web servers and are efficient replacement for CGI scripts. CGI was one of the earliest and most prominent server-side dynamic content solutions, so before going forward it is especially important to know the difference between CGI and the Servlets.

**5.4 JAVA SERVLETS:**

Java Servlet is a generic server extension that means a java class can be loaded dynamically to expand the functionality of a server. Servlets are used with web servers and run inside a Java Virtual Machine (JVM) on the server, so these are safe and portable. Unlike applets they do not require support for java in the web browser. Unlike CGI, servlets do not use multiple processes to handle separate request. Servlets can be handled by separate threads within the same process. Servlets are also portable and platform independent.

A web server is the combination of computer, and the program installed on it. Web server interacts with the client through a web browser. It delivers the [web pages](http://www.roseindia.net/servlets/IntroductionToWebServer.shtml) to the client and to an application by using the web browser and  he HTTP protocols respectively.

The define the web server as the package of  large number of programs installed on a computer connected to Internet or intranet for downloading the requested files using [File Transfer](http://www.roseindia.net/servlets/IntroductionToWebServer.shtml) Protocol, serving e-mail and building and publishing web pages. A web server works on a client server model.

**5.5 CONCLUSION:**

JSP and Servlets are gaining rapid acceptance as means to provide dynamic content on the Internet. With full access to the Java platform, running from the server in a secure manner, the application possibilities are almost limitless. When JSPs are used with Enterprise JavaBeans technology, e-commerce and database resources can be further enhanced to meet an enterprise's needs for web applications providing secure transactions in an open platform. J2EE technology makes it easy to develop, deploy and use web server applications instead of mingling with other technologies such as CGI and ASP. There are many tools for facilitating quick web software development and to easily convert existing server-side technologies to JSP and Servlets.

**CHAPTER 6**

**IMPLEMENTATION**

* 1. **GENERAL:**

This chapter describes the implementation of searched based application. It deals with the source code for main viewpoint for Anonymous Database Management.

* 1. **CODINGS:**

<!**DOCTYPE** html>

<**html** lang=*"en"* xmlns:th=*"http://www.thymeleaf.org"*>

<**head**>

<**meta** charset=*"UTF-8"*>

<**meta** name=*"viewport"* content=*"width=device-width, initial-scale=1.0"*>

<**title**>Welcome | Doctors</**title**>

<**link** href=*"https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"* rel=*"stylesheet"* integrity=*"sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"* crossorigin=*"anonymous"*>

<**script** src=*"https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"* integrity=*"sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"* crossorigin=*"anonymous"*></**script**>

<**link** rel=*"shortcut icon"* href=*"C:\Users\saura\Downloads\fabicon-removebg-preview (2).png"*>

<**body**>

<!-- Header -->

<**header**>

<**div** class=*"motto"*><**b**>Welcome</**b**></**div**>

<**div** class=*"slideshow"*>

<**img** src=*"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRdji1931PbuaaeOzkiWxZwCP0Y-l4Xi1WwR1sD38DAMHght4-40fDGDgm8uf3LQa0zNBU&usqp=CAU"* alt=*"Image 1"*>

<**img** src=*"https://t3.ftcdn.net/jpg/05/65/55/96/360\_F\_565559600\_YuYjetRZf0xO4hEzElmYPtWXI7bdpXeE.jpg"* alt=*"Image 2"*>

<**img** src=*"https://plus.unsplash.com/premium\_photo-1674841252366-162fc9b8f983?q=80&w=1887&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 3"*>

<**img** src=*"https://plus.unsplash.com/premium\_photo-1673351534430-f5dde7d87a60?q=80&w=2002&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 4"*>

<**img** src=*"https://plus.unsplash.com/premium\_photo-1682141160157-d6a296a65574?q=80&w=1932&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 5"*>

<**img** src=*"https://plus.unsplash.com/premium\_photo-1674499074438-8f611a3569f6?q=80&w=2069&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 6"*>

<**img** src=*"https://images.unsplash.com/photo-1638202993928-7267aad84c31?q=80&w=1887&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 7"*>

<**img** src=*"https://images.unsplash.com/photo-1550831107-1553da8c8464?q=80&w=1887&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 8"*>

<**img** src=*"https://images.unsplash.com/photo-1505751172876-fa1923c5c528?q=80&w=2070&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 9"*>

<**img** src=*"https://images.unsplash.com/photo-1578496479763-c21c718af028?q=80&w=2070&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 10"*>

<**img** src=*"https://images.unsplash.com/photo-1543333995-a78aea2eee50?q=80&w=2070&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 11"*>

<**img** src=*"https://images.unsplash.com/photo-1582560469781-1965b9af903d?q=80&w=1993&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 12"*>

<**img** src=*"https://plus.unsplash.com/premium\_photo-1661779616524-1825ac40c9d6?q=80&w=2070&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"* alt=*"Image 13"*>

</**div**>

<**style**>

/\* Reset CSS \*/

{

margin: *0*;

padding: *0*;

box-sizing: *border-box*;

}

body {

font-family: *Arial, sans-serif*;

background-color: *#f5f5f5*;

color: *#333*;

margin: *0*;

padding: *0*;

}

/\* Header \*/

header {

position: *relative*;

overflow: *hidden*;

width: *100%*;

height: *400px*; /\* Adjust height as needed \*/

display: *flex*;

justify-content: *center*;

align-items: *center*;

}

.slideshow {

position: *absolute*;

top: *0*;

left: *0*;

width: *400%*;

height: *100%*;

display: *flex*;

animation: *slideImages 60s linear infinite*;

}

.slideshow img {

width: *25%*;

height: *auto*;

object-fit: *cover*;

}

@keyframes slideImages {

0% {

transform: *translateX(0%)*;

}

100% {

transform: *translateX(-100%)*;

}

}

.logo {

position: *relative*;

top: *20px*;

left: *10px*;

z-index: *1*;

}

.logo img {

width: *400px*; /\* Adjust the width of the logo \*/

height: *auto*;

filter: *brightness(150%)*; /\* Increase brightness for better visibility \*/

}

.motto {

position: *absolute*;

bottom: *20px*;

left: *50%*;

transform: *translateX(-50%)*;

z-index: *1*;

color: *#FFA500*; /\* Adjust color for better visibility \*/

font-size: *24px*; /\* Increase font size for better visibility \*/

}

/\* Navigation \*/

nav {

text-align: *center*;

padding: *10px 0*;

background-color: *rgb(0, 0, 0)*;

}

nav a {

color: *#fff*;

text-decoration: *none*;

padding: *10px 20px*;

}

nav a:hover {

background-color: *#555*;

}

/\* Main Content \*/

.main-content {

padding: *20px*;

display: *flex*;

justify-content: *space-between*;

align-items: *center*;

flex-wrap: *wrap*;

}

.main-content .left-column {

flex: *1*;

max-width: *50%*;

padding-right: *20px*;

}

.main-content .right-column {

flex: *1*;

max-width: *50%*;

padding-left: *20px*;

}

/\* Services Section Styles \*/

/\* Services Section Styles \*/

.services {

background-color: *#f9f9f9*;

padding: *20px*;

margin-bottom: *20px*;

}

.services h2 {

text-align: *center*;

margin-bottom: *20px*;

}

.service-container {

display: *flex*;

justify-content: *space-between*;

align-items: *flex-start*;

}

.service {

flex: *1*;

margin: *0 10px*;

background-color: *#fff*;

border-radius: *8px*;

box-shadow: *0 0 20px rgba(0, 0, 0, 0.1)*;

overflow: *hidden*;

}

.service-info {

padding: *20px*;

}

.service h3 {

margin-bottom: *10px*;

color: *#333*;

</**html**>

**package com.spring.graph.api.encryptionservice;**

**import org.springframework.stereotype.Service;**

**import com.spring.graph.api.algorithms.AESUtil;**

**import javax.crypto.SecretKey;**

**@Service**

**public class EncryptionService {**

**private final SecretKey secretKey;**

**public EncryptionService() throws Exception {**

**// Generate a new AES key. You can save this key securely and reuse it.**

**this.secretKey = AESUtil.generateKey(256);**

**}**

**public String encryptData(byte[] bs) throws Exception {**

**return AESUtil.encrypt(bs, secretKey);**

**}**

**public String decryptData(String encryptedData) throws Exception {**

**return AESUtil.decrypt(encryptedData, secretKey);**

**}**

**public String getSecretKey() {**

**return AESUtil.keyToString(secretKey);**

**}**

**}**

**package com.spring.graph.api.controller;**

**import java.io.IOException;**

**import java.nio.file.Files;**

**import java.nio.file.Path;**

**import java.nio.file.Paths;**

**import java.security.SecureRandom;**

**import java.util.List;**

**import java.io.File;**

**import org.springframework.http.HttpHeaders;**

**import org.springframework.http.MediaType;**

**import org.springframework.web.bind.annotation.GetMapping;**

**import org.springframework.web.bind.annotation.RequestParam;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.core.io.FileSystemResource;**

**import org.springframework.http.ResponseEntity;**

**import org.springframework.ui.Model;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.PathVariable;**

**import org.springframework.web.bind.annotation.PostMapping;**

**import org.springframework.web.multipart.MultipartFile;**

**import com.spring.graph.api.entity.Adminentity;**

**import com.spring.graph.api.entity.Docterreg;**

**import com.spring.graph.api.entity.sdocrequet;**

**import com.spring.graph.api.repository.Docregrepository;**

**import com.spring.graph.api.repository.patientdocrepo;**

**import com.spring.graph.api.services.Docservice;**

**import com.spring.graph.api.services.patientdocservice;**

**import jakarta.servlet.http.HttpSession;**

**@org.springframework.stereotype.Controller**

**public class Controller {**

**@Autowired**

**Docservice docservice;**

**@Autowired**

**private Docregrepository docrepo;**

**@Autowired**

**private patientdocrepo prepo;**

**@Autowired**

**private patientdocservice pservice;**

**private static final String CHARACTERS = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";**

**private static final int KEY\_LENGTH = 5;**

**@GetMapping("/")**

**public String index() {**

**return "index";**

**}**

**@GetMapping("/Docloginn")**

**public String home() {**

**return "Doctors";**

**}**

**@GetMapping("/drequest")**

**public String docreq() {**

**return "drequest";**

**}**

**@GetMapping("/logout")**

**public String logout() {**

**return "Doctors";**

**}**

**@GetMapping("/adminlogin")**

**public String adminlogin() {**

**return "Adminlogin";**

**}**

**@GetMapping("/docreg")**

**public String docreg() {**

**return "docterreg";**

**}**

**//**

**@PostMapping("/docterreg")**

**public String docterreg(@RequestParam("name") String name, @RequestParam("image") MultipartFile Image,**

**@RequestParam("name") String Lname, @RequestParam("Email") String Email,**

**@RequestParam("password") String password,**

**@RequestParam("doctors") String doctors,**

**@RequestParam("contact") String contact,**

**Model model) {**

**// Create a new Dress object**

**Docterreg ob = new Docterreg();**

**ob.setName(name);**

**ob.setStatus("request");**

**ob.setEmail(Email);**

**ob.setDoctors(doctors);**

**ob.setPassword(password);**

**ob.setContact(contact);**

**boolean a = docservice.checkEmail(Email);**

**String message = (a == true) ? "This Mail was Already Exist.." : "Register successfully..";**

**model.addAttribute("msg", message);**

**if (a == false) {**

**try {**

**// Save the dress image**

**String fileName = Image.getOriginalFilename();**

**byte[] imageBytes = Image.getBytes();**

**// Specify the directory where the image will be saved**

**String directory = "src/main/resources/static/images/";**

**// Create the directory if it doesn't exist**

**Files.createDirectories(Paths.get(directory));**

**// Specify the image file path**

**Path imagePath = Paths.get(directory + fileName);**

**// Write the image bytes to the specified path**

**Files.write(imagePath, imageBytes);**

**ob.setImage(fileName);**

**} catch (IOException e) {**

**// Handle the exception (e.g., log the error, show an error message)**

**e.printStackTrace();**

**}**

**docrepo.save(ob);**

**return "Doctors";**

**} else {**

**return "error";**

**}**

**}**

**//Doctor login logics**

**@PostMapping("/dlog")**

**public String loginus(Docterreg sell, Model model) {**

**//UserDtls user = new UserDtls();**

**System.out.println(sell.getEmail());**

**System.out.println(sell.getPassword());**

**Docterreg st=docservice.getdocByEmailAndPassword(sell.getEmail(), sell.getPassword());**

**if (st != null) {**

**// Session attributes are already set in UserService**

**System.out.println(sell.getEmail());**

**// Add email attribute to the model**

**return "docinterface";**

**} else {**

**System.out.println(st);**

**return "Doctors";**

**}**

**}**

**//admin**

**@PostMapping("/adminauthentication")**

**public String loginUser(@ModelAttribute Adminentity hos, Model m) {**

**if (hos.getEmail().equals("admin@gmail.com") && hos.getPassword().equals("admin")) {**

**System.out.println("success");**

**return "adminhome";**

**} else {**

**System.out.println("failure");**

**m.addAttribute("msg", "Invalid username or password");**

**return "error";**

**}**

**}**

**@GetMapping("/viewrequest")**

**public String viewpatient(HttpSession session, Model model) {**

**Docterreg existingUser = docrepo.findByStatus("request");**

**if (existingUser != null) {**

**model.addAttribute("doctorRequests", existingUser);**

**System.out.println(" userappointment " + existingUser.toString());**

**return "drequest";**

**}**

**else {**

**System.out.println("No user found with status 'request'");**

**return "nodata"; // Return a view indicating no data found**

**}**

**}**

**@GetMapping("/viewPDF/{filename}")**

**public ResponseEntity<FileSystemResource> viewPDF(@PathVariable("filename") String filename) {**

**// Directory where PDF files are stored**

**String directory = "src/main/resources/static/images/";**

**// File path based on filename**

**String filePath = directory + filename;**

**// Create a file object**

**File file = new File(filePath);**

**// Check if the file exists**

**if (file.exists()) {**

**// Set content type as application/pdf**

**HttpHeaders headers = new HttpHeaders();**

**headers.setContentType(MediaType.APPLICATION\_PDF);**

**// Return ResponseEntity with the file**

**return ResponseEntity.ok()**

**.headers(headers)**

**.body(new FileSystemResource(file));**

**} else {**

**// Return ResponseEntity with 404 Not Found status**

**return ResponseEntity.notFound().build();**

**}**

**}**

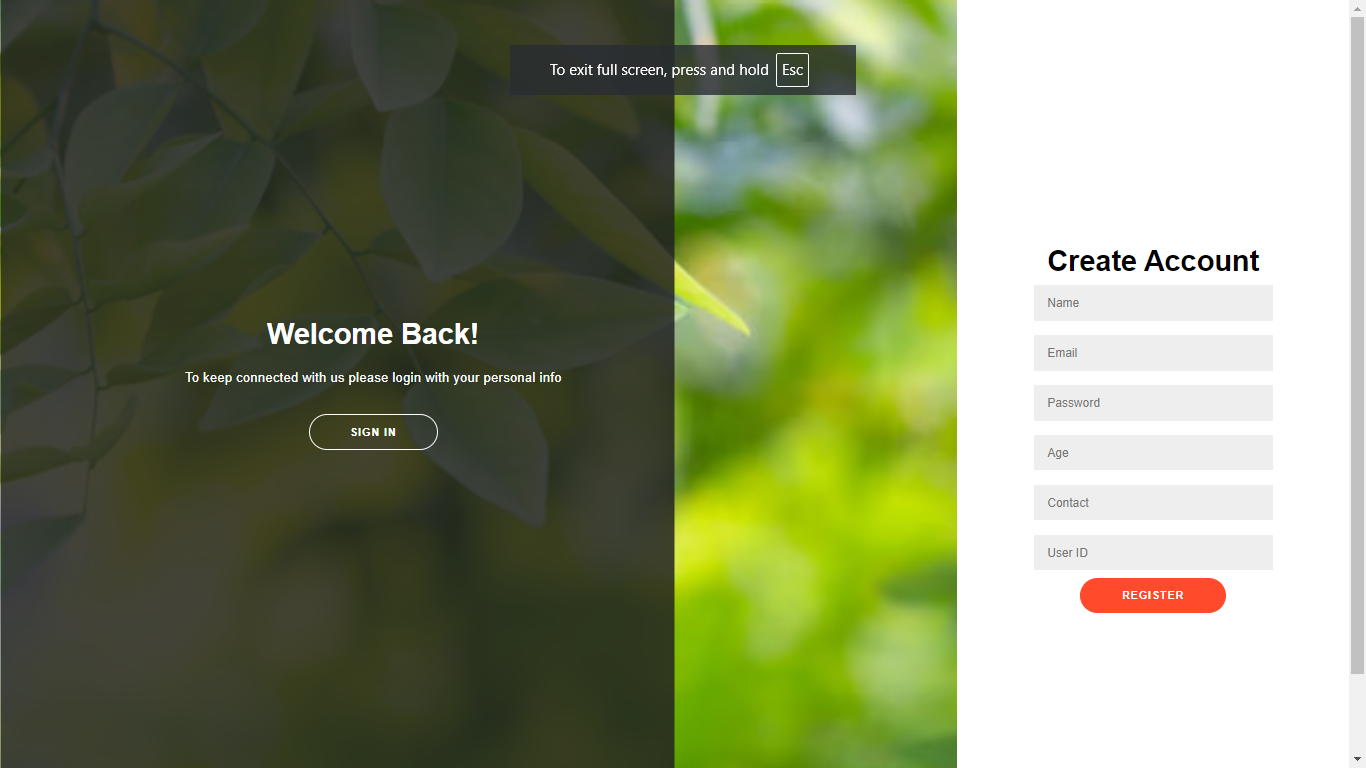
**CHATPTER 7**

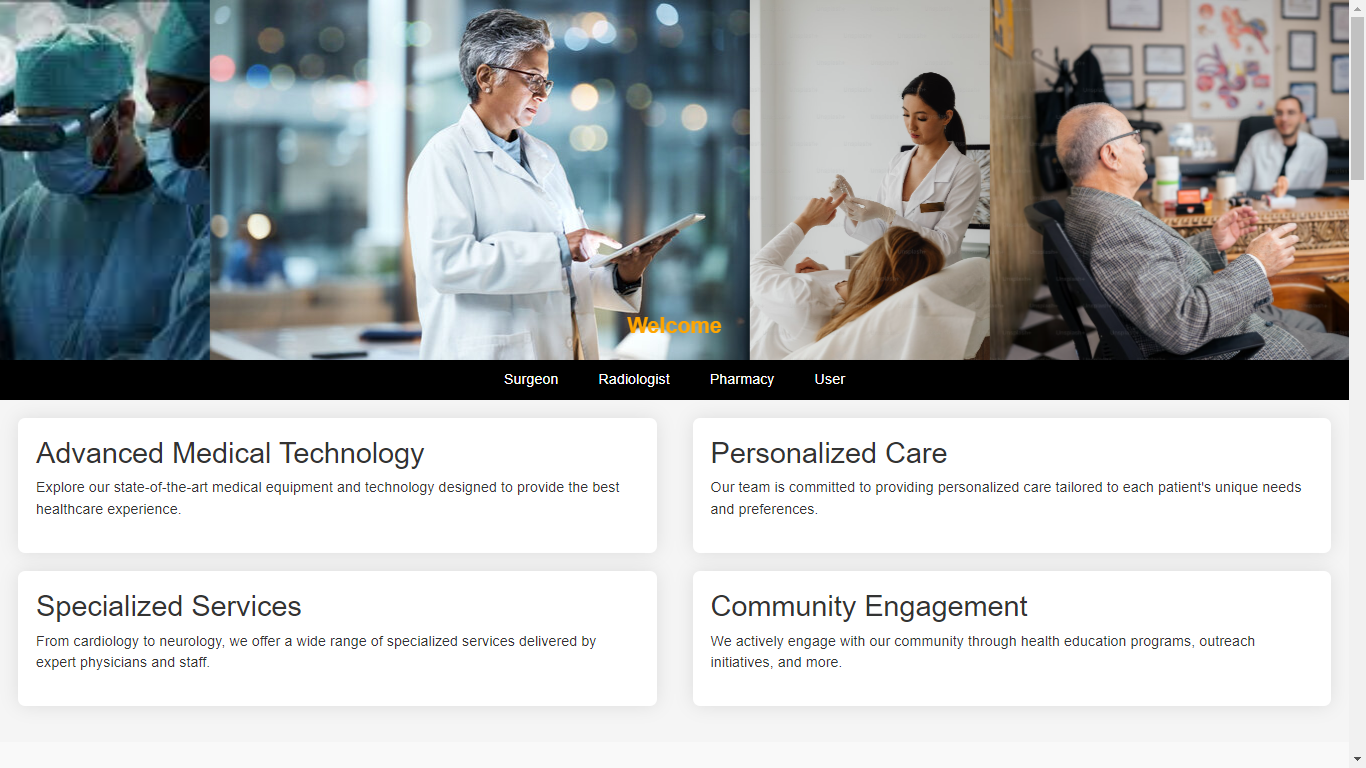
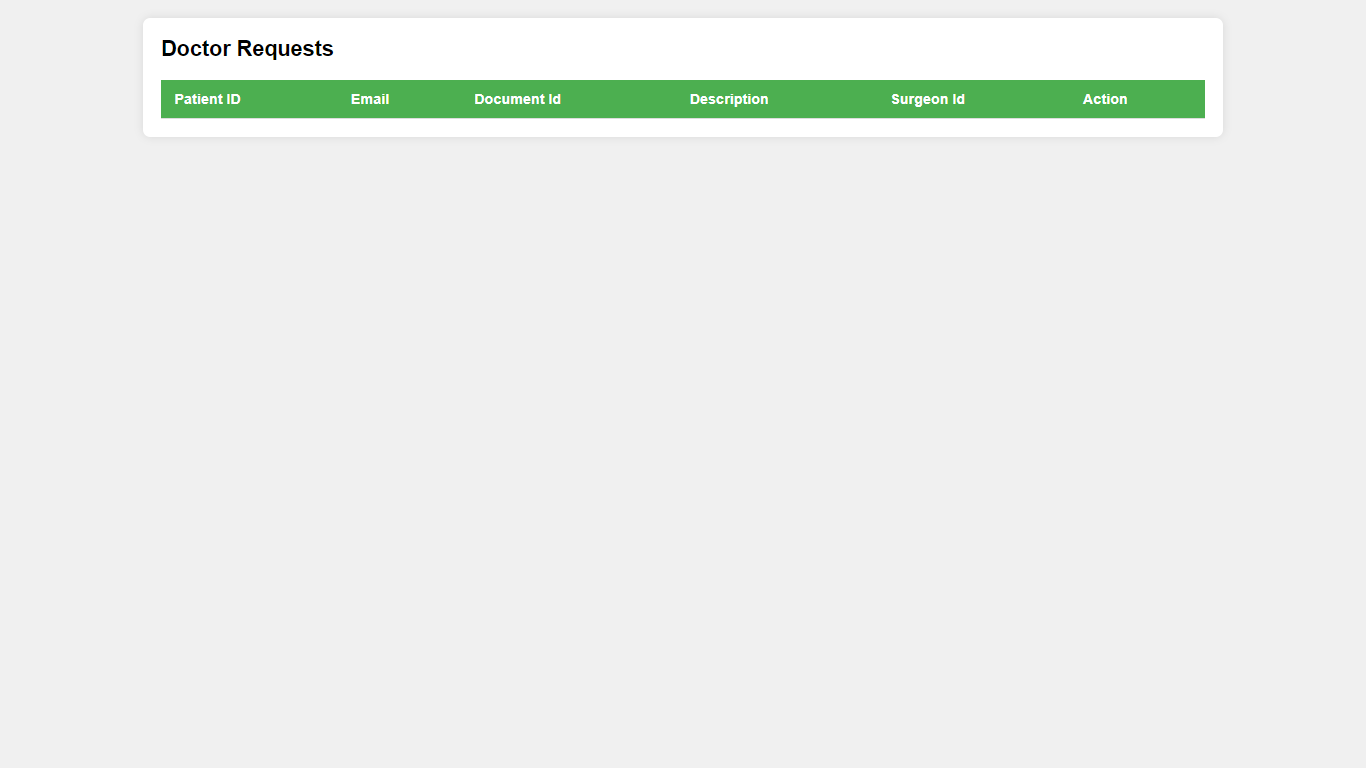
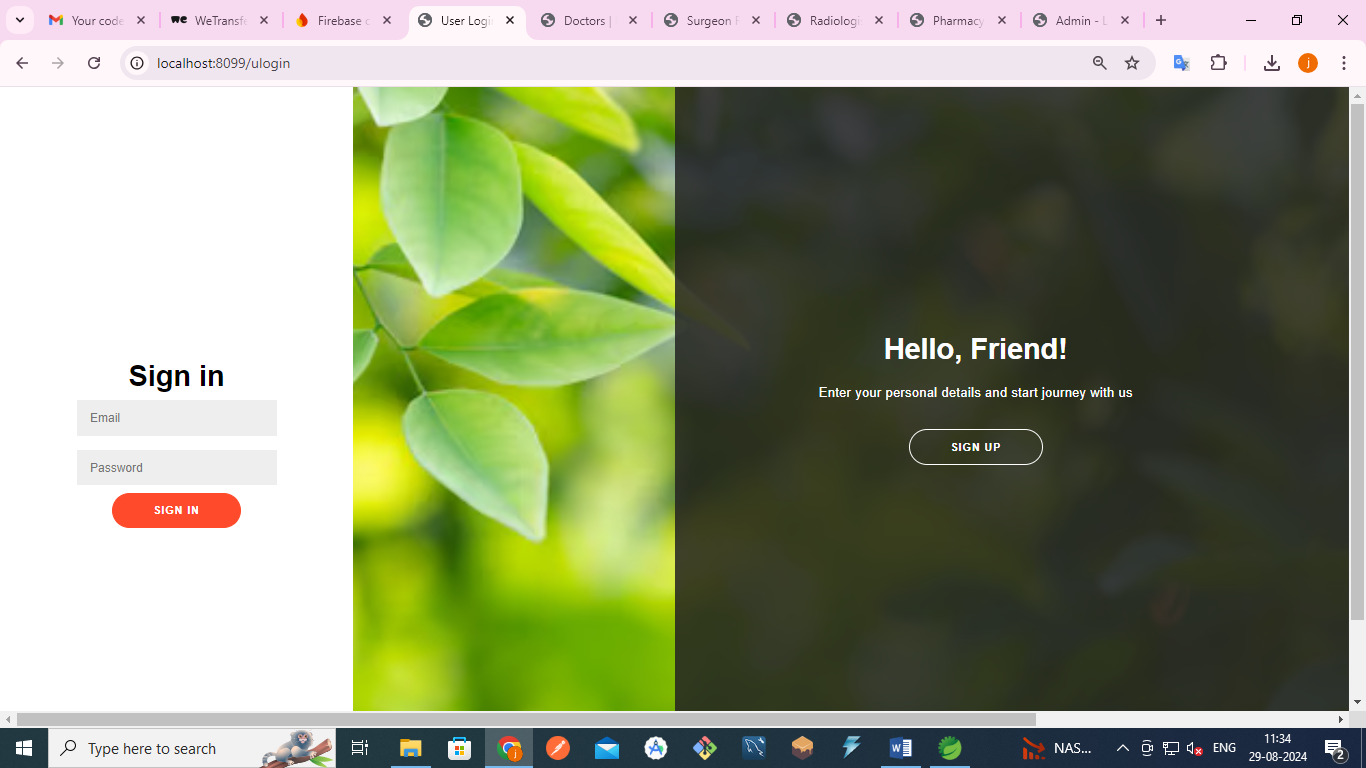
**SNAPSHOTS**

**7.1 GENERAL:**

This section presents a series of snapshots illustrating the key features and user interfaces of the Secure Healthcare Management System developed using blockchain technology. The snapshots provide a visual representation of how the system functions from both the patient and healthcare provider perspectives.

**7.2 SNAPSHOTS:**

****

****

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 FEASIBILITY STUDY:**

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of the existing business or proposed venture, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for success.

In its simplest term, the two criteria to judge feasibility are cost required and value to be attained. As such, a well-designed feasibility study should provide a historical background of the business or project, description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation.

They are 3 types of Feasibility

* Economical feasibility
* Technical feasibility
* Operational feasibility

**8.1.1 ECONOMICAL FEASIBILITY:**

The assessment is based on an outline design of system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in terms of volumes of data, trends, frequency of updating, etc. to estimate whether the new system will perform adequately or not.

**8.1.2 TECHNICAL FEASIBILITY:**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand for the available technical resources.

**8.1.3 OPERATIONAL FEASIBILITY:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

**8.2 SYSTEM TESTING:**

The software which has been developed, has to be tested to prove its validity. Testing is the least creative phase of the whole cycle of system design. In the real sense it is the phase which helps to bring out the creativity of the other phases makes it shine.

**8.2.1. VARIOUS LEVELS OF TESTING:**

1. White Box Testing

2. Black Box Testing

3. Unit Testing

4. Functional Testing

5. Performance Testing

6. Integration Testing

7. Validation Testing

8. System Testing

9. Output Testing

10. User Acceptance Testing

**8.2.1.1. WHITE BOX TESTING:**

White-box testing, sometimes called glass-box, is a test case design method that uses the control structure of the procedural design to derive test cases. Using White Box testing methods, we can derive test cases that

• Guarantee that all independent paths within a module have been exercised at least once

• Exercise all logical decisions on their true and false sides.

• Execute all loops at their boundaries and within their operational bounds.

• Exercise internal data structures to assure their validity.

**8.2.1.2. BLACK BOX TESTING:**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, like most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a test in which the software under the test is treated as a black box. You cannot “see” into it. The test provides input and responds to outputs without considering how the software works.

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

**8.2.1.3. UNIT TESTING:**

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process.

Unit testing is software verification and validation method in which the individual units of source code are tested fit for use. A unit is the smallest testable part of an application. In this testing, each class is tested to be working satisfactorily.

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. it is done after the completion of an individual unit before integration.

**8.2.1.4. FUNCTIONAL TESTING:**

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes what the system does. Functional testing differs from system testing in that functional testing "verifies a program by checking it against ... design document(s) or specification(s)", while system testing "validate a program by checking it against the published user or system requirements" (Kane, Falk, Nguyen 1999, p. 52). Functional testing typically involves five steps. The identification of functions that the software is expected to perform

1. The creation of input data based on the function's specifications

2. The determination of output based on the function's specifications

3. The execution of the test case

4. The comparison of actual and expected outputs.

**8.2.1.5. PERFORMANCE TESTING:**

In general testing is performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system.

**8.2.1.6. INTEGRATION TESTING:**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when put together. The problem of course, is “putting them together”- interfacing. There may be the chances of data loss across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready. All the errors found in the system are corrected for the next phase.

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested, and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing.

**8.2.1.7. VALIDATION TESTING:**

Verification and Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it full fills its intended purpose. These are critical components of a quality management system such as ISO 9000. The words "verification" and "validation" are sometimes preceded with "Independent" (or IV&V), indicating that the verification and validation is to be performed by a disinterested third party.

It is sometimes said that validation can be expressed by the query "Are you building the right thing?" and verification by "Are you building it right?". In practice, the usage of these terms varies. Sometimes they are even used interchangeably.

**8.2.1.8. SYSTEM TESTING:**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all the "integrated" software components that have passed integration testing and the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification.

**8.2.1.9. OUTPUT TESTING:**

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output generated or considered into two ways. One is on screen and another is printed format. The output comes as the specified requirements by the user. Hence output testing does not result in any correction in the system.

**8.2.1.10. USER ACCEPTANCE TESTING:**

User acceptance of a system is the factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required.

• Input screen design.

• Output screen design.

• Online message to guide users.

• Format of the ad-hoc reports and other outputs.

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in system testing. After preparing the test data the system under study is tested using the test data. While testing the system by using test data errors are again uncovered and correct.

**CHAPTER 9**

**APPLICATION AND FUTURE ENHANCEMENT**

**9.1 GENERAL:**

The integration of blockchain technology in healthcare management addresses the growing concerns of data security, privacy, and interoperability. This system ensures that patient records are securely encrypted, transmitted, and accessed only by authorized entities. By leveraging blockchain’s decentralized architecture, the proposed system eliminates the risk of single-point failures and enhances data integrity.

In the developed system, each patient’s medical information is stored as a secure transaction on a private blockchain network. Smart contracts control data access and sharing based on pre-defined permissions, ensuring only authenticated users can access or update sensitive information. This design fosters trust among stakeholders—patients, doctors, and hospitals—while adhering to compliance standards such as HIPAA and GDPR.

**9.2 APPLICATIONS:**

The blockchain-based secure healthcare management system can be applied in several real-world scenarios:

* Hospitals and Clinics: Secure and seamless sharing of patient records across departments and facilities without data duplication or breaches.
* Telemedicine: Real-time encrypted sharing of health data during virtual consultations, maintaining confidentiality and compliance.
* Medical Research: Anonymized patient data can be securely shared with research institutions, ensuring traceability and consent.
* Pharmaceutical Supply Chain: Integration with medical supply chains to trace the origin and authenticity of drugs prescribed to patients.
* Insurance Processing: Automated and verifiable claims processing using smart contracts, reducing fraud and delays.
* Cross-border Healthcare: Secure and authorized access to patient records for travelers or migrants seeking treatment in different regions.

**9.3 FUTURE ENHANCEMENTS:**

To further improve and scale the system, the following future enhancements can be considered:

* **AI Integration**: Incorporate artificial intelligence for predictive analytics and intelligent diagnostics using the securely stored patient data.
* **IoT Device Integration**: Enable real-time data collection from wearable and implantable health monitoring devices, with blockchain ensuring data authenticity.
* **Interoperability Standards**: Adopt emerging healthcare data standards such as HL7 FHIR to support broader system compatibility.
* **Decentralized Identity (DID)**: Implement blockchain-based patient identification systems for more secure and portable digital health identities.
* **Scalability Improvements**: Use Layer-2 blockchain solutions or sidechains to enhance transaction throughput and reduce latency.
* **Mobile Application**: Develop a user-friendly mobile app for patients and doctors to access and manage records with biometric authentication.
* **Smart Health Cards**: Introduce blockchain-enabled health cards that carry patient credentials and medical history for emergency access.

**CHAPTER 10**

**CONCLUSION & REFERENCES**

**10.1 CONCLUSION:**

In summary, your project integrates blockchain technology to enhance the integrity and confidentiality of patient data within a healthcare system. The system involves multiple roles, including client specialists, radiologists, surgeons, pharmacists, and administrators, each with specific responsibilities and access rights. After a patient registers and logs in, they receive a unique encrypted number from their specialist, which is then forwarded to various departments such as surgery, radiology, and pharmacy. This unique number ensures that sensitive patient information is protected and cannot be misused. The blockchain framework guarantees data integrity by providing an immutable ledger, while encryption and a structured endorsement process safeguard patient confidentiality. This approach not only streamlines data handling across different departments but also builds a secure and reliable system that fosters trust and efficiency in patient care.

**10.2 REFERENCES:**

1. M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain Technologies for the Internet of Things: Research Issues and Challenges," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2188–2204, Apr. 2019.

2. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

3. S. B. Zanjal and G. R. Teli, "Medical Records Management Using Blockchain Technology," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 1135–1139, 2019.

4. X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control," Journal of Medical Systems, vol. 40, no. 10, Oct. 2016.

5. M. Mettler, "Blockchain technology in healthcare: The revolution starts here," 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), pp. 1–3, 2016.

6. A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using Blockchain for Medical Data Access and Permission Management," 2016 2nd International Conference on Open and Big Data (OBD), pp. 25–30, 2016.

7. L. Fan et al., "A Blockchain-Based Privacy-Preserving Scheme for Large-Scale Health Data," IEEE Access, vol. 8, pp. 101073–101086, 2020.

8. S. P. R. K. Chaurasia, "Blockchain for Healthcare Data Security: State of the Art and Future Trends," International Journal of Medical Informatics, vol. 166, pp. 104876, 2023.

9. N. Kumar, V. N. S. S. R. S. S. Reddy, and R. S. Chouhan, "Recent Advances in Blockchain-Based Healthcare Systems and Data Privacy," Future Generation Computer Systems, vol. 130, pp. 307–325, 2023.

10. D. O. Adeyemo and A. A. Alaba, "Blockchain-Based Approaches for Secure Health Data Management," Journal of Healthcare Engineering, vol. 2023, Article ID 7261990, 2023.