PRIVACY PRESERVING BLOCKCHAIN BASED SYSTEM FOR DISEASES MANAGEMENT

A PROJECT REPORT

Submitted by

HARSHINI B [REGISTER NO:211419104098]

HEMAVARSHINI D [REGISTER NO:211419104103]

NANDHINI G [REGISTER NO:211419104174]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"PRIVACY PRESERVING BLOCKCHAIN BASED SYSTEM FOR DISEASES MANAGEMENT"** is the bonafide work of "**HARSHINI.B (211419104098), HEMAVARSHINI.D (211419104103) and NANDHINI.G (211419104174)"** who carried out the project work under Mrs. R. DEVI, M.E., supervision.

SIGNATURE

Dr.L.JABASHEELA, M.E., Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mrs.R.DEVI, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR (G-1)

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENT

We HARSHINI.B (211419104098), HEMAVARSHINI.D (211419104103) and NANDHINI.G (211419104174) hereby declare that this project report titled “PRIVACY PRESERVING BLOCKCHAIN BASED SYSTEM FOR DISEASES MANAGEMENT”, under the guidance of Mrs. R. DEVI, M.E., is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

HARSHINI.B

HEMAVARSHIN.D

NANDHINI.G

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR, M.E., Ph.D.,** and **Dr.SARANYASREE SAKTHI KUMAR B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.,** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr.L. JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my project guide **Mrs. R. DEVI, M.E.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

HARSHINI.B (211419104004),
HEMAVARSHINI.D (211419104129),
NANDHINI.G (211419104189)

ABSTRACT

The shortage of people, institutions, and pharmaceuticals in public health systems is one of the key problems affecting the healthcare industry in many developing nations. Information and communication technology has demonstrated its capacity to raise data security, lower costs, and improve medical quality during the past decade. These technologies can be used by developing nations to enhance healthcare services and upkeep. Patient's medical information is frequently provided using the Internet of Things. Yet, there are more security and privacy vulnerabilities with this technology. We describe a Blockchain-based Health Information Exchange (HIE) in our system. We suggest using Blockchain technology to improve security and privacy in healthcare systems in addition to data encryption. In comparison to existing methods, data are stored in an Inter Planetary File System (IPFS) in off-chain database. To ensure blockchain scalability, re-encryption proxies with the aid of Ethereum blockchain technology are employed based on Proof of Authority (PoA) to speed up the data storage. Unfortunately, Inter Planetary File System (IPFS) lacks strong economic incentives and is untrustworthy with regard to private data. However, security concerns are solved by combining the Advanced Encryption Standard (AES) and Secure Hash Algorithm 256-bit (SHA-256) algorithms with Blockchain to encrypt data and restrict access to it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

SHORTCUT	ABBREVATION	PAGE NO.
HIE	Health Information Exchange	2
IPFS	Inter Planetary File System	2
PoA	Proof of Authority	2
AES	Advanced Encryption Standard	2
SHA-256	Secure Hash Algorithm 256-bit	2
EHR	Electronic Health Record	5
HTML	Hypertext Markup Language	16
CSS	Cascading Style Sheets	16
JSP	Jakarta Server Pages	16
SQL	Structured Query Language	16

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 PROBLEM DEFINITION

Healthcare sector in many developing countries is facing several challenges due to cultural, cost, and economic conditions. These challenges include the shortage of people, institutions, and pharmaceuticals in public health systems is one of the key problems affecting the healthcare industry in many developing nations. Information and communication technology has demonstrated its capacity to raise data security, lower costs, and improve medical quality during the past decade. These technologies can be used by developing nations to enhance healthcare services and upkeep. Patient's medical information is frequently provided using the Internet of Things. Yet, there are more security and privacy vulnerabilities with this technology. We describe a Blockchain-based Health Information Exchange (HIE) in our system. The most remarkable feature about Blockchain is how much it lowers the severity of a data breach. There are numerous shared copies of the same data base in Blockchain, in contrast to traditional methods, which makes it more difficult to carry out a data breach attack or cyber-attack. With all of its anti-fraud capabilities, block chain technology has the potential to change a number of business sectors, including the healthcare industry, and make procedures smarter, more secure, transparent, and more efficient compared to the traditional processes. We suggest using Blockchain technology to improve security and privacy in healthcare systems in addition to data encryption. In comparison to existing methods, data are stored in an Inter Planetary File System (IPFS) in off-chain database. To ensure blockchain scalability, re-encryption proxies with the aid of Ethereum blockchain technology are employed based on Proof of Authority (PoA) to speed up the data storage. Unfortunately, Inter Planetary File System (IPFS) lacks strong economic incentives and is untrustworthy with regard to private data. However, security

concerns are solved by combining the Advanced Encryption Standard (AES) and Secure Hash Algorithm 256-bit (SHA-256) algorithms with Blockchain to encrypt data and restrict access to it. The benefits of Advanced Encryption Standard (AES) algorithm**,** it uses higher length key sizes such as 128, 192 and 256 bits for encryption. Hence it makes AES algorithm more robust security protocol against hacking. It is most common security protocol used for wide variety of applications such as wireless communication, financial transactions, e-business, encrypted data storage etc. For 128 bits, about 2^{128} attempts are needed to break which makes it very difficult to hack in other words no one can hack the information. For the security concerns Secure Hash Algorithm 256-bit (SHA-256) is employed which obtain the leading position for the most secured algorithm and it's used for cryptographic security. Cryptographic hash algorithms produce irreversible and unique hashes where no two input values can produce same hash output. The larger the number of possible hashes, the smaller the chance that two values will create the same hash.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 Access Control and Privacy-Preserving Blockchain-Based System for Diseases Management

AUTHOR: Kebira Azbeg, Ouail Ouchetto, Said Jai Andaloussi

YEAR: 2022

In many developing countries, the healthcare sector is facing several challenges, mainly due to the lack of personal, institutions, and medications in public health systems. But the information and communication technology has proved its ability to improve medical quality, promote data security. In addition to data encryption, we propose to use Blockchain technology to enhance security and privacy in healthcare systems. To ensure Blockchain scalability, data are stored in an Inter Planetary file system (IPFS) off-chain database. We use an Ethereum Blockchain based on proof of authority (PoA) to speed up the data storage.

ADVANTAGES:

It is a protocol, hypermedia, file sharing Peer to Peer network for storing and sharing data.

DISADVANTAGES:

It is not fit to store large amount of data and not have the incentive layers to verify the integrity of the data that it stores.

2.2 Using Blockchain for Electronic Health Records

AUTHOR: Ayesha Shahnaz, Usman Qamar, Ayesha Khalid

YEAR: 2019

The Electronic Health Record (EHR) systems face problems regarding data security, integrity and management. It discusses how the blockchain technology can be used to transform the EHR systems and could be a solution of these issues. The aim of this proposed framework is firstly to implement blockchain technology for EHR and secondly to provide secure storage of electronic records by defining granular access rules for the users of the proposed framework. Moreover, this framework also discusses the blockchain technology in general via use of off-chain storage of the records and provides the EHR system with the benefits of having a scalable, secure and integral blockchain-based solution.

ADVANTAGES:

It allows the blockchain to validate and confirm transactions and operations, without the need for a third-party intermediary.

DISADVANTAGES:

The proof of work (PoW) is costly and require a plenty of computing time and it is energy intensive.

2.3 A Patient-Centric Health Information Exchange Framework Using Blockchain Technology

AUTHOR: Yan Zhuang, Lincoln R. Sheets, Yin-Wu Chen, Zon-Yin Shae, Jeffrey J.P. Tsai, Chi-Ren Shyu

YEAR: 2020

The Office of the National Coordinator (ONC) for Health Information Technology is seeking patient-centric HIE designs that shift data ownership from providers to patients. After investigating the current workflow of HIE, this paper provides a feasible solution to these challenges by utilizing the unique features of blockchain, a distributed ledger technology which is considered “un-hackable”.

By personalizing data segmentation and an “allowed list” for clinicians to access their data, this design achieves patient-centric HIE

ADVANTAGES:

It includes decentralization and easier auditability, cost reductions, and automation.

DISADVANTAGES:

The inefficiency in a distributed ledger technology, there is a possibility of 51% of attack.

2.4 A Blockchain-Based Approach for Drug Traceability in Healthcare Supply Chain

AUTHOR: Ahmad Musamih, Khaled Salah, Raja Jayaraman, Junaid Arshad, Mazin Debe, Yousof Al-Hammadi, Samer Ellahham

YEAR: 2021

Healthcare supply chains are complex structures spanning across multiple organizational and geographical boundaries, providing critical backbone to services vital for everyday life. The inherent complexity of such systems can introduce impurities including inaccurate information, lack of transparency and limited data provenance. In this article, we present an Ethereum blockchain-based approach leveraging smart contracts and decentralized off-chain storage for efficient product traceability in the healthcare supply chain.

ADVANTAGES:

The Ethereum technology is decentralized, interoperability and open-source permissioned network.

DISADVANTAGES:

It uses a complicated programming language and investing can be risky.

2.5 Lightweight Blockchain for Healthcare

AUTHOR: Leila Ismail, Huned Materwal, sherali Zeadally

YEAR: 2019

Healthcare data management has been gaining a lot of attention in recent years because of its high potential to provide more accurate and cost-efficient patient care. The traditional client-server and cloud-based healthcare data management systems suffer from the issues of single point of failure, data privacy, centralized data stewardship, and system vulnerability. The replication mechanism, and privacy and security features of blockchain have a promising future in the healthcare domain as they can solve some of the inherent issues of the health management system.

ADVANTAGES:

It generates 11 times lower network traffic compared to the Bitcoin network as the number of blocks increases and shows a speedup of 67% in ledger.

DISADVANTAGES:

Since this proposed architecture does not fork the transactions on all the nodes as in Bitcoin and only the cluster heads maintain a copy of the ledger.

2.6 A Blockchain-Based Medical Data Sharing and Protection Scheme

AUTHOR: Leila Ismail, Huned Materwal, sherali Zeadally

YEAR: 2019

This proposes a medical data sharing and protection scheme based on the hospital's private blockchain to improve the electronic health system of the hospital. Firstly, the scheme can satisfy various security properties such as decentralization, openness, and tamper resistance. A reliable mechanism is created for the doctors to store medical data or access the historical data of patients while meeting privacy preservation. It allows patients who get the same symptoms to conduct mutual authentication and create a session key for their future communication about the illness. The proposed scheme is implemented by using PBC and OpenSSL libraries.

ADVANTAGES:

Identity-based encryption allows the sender of a message to encrypt it without the need for the recipient's public key to have been sent or certified earlier.

DISDAVNTAGES:

There is a chance to be hacked by the unknown user by knowing the users' details.

2.7 A Survey on Blockchain-Based Self-Sovereign Patient Identity in Healthcare

AUTHOR: Bahar Houtan, Abdelhakim Senhaji Hafid, Dimitrios Makrakis

YEAR: 2020

Distributed Ledger Technology (DLT) is a novel method which would allow to securely record time-stamped data and enable patient-driven health and identity records. In this paper, we review the state-of-the-art in Blockchain (BC)-based self-sovereignty and patient data records in healthcare. Our motivation is to investigate the potential of BC technology for use in the patient data and identity management. As a distributed decentralized technology, BC can be very beneficial, giving patients control over their own data and self-sovereign

identity. Electronic Health Records (EHR) and Patient Health Records (PHR) are used to record patient data, such as the doctor's notes upon a visit and radiology images. Hence, they include critical information regarding patient's privacy and identity.

ADVANTAGES:

It includes decentralization and easier auditability, cost reductions, and automation.

DISADVANTAGES:

The inefficiency in a distributed ledger technology, there is a possibility of 51% of attack.

2.8 Healthcare Applications Using Blockchain Technology: Motivations and Challenges

AUTHOR: Sadia Ramzan, Aqsa Aqdus, Vinayakumar Ravi, Deepika Koundal, Rashid Amin, Mohammed A. Al Ghamdi

YEAR:2020

In this, they provide a review of blockchain technology in healthcare. We present a detailed introduction, history, technical information, and types of blockchain technology. Motivations behind this technology and top healthcare projects completed using this technology are also discussed. This article is classified into three groups based on blockchain applications with their use cases. The evaluation of medical care technologies and relevant applications based on blockchain technology, such as sharing electronic medical records, remote patient monitoring, and supply chain management, are also discussed. In revolutionizing the healthcare industry, we illustrate the potential of blockchain technology. We have also focused on identifying the limitations of previous approaches.

ADVANTAGES:

It has better collaboration with suppliers, better quality control, improved risk mitigation and more agile business

DISADVANTAGES:

It is expensive to implement, complicated programming language, requires trained and personalized staff, lack if reliability and co-ordination.

2.9 A Secure System for Pervasive Social Network-Based Healthcare

AUTHOR: Jie Zhang, Nian Xue, Xin Huang

YEAR: 2016

This paper proposes a secure system for PSN-based healthcare. Two protocols are designed for the system. The first one is an improved version of the IEEE 802.15.6 display authenticated association. It establishes secure links with unbalanced computational requirements for mobile devices and resource-limited sensor nodes. The second protocol uses blockchain technique to share health data among PSN nodes. We realize a protocol suite to study protocol runtime and other factors. In addition, human body channels are proposed for PSN nodes in some use cases. The proposed system illustrates a potential method of using blockchain for PSN-based applications.

ADVANTAGES:

Policy Service Node is a node that handles traffic between network devices and ISE.

DISADVANTAGES:

Policy Service Node does not provide full access to administration GUI.

2.10 Application of Hyperledger In The Hospital Information Systems: A Survey

AUTHOR: Zeqi Leng, Zhenjiang Tan, Kunhao Wang

YEAR: 2021

Hyperledger is considered to be the most mature consortium chain technology. Compared with other blockchain technologies, Hyperledger focuses on the application development of enterprise-level standards. Due to its unique permission management, fine-grained access control, pluggable consensus algorithm and higher transaction performance, Hyperledger has been increasingly used in the hospital information systems.

ADVANTAGES:

This version's key advantages are simplified data sharing, upgraded smart contract technology, and speedier transactions. It has a predefined community of participants, and access to the network is restricted only to them.

DISADVANTAGES:

It does not have much-skilled programmers, lack of use cases, got a complex architecture with minimum APIs and SDKs and not a Network fault-tolerant.

CHAPTER 3

SYSTEM ANALYSIS

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In existing system, to ensure the security and reliability of sharing patient's medical information, the Inter Planetary File System (IPFS) is employed which is a hypermedia protocol for sharing files in Peer-to-Peer network with high performance. Unfortunately, the Inter Planetary File System (IPFS) lacks strong economic incentives and is untrustworthy with regard to private data as well as consumes a lot of bandwidth which is not appreciated by metered internet users with limited storage and not guarantee permanence. To encrypt and decrypt the data, Proxy re-encryption is used which is a special type of public-key encryption that permits a proxy to transform ciphertexts from one public key to another, without the proxy being able to learn any information about the original message. These two techniques are combined with the Ethereum Blockchain technology, a large and committed global community and the largest ecosystem in blockchain and cryptocurrency. However, Ethereum is extremely volatile. It uses a complicated programming language and investing in Ethereum can be risky.

3.2 PROPOSED SYSTEM

The proposed system has overcome all the disadvantages listed in the existing system with the help of algorithms in the blockchain to ensure more security and privacy while sharing the information. So, the security concerns are solved by combining the Advanced Encryption Standard (AES) and Secure Hash Algorithm 256-bit (SHA-256) algorithms with Blockchain to encrypt data and restrict access to it. The benefits of Advanced Encryption Standard (AES) algorithm**,** It uses higher length key sizes such as 128, 192 and 256 bits for encryption. Hence it makes AES algorithm more robust security protocol against hacking. It is most common security protocol used for wide variety of

applications such as wireless communication, financial transactions, e-business, encrypted data storage etc. For 128 bits, about 2^{128} attempts are needed to break which makes it very difficult to hack in other words no one can hack the information. For the security concerns Secure Hash Algorithm 256-bit (SHA-256) is employed which obtain the leading position for the most secured algorithm and it's used for cryptographic security. Cryptographic hash algorithms produce irreversible and unique hashes where no two input values can produce same hash output. The larger the number of possible hashes, the smaller the chance that two values will create the same hash.

ADVANTAGES OF PROPOSED SYSTEM:

- It gives standard and valid solution to process the data with the hash function.
- Data is encrypted by using Advanced Encryption Standard (AES) algorithm.
- To ensure the security concerns about sharing the information, (SHA-256) Secure Hash Algorithm 256-bit algorithm is employed.

3.3. 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:

- Economic feasibility
- Technical feasibility
- Operational feasibility

3.3.1. ECONOMIC FEASIBILITY:

Here, we track down the absolute expense and advantage of the proposed framework over current framework. For this venture, the primary expense is administration cost. This fills in as a straightforward web application which is cost productive. Additionally, it is basic in activity and doesn't cost preparing or fixes.

3.3.2. TECHNICAL FEASIBILITY:

It incorporates figuring out advancements for the undertaking, both equipment and programming. Here, the base equipment prerequisite is 2GB RAM and Core i5 Processor and the product necessities incorporate Windows OS, MySQL and Eclipse IDE. The backend innovation utilized is JAVA. Since, it is stage free and can be utilized in assortment of uses.

3.3.3 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.

3.4. HARDWARE REQUIREMENTS:

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system

System	Core i5 Processor
Hard Disk	250 GB
Monitor	15'' LED
Input Devices	Keyboard, Mouse
Ram	2 GB

Table 3.1 Hardware Requirement

3.5. SOFTWARE REQUIREMENTS:

The software requirements are the specification of the system. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

Operating System	Windows 10
Language	Java
Ide	Eclipse
Front End	HTML, CSS, JSP
Back End	My SQL 5.5

Table 3.1 Software Requirement

CHAPTER 4

SYSTEM DESIGN

CHAPTER 4

SYSTEM DESIGN

4.1. ER DIAGRAM

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

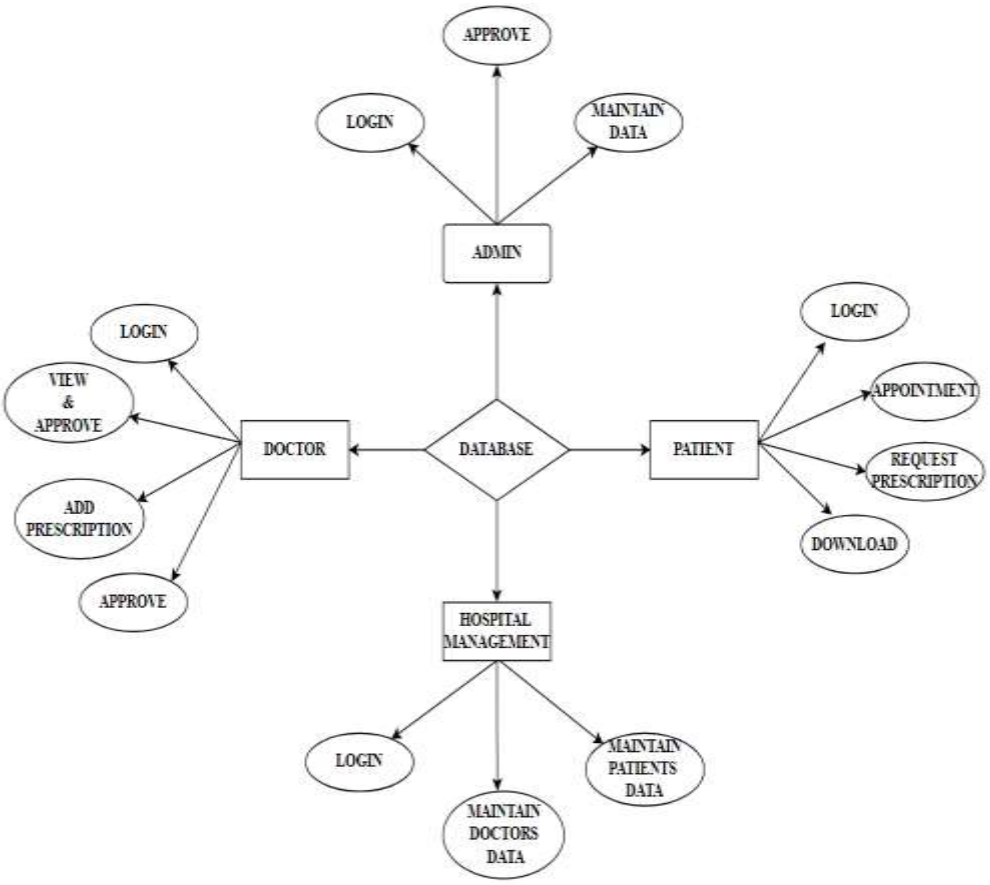


Fig 4.1. ER DIAGRAM

4.2 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a visual representation of the information flow through a process or system. DFDs help you better understand process or system operation to discover potential problems, improve efficiency, and develop better processes. Data Flow Diagram symbols are standardized notations, like rectangles, circles, arrows, and short-text labels, that describe a system or process' data flow direction,data inputs, data outputs, data storage points, and its various sub-processes.

Here each level goes deeper in representing the flow of the system starting from theHigh-level data flow diagram to low level data flow diagram.

4.2.1 LEVEL 0

Fig 4.2. DFD LEVEL 0

4.2.2 LEVEL 1

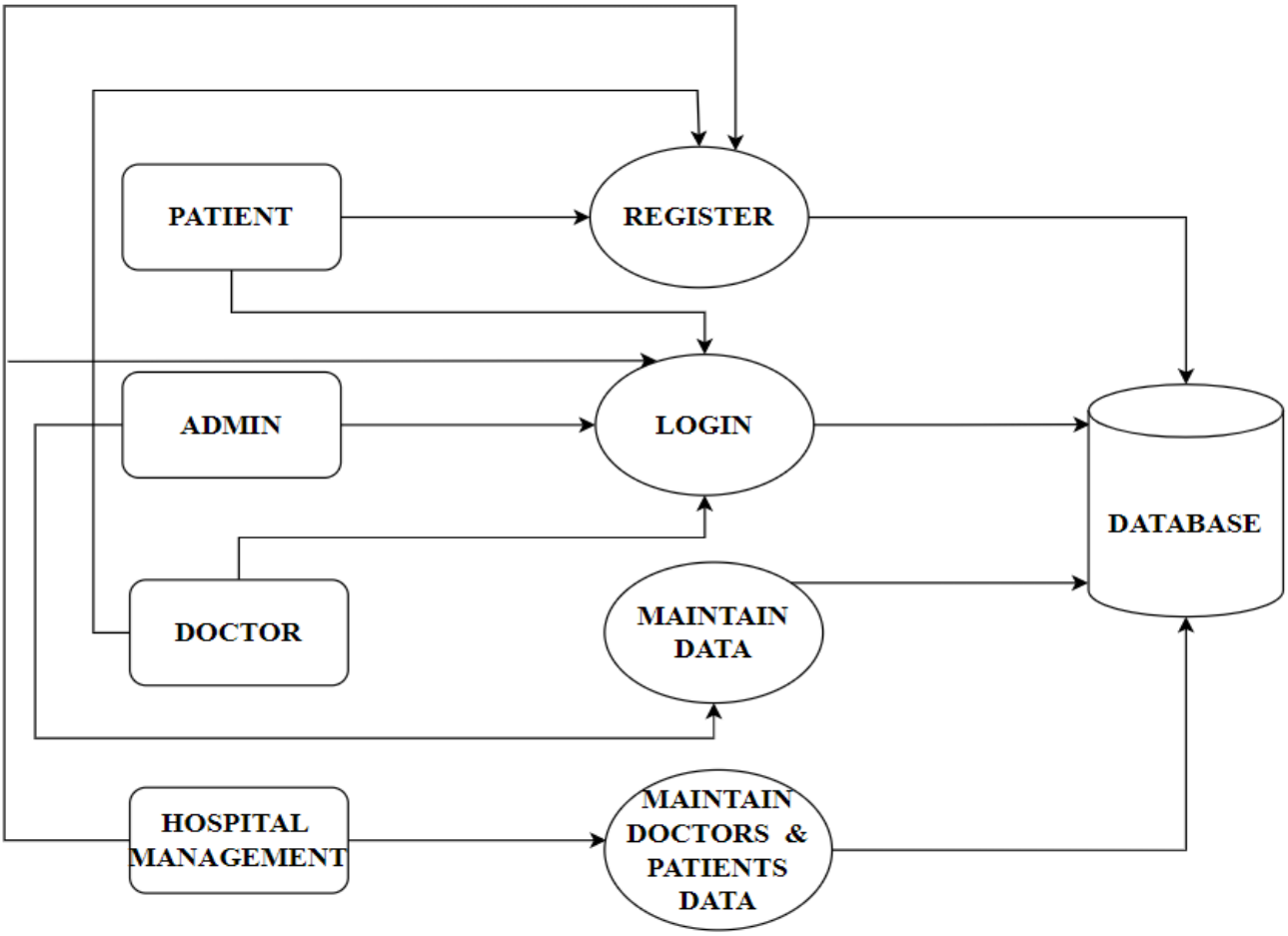


Fig 4.3. DFD LEVEL 1

4.2.3 LEVEL 2 (DOCTOR)

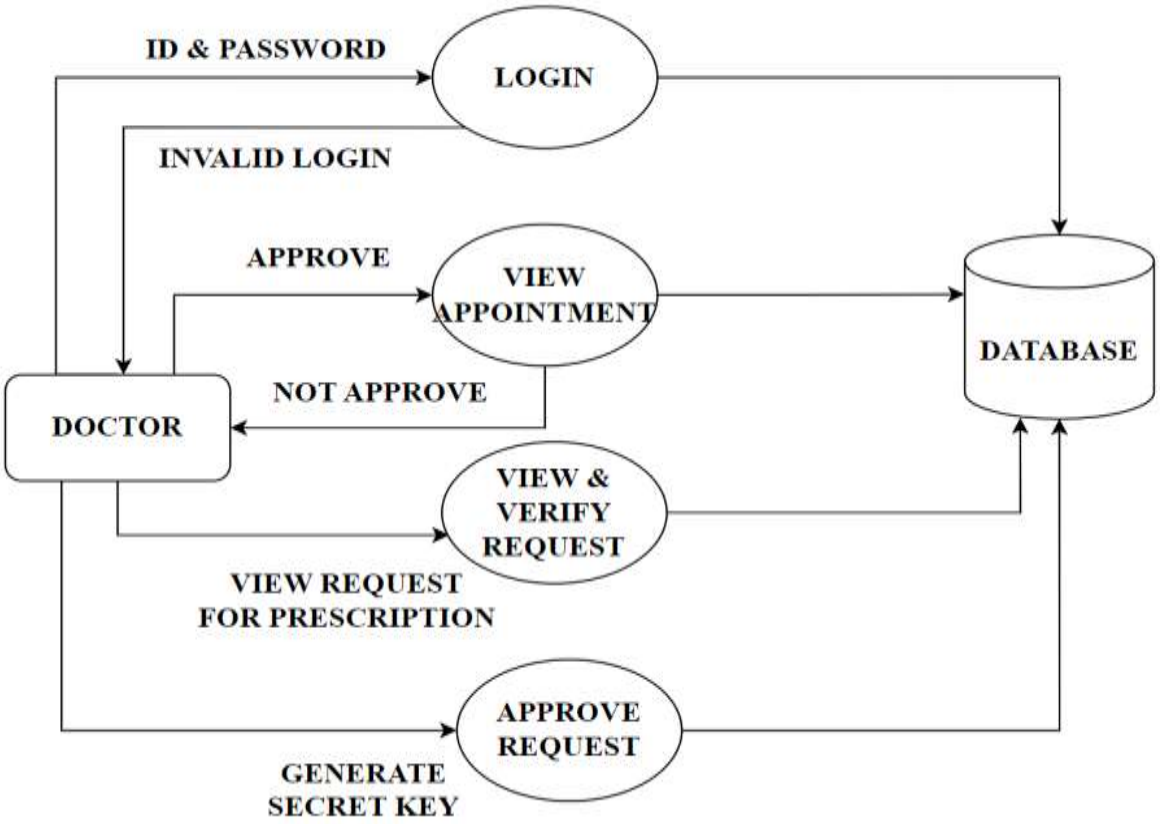


FIG 4.4 DFD LEVEL 2 (DOCTOR)

4.2.4 LEVEL 2 (PATIENT)

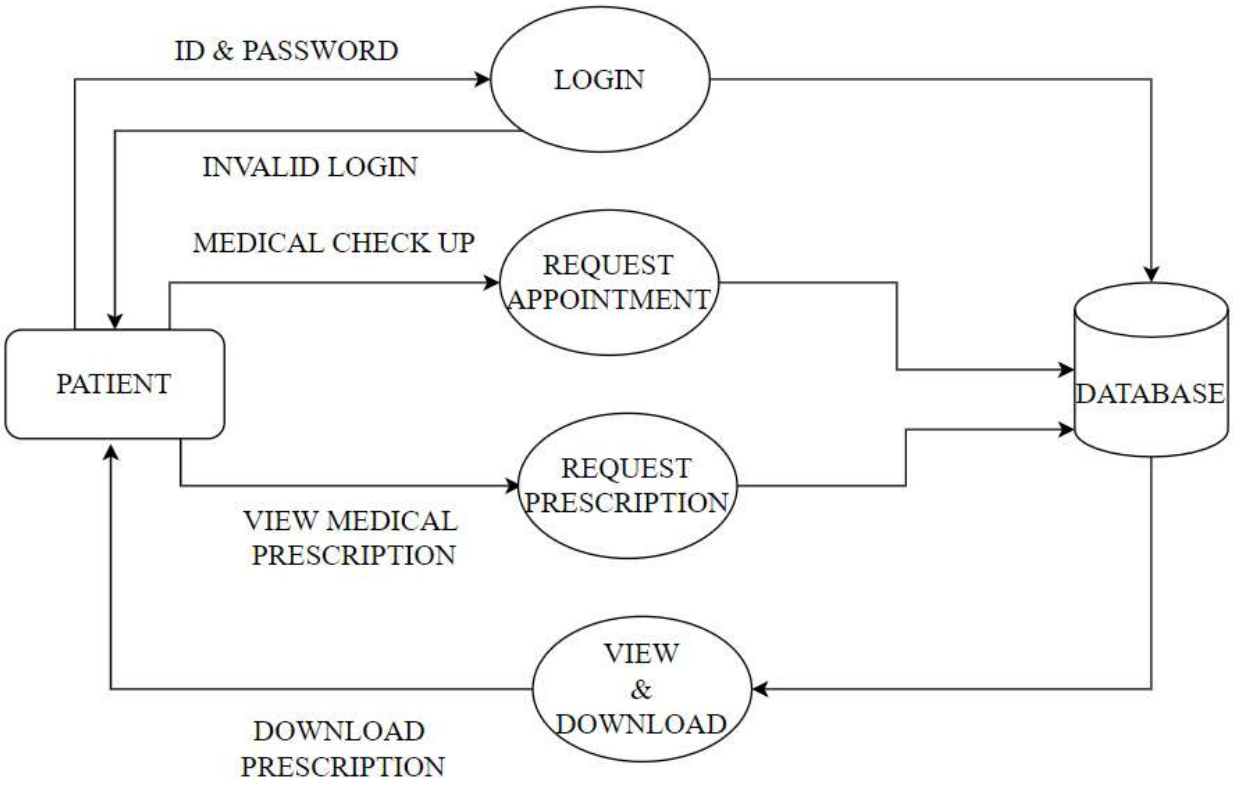


FIG 4.5. DFD LEVEL 2 (PATIENT)

4.3 UML DIAGRAMS

4.3.1 USECASE DIAGRAM

Use case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

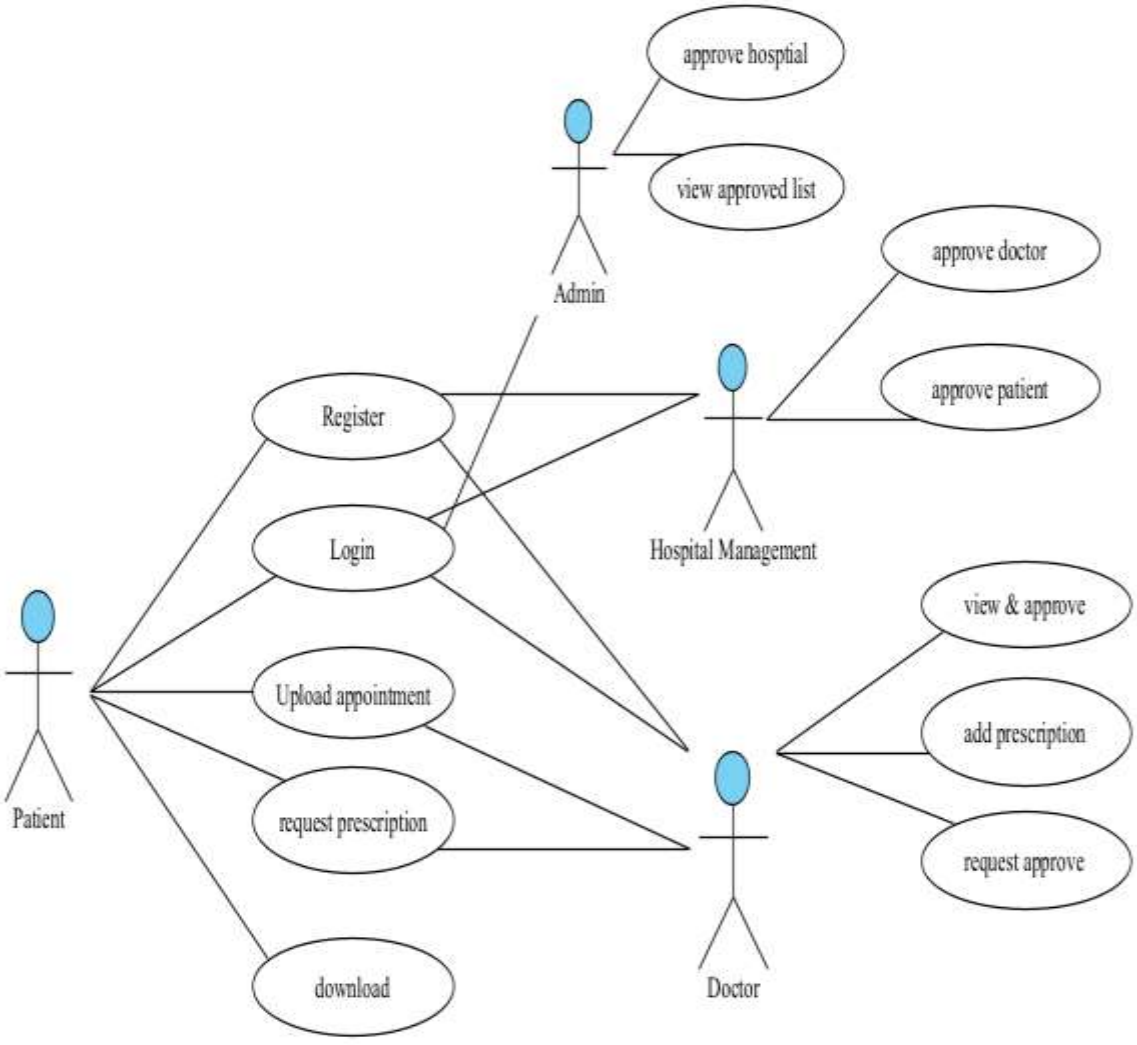


FIG 4.6. USECASE DIAGRAM

4.3.2 CLASS DIAGRAM

A class diagram is a visual representation of class objects in a model system, categorized by class types. Each class type is represented as a rectangle with three compartments for the class name, attributes, and operations. Objects are represented as ovals that contain class names inside class name compartments.

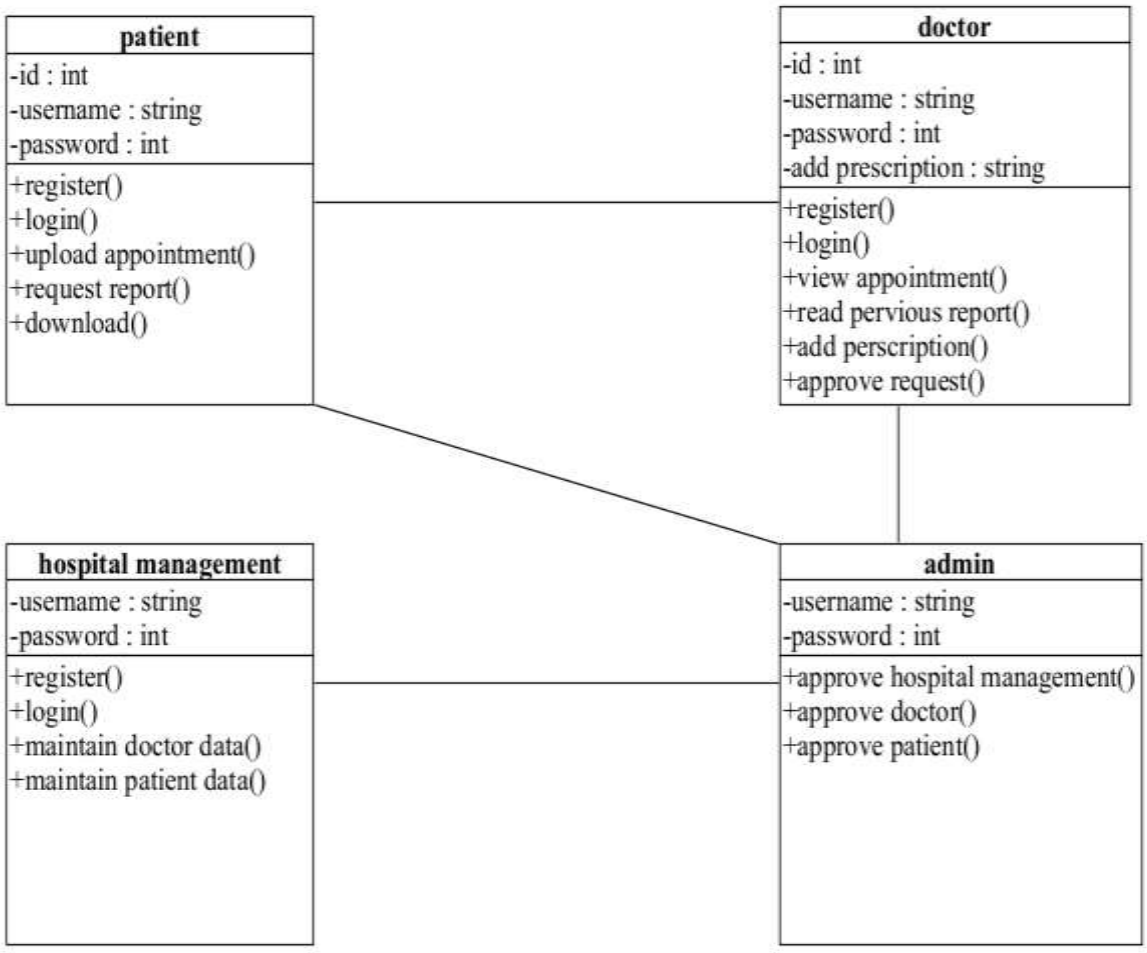


FIG 4.7. CLASS DIAGRAM

4.3.3 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality.

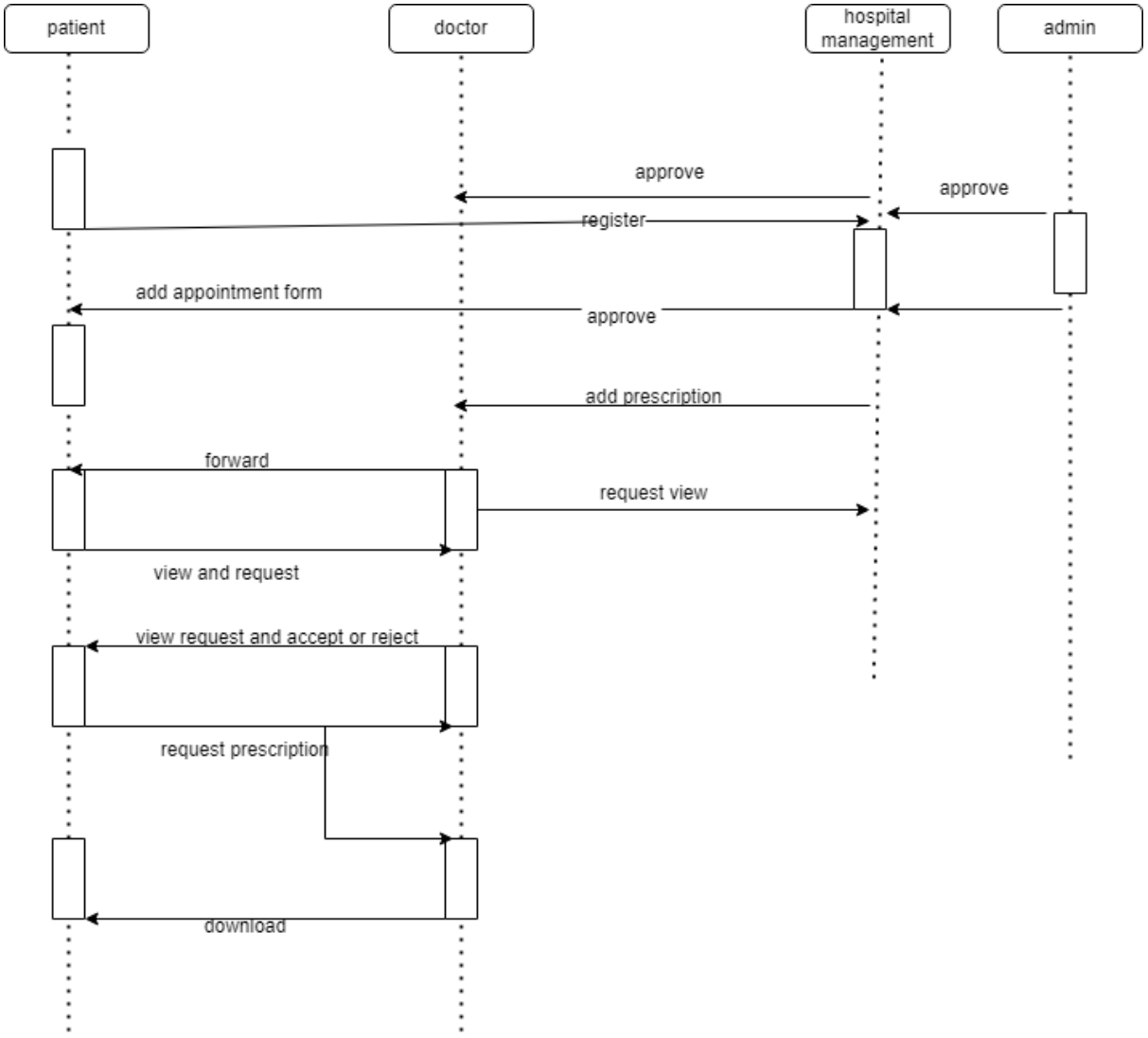


FIG 4.8. SEQUENCE DIAGRAM

4.3.4 ACTIVITY DIAGRAM

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.

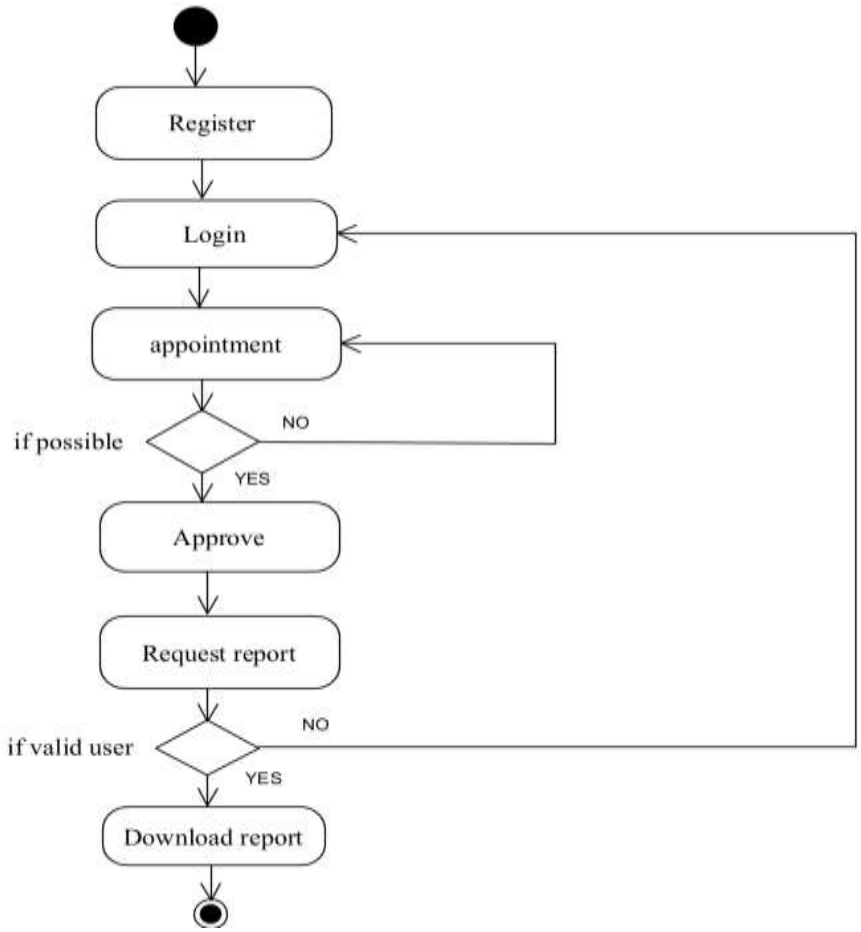


FIG 4.9. ACTIVITY DIAGRAM

CHAPTER 5

SYSTEM ARCHITECTURE

CHAPTER 5
SYSTEM ARCHITECTURE

5.1 ARCHITECTURE OVERVIEW

The project's user interface will be improved in this chapter. The design of a website page that a user can interact with is the very first point of interaction with the application through the programme. The development of the user interface uses front-end technologies. Every project needs a database since it controls all storage-related data and references. Additionally, this project maintains and handles data pertaining to hospital administrations, doctors, and patients. Because of this, the architecture of the databases calls for the storing of information starting with the patient appointment form and ending with the process of downloading the patient's medication. The modules developed in the application are hospital management, doctor, patient and the admin. The admin verifies the hospital's registration and licenses prior approving the hospital management's registration. The hospital management verifies and authorizes the doctor's registration. The patient can fill out the appointment form for the consultation or diagnosis after registration. The doctor will then choose whether to approve or reject the appointment form. Once the appointment has been confirmed, the patient can visit the doctor for a consultation or a diagnosis. The doctor also can upload the patient's prescription and view any prior prescriptions or reports pertaining to the diagnosis as well as the patient's past medical history. The prescription can then be downloaded by the patient upon request to the doctor, who must first approve the request before the prescription can be accessed. This prescription data is encrypted and decrypted, as well as stored in a database.

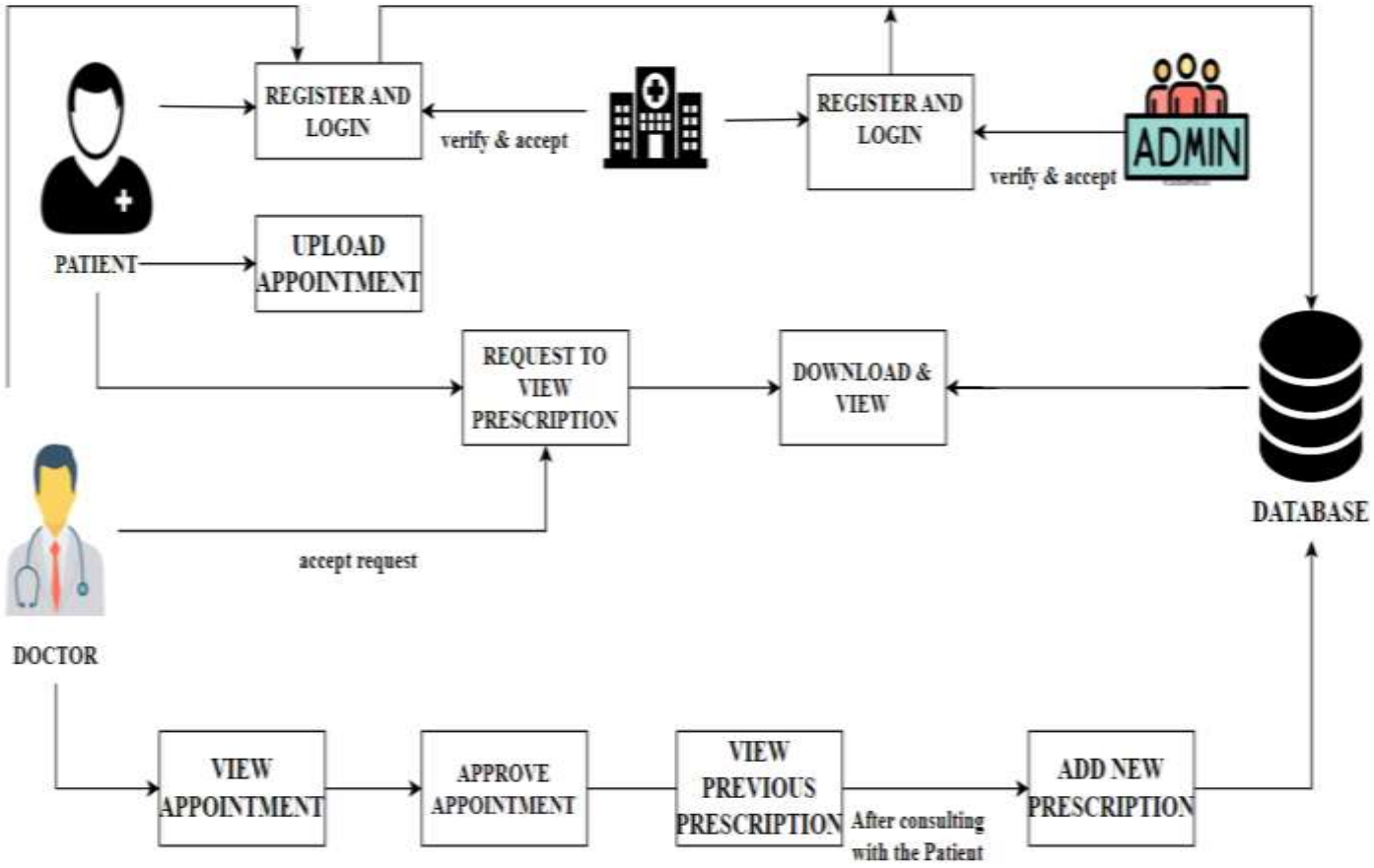


FIG 5.1. SYSTEM ARCHITECTURE

5.2 MODULE DESIGN SPECIFICATION

- Admin
- Hospital management
- Doctor
- Patient

ADMIN:

In this module, they confirm and approve the request of hospital management after verifying their hospital Registration and License Required of that particular hospital.

HOSPITAL MANAGEMENT:

In this module, the hospital management verifies and approve the doctor's request and then they maintain the data of their respective doctor and their performances.

DOCTOR:

In this module, the doctor views the appointment form which have been uploaded by the patient and he either approve the application or not, and then proceed further process. The doctor adds the prescription of patient after the diagnosis and he can view the previous report or prescription of that particular patient and he cannot duplicate the patient's previous prescription

PATIENT:

In this module, the user uploads the appointment form of particular hospital. If the doctor accepts the appointment form, then the patient can visit the doctor for diagnosis. The patient can view their prescription by requesting the prescription report to the doctor. If the doctor accepts the request and then the patient can download the report or the prescription.

CHAPTER 6

SYSTEM IMPLEMENTATION

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 ALGORITHMS

6.1.1 AES (Advanced Encryption Standard):

The AES Encryption algorithm, also referred to as the Rijndael algorithm, is a symmetric block cypher algorithm with a 128-bit block/chunk size. It uses keys of 128, 192, and 256 bits to convert each of these discrete blocks. In order to create the cipher text, it encrypts the blocks and combines them. A substitution-permutation network, or SP network, is the foundation of this system. It involves a number of interconnected processes, some of which include bit shuffles and others which involve substitutions, which are the act of changing inputs into particular outputs (permutations). When sending files in an encrypted manner between colleagues, AES is employed. Blocks of 128 bits each are used for AES encryption of data. Based on the length of the key, the number of rounds will vary as follows:

1. For 128bit key there are 10 rounds
2. For 192bit key there are 12 rounds
3. For 256bit key there are 14 rounds

a. ENCRYPTION:

Each round comprises of 4 steps:

1. Sub Bytes
2. Shift Rows
3. Mix Columns
4. Add Round Key

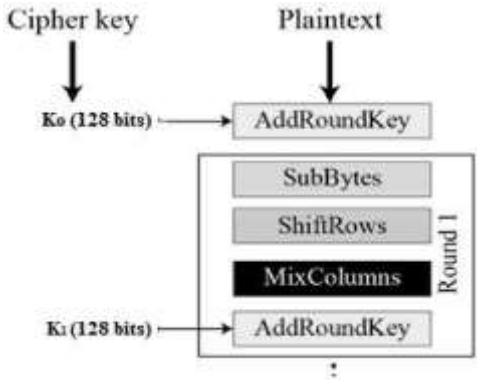


Fig 6.1. Encryption Process

SUB BYTES:

At this stage, one byte is replaced with another. It is carried out utilizing a lookup table also known as the S-box. The state array's byte a(i,j) is replaced with a Sub Byte S(a(i,j)) in this Sub Byte step. In order to replace the sixteen input bytes, a constant table (S-box) provided in the design is looked up. The resultant is in a form of matrix with 4 rows and 4 columns.

SHIFT ROWS:

The matrix's four rows are all moved to the left. The right side of the row is used to re-insert any entries that "slip off." This is how shift is done:

1. The top row is not moved.
2. The second row has been moved left one (byte) place.
3. The third row has been moved two spaces to the left.
4. The fourth row has been moved three spaces to the left.

The outcome is a new matrix made up of the same 16 bytes but with their positions altered.

MIXCOLUMNS:

A unique mathematical function is used to change each column of four bytes. The four bytes of one column are entered by this function, which returns four entirely new bytes that replace the original column. The outcome would be another new matrix with 16 more bytes. It must be noted that the final round does not include this phase.

ADDROUNDKEY:

The 16 bytes of the matrix are now addressed as 128 bits, and they are XORed (XOR Operation) now with 128 bits of the round key. The cipher text will be produced if this is the final round. If not, the 128 bits that are produced are translated into 16 bytes, and the process starts over again.

b. DECRYPTION:

An AES cipher text's decryption procedure is quite identical to its encryption procedure in reverse. The four stages are carried out in reverse order with each round.

1. Add round key
2. Mix columns
3. Shift rows
4. Byte substitution

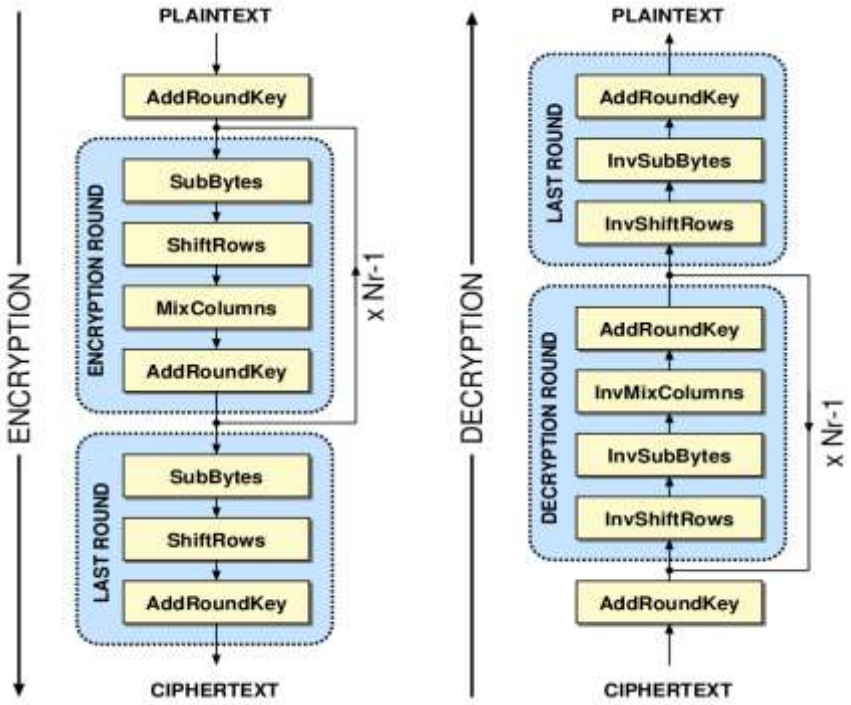


Fig 6.2. AES Architecture

6.1.2 SHA-256 (Secure Hash Algorithm – 256):

Secure Hash Algorithm, or SHA, is the name of the family of algorithms that includes SHA 256. The 256 in the name indicates that no matter how large the plaintext or clear text is, the hash result will always be 256 bits. During the execution of the SHA-256 algorithm, a piece of data is transmitted via a function that applies mathematical operations to the plaintext. The hash function produces the hash value/digest, which is referred to as its output. In contrast to certain other well-known hashing algorithms, SHA-256 is secure and has not been "broken," which is the main reason why technology employs it. Also, it has no known flaws that would make it unstable. The hash value is always 256 bits no matter how big the clear text or plaintext is. The number 256, which is used in the name to denote the very last hash digest value, has this meaning.

HASHING:

A hash value, also known as a hash function, is produced by a mathematical operation and an arbitrary-sized file, such as an email, document, picture, or other type of data. As a one-way function that is specific to the file being hashed, a derived hash cannot be reversed to find more files that would provide the same hash result.

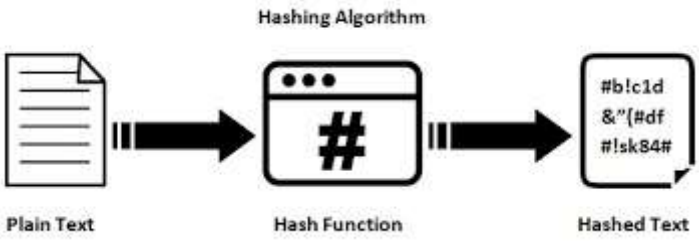


Fig 6.3. Hashing

CHARACTERISTICS OF SHA-256 ALGORITHM:

- Message Length: The clear text's length should be no more than 264 bits. To maintain the highest level of randomness in the digest, the size has to fit the comparison area.
- Digest Length: The hash digest for the SHA-256 algorithm must be 256 bits long, for the SHA-512 algorithm 512 bits, etc. Huge digests often represent a large increase in calculations at the expense of storage and efficiency.
- Irreversible: Every hash function, including SHA 256, is irreversible by design. If you have created the digest already, neither you nor the digest should receive the original cost when you run it by the hash function once more.

STEPS IN SHA-256 ALGORITHM:

1. PADDING BITS:

In order to make our original message's length match the minimum length needed for the hash function, the first step of the hashing function is attaching bits to it. The message is given a few more bits, leaving the length exactly 64 bits shy of a factor of 512. The initial bit of the addition should be a one, and the following bits should all be zeros.

M + P + 64 = n x 512

(i.e.) M = length of original message

P = padded bits

n = constant

1. PADDING LENGTH:

The resultant plaintext is then multiplied by 64 bits to make it a factor of 512. Using the modulus of the original clear text without the padding, the 64 bits of characters are calculated.

2. INITIALIZE BUFFER:

A 64-key array (K [0] to K [63]) with eight buffers each holding a default value is initialized.

3. COMPRESSION FUNCTION:

The algorithm divides the whole message into many blocks, each of 512 bits. The output of each block serves as the input for the subsequent block during each block's 64 rounds of execution. As a result, after each iteration, the block's final output is used as the input for the subsequent block. When you get to the final 512-bit block, you consider the output of the entire cycle to be the overall

hash digest. According to the name of this algorithm, this digest will have a length of 256 bits.

APPLICATION OF SHA-256:

- Digital Signature Verification
- Password Hashing
- SSL Handshake
- Integrity Checks

CHAPTER 7
PERFORMANCE ANALYSIS

CHAPTER 7

PERFORMANCE ANALYSIS

7.1. WEB TESTING

A web testing process is a well-defined and organized method that enables Quality Assurance teams to ensure fast and efficient test cases for their websites and applications. For web-based application testing or software application testing, they can rely on automation which reduces the burden of repetitive and routine tasks. It is employed to ensure Cross Browser Compatibility, Responsiveness as well as ensure Security. Broken Links in the (SEO) Search Engine Optimization as well as Cookie Testing are verified through this testing. In this testing, the following technique may be performed depending on the web testing requirements.

1. Functionality Testing of a Website:

Functionality Testing of a Website is a process that includes several testing parameters like user interface, APIs, database testing, security testing, client and server testing and basic website functionalities. It is very convenient and it allows users to perform both manual and automated testing. It is performed to test the functionalities of each feature on the website.

2. Usability testing:

Usability Testing has now become a vital part of any web-based project. It can be carried out by testers like you or a small focus group similar to the target audience of the web application.

3. Interface Testing:

Three areas to be tested in this interface testing such as Application, Web and Database Server

- Application: Test requests are sent correctly to the Database and output at the client side is displayed correctly. Errors if any must be caught by the

application and must be only shown to the administrator and not the end user.

- Web Server: Test Web server is handling all application requests without any service denial
- Database Server: Make sure queries sent to the database give expected results.

4. Database Testing:

Database is one of the critical components in web application. Testing activities will include:

- Test if any errors are shown while executing queries
- Data Integrity is maintained while creating, updating or deleting data in database.
- Check response time of queries and fine tune them if necessary.
- Test data retrieved from your database is shown accurately in your web application

5. Compatibility testing:

Compatibility tests ensures that your web application displays correctly across different devices. This would include Browser Compatibility Test.

Browser Compatibility Test:

Same website in different browsers will display differently. You need to test if your web application is being displayed correctly across browsers, JavaScript, AJAX and authentication is working fine. You may also check for Mobile Browser Compatibility.

6. Performance Testing:

This will ensure your site works under all loads. Software Testing activities will include but not limited to:

- Website application response times at different connection speeds
- Load test your web application to determine its behavior under normal and peak loads
- Test if a crash occurs due to peak load, how does the site recover from such an event
- Make sure optimization techniques like zip file compression, browser and server-side cache enabled to reduce load times

7. Security testing:

Security Testing is vital for e-commerce website that store sensitive customer information like credit cards. Testing Activities will include-

- Test unauthorized access to secure pages should not be permitted
- Restricted files should not be downloadable without appropriate access
- Check sessions are automatically killed after prolonged user inactivity
- On use of SSL certificates, website should re-direct to encrypted SSL pages.

7.2 TEST CASES & REPORTS

TESTCASE OBJECTIVES

MODULE NAME: DOCTOR

Test Case Id	Test Case Name	Input	Expected Output	Actual Output	Test Result (Pass/Fail)
TC01	Register	Username, Phone no, Email ID, Password	User should be redirected to login page (Scr 2.6)	User should be redirected to login page (Scr 2.6)	Pass
TC02	Login	Email, Password	Login Successful (Scr 2.7)	Login Successful (Scr 2.7)	Pass
TC03	Response	Verify Patient ID	Request Accepted (Scr 2.11)	Request Accepted (Scr 2.11)	Pass
TC04	Add Prescription	File type, PDF	Uploaded Successfully (Scr 2.13)	Uploaded Successfully (Scr 2.13)	Pass

MODULE NAME: PATIENT

Test Case Id	Test Cases Name	Input	Expected Output	Actual Output	Test Result (Pass/Fail)
TC01	Registration	Username, Phone number, Email ID, Password,	User should be redirected to login page (Scr 2.8)	User should be redirected to login page (Scr 2.8)	Pass
TC02	Login	Email, Password	Login Successful (Scr 2.9)	Login Successful (Scr 2.9)	Pass
TC03	Request	File	Request sent to doctor (Scr 2.14)	Request sent to doctor (Scr 2.14)	Pass
TC04	Download	File	Downloaded Successfully (Scr 2.15)	Downloaded Successfully (Scr 2.15)	Pass

MODULE NAME: ADMIN

Test Case Id	Test Cases Name	Input	Expected Output	Actual Output	Test Result (Pass/Fail)
TC01	Login	Email, Password	Login Successful (Scr 2.2)	Login Successful (Scr 2.2)	Pass
TC02	Response	Verify Hospital registration and license	Request Accepted (Scr 2.4)	Request Accepted (Scr 2.4)	Pass

MODULE NAME: HOSPITAL MANAGEMENT

Test Case Id	Test Cases Name	Input	Expected Output	Actual Output	Test Result (Pass/Fail)
TC01	Register	Email ID, Password, Hospital License	User should be redirected to login page (Scr 2.3)	User should be redirected to login page (Scr 2.3)	
TC02	Login	Email, Password	Login Successful (Scr 2.5)	Login Successful (Scr 2.5)	Pass

CHAPTER 8
CONCLUSION

CHAPTER 8

CONCLUSION

8.1CONCLUSION

Several sectors, including machine learning, banking and the military, are using the new blockchain. It also has some limitations, such as immutability in some applications where data can be modified. Despite these limitations, we must embrace and promote its widespread adoption across multiple industries. Users share a significant amount of data every day. In this project, we have provided a more secure way to share files and store them on a trusted platform like blockchain. This technology can partially eliminate all the disadvantages associated with a centralized network to share confidential health information within a hospital or between hospitals on a patient-by-patient basis. Sensitive data must be protected from unauthorized access. We explored how blockchain meets the complex security, storage, sharing and authentication requirements of data exploration.

8.2 FUTURE ENHANCEMENT

As a future work of this project, we will implement a real-world database system, Improving the efficiency of the protocol, according to the number and size of exchanged messages, and then implementing two additional algorithms.

APPENDICES

APPENDICES

APPENDICES 1. CODING

1.1 SERVER-SIDE CODING

Adminlogin.java

```
package servlet;
import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import dbcon.Connecttodata;
/**
 * Servlet implementation class Adminlogin
 */
@WebServlet("/Adminlogin")
public class Adminlogin extends HttpServlet {
	Connection con;
	private static final long serialVersionUID = 1L;
/**
* @see HttpServlet#HttpServlet()
```

```
*/
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
// TODO Auto-generated method stub
String email = req.getParameter("email");
String ps = req.getParameter("password")
int z=0;
try {
con= Connecttodata.create();
Statement st = con.createStatement();
ResultSet rs= st.executeQuery("SELECT * FROM medicalrecord.admin WHERE email= '"+email+"'and password= '"+ps+"'" );
while(rs.next())
{
z=1;
}
if(z==0) {
resp.sendRedirect("adminlogin.jsp");
} else{
resp.sendRedirect("adminloginsuccess.jsp");
}
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

AES.java

```
package servlet;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import sun.misc.*;
public class AES
{
private static String algorithm = "AES";
private static byte[] keyValue=new byte[]
{ 'A', 'S', 'e', 'c', 'u', 'r', 'e', 'S', 'e', 'c', 'r', 'e', 't', 'K', 'e', 'y' };
// Performs Encryption
public static String encrypt99(String plainText) throws Exception
{
Key key = generateKey();
Cipher chiper = Cipher.getInstance(algorithm);
chiper.init(Cipher.ENCRYPT_MODE, key);
byte[] encVal = chiper.doFinal(plainText.getBytes());
String encryptedValue = new BASE64Encoder().encode(encVal);
return encryptedValue;
}
// Performs decryption
 public static String decrypt(String encryptedText) throws Exception
{
// generate key
```

```
Key key = generateKey();
Cipher chiper = Cipher.getInstance(algorithm);
chiper.init(Cipher.DECRYPT_MODE, key);
 byte[] decordedValue = new BASE64Decoder().decodeBuffer(encryptedText);
byte[] decValue = chiper.doFinal(decordedValue);
String decryptedValue = new String(decValue);
return decryptedValue;
}
//generateKey() is used to generate a secret key for AES algorithm
private static Key generateKey() throws Exception
{
Key key = new SecretKeySpec(keyValue, algorithm);
return key;
}
}
```

AES2.java

```
package servlet;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import sun.misc.*;
public class AES2
{
```

```
private static String algorithm = "AES";
private static byte[] keyValue=new byte[]
{ 'A', 'S', 'e', 'c', 'u', 'r', 'e', 'S', 'e', 'c', 'r', 'e', 't', 'K', 'e', 'y' };
// Performs Encryption
public static String encrypt99(String plainText) throws Exception
{
Key key = generateKey();
cipher chiper = Cipher.getInstance(algorithm);
chiper.init(Cipher.ENCRYPT_MODE, key);
byte[] encVal = chiper.doFinal(plainText.getBytes());
String encryptedValue = new BASE64Encoder().encode(encVal);
return encryptedValue;
}
// Performs decryption
public static String decrypt(String encryptedText) throws Exception
{
// generate key
Key key = generateKey();
Cipher chiper = Cipher.getInstance(algorithm);
chiper.init(Cipher.DECRYPT_MODE, key);
byte[] decordedValue = new BASE64Decoder().decodeBuffer(encryptedText);
byte[] decValue = chiper.doFinal(decordedValue);
String decryptedValue = new String(decValue);
return decryptedValue;
}
//generateKey() is used to generate a secret key for AES algorithm
```

```
private static Key generateKey() throws Exception
{
Key key = new SecretKeySpec(keyValue, algorithm);
return key;
}
}
```

Appointmentreg.java

```
package servlet;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import dbcon.Connecttodata;
/**
 * Servlet implementation class Appointmentreg
 */
@WebServlet("/Appointmentreg")
public class Appointmentreg extends HttpServlet {
private static final long serialVersionUID = 1L;
```

```
/**
* @see HttpServlet#HttpServlet()
*/
public Appointmentreg() {
super();
// TODO Auto-generated constructor stub
}
/**
* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
*/
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
// TODO Auto-generated method stub
}
/**
* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
*/
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
// TODO Auto-generated method stub
String hosname = req.getParameter("hosname");
String name = req.getParameter("name");
String email = req.getParameter("email");
String gender = req.getParameter("gender");
String age = req.getParameter("age");
```

```
String patientid = req.getParameter("patientid");
String address = req.getParameter("address");
String bloodgroup = req.getParameter("bloodgroup");
String specialist = req.getParameter("specialist");
String description = req.getParameter("description");
HttpSession session = req.getSession();
session.setAttribute("emaill", email);
int z= 0;
try {
Connection con= Connecttodata.create();
PreparedStatement ps = con.prepareStatement("INSERT INTO medicalrecord.
appointmentform VALUES(id,?,?,?,?,?,?,?,?,?,?,?,?,?)");
ps.setString(1, name);
ps.setString(2, email);
ps.setString(3, hosname);
ps.setString(4, gender);
ps.setString(5, age);
ps.setString(6, patientid);
ps.setString(7, address);
ps.setString(8, bloodgroup);
ps.setString(9, specialist);
ps.setString(10, description);
ps.setString(11, "appointment");
ps.setString(12, " ");
ps.setString(13, " ");
int row=ps.executeUpdate();
z=1;
```

```
if (z==row) {
resp.sendRedirect("loginsuccess.jsp");
}else{
resp.sendRedirect("error.jsp");
}
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

Doctorlogin.java

```
package servlet;
import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import dbcon.Connecttodata;
/**
```

```
* Servlet implementation class Doctorlogin
*/
@WebServlet("/Doctorlogin")
public class Doctorlogin extends HttpServlet {
Connection con;
private static final long serialVersionUID = 1L;
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
// TODO Auto-generated method stub
String email = req.getParameter("email");
String ps = req.getParameter("password");
int z=0;
try {
con= Connecttodata.create();
Statement st = con.createStatement();
ResultSet rs= st.executeQuery("SELECT * FROM `medicalrecord`.`doctor`
WHERE email='"+email+"'and password='"+ps+"' and status='Approved'" );
HttpSession session = req.getSession();
rs.next();
String namee =rs.getString(2);
String r1 =rs.getString(6);
String r2 =rs.getString(9);
String r3 =rs.getString(7);
String r4 =rs.getString(8);
String r5 =rs.getString(9);
session.setAttribute("doctorname", namee);
session.setAttribute("hospitalname", r3);
```

```
session.setAttribute("hospitalid", r4);
session.setAttribute("doctorid", r1);
session.setAttribute("specia", r2);
session.setAttribute("specialist", r5);
z=1;
String r =rs.getString("name");
System.out.println(r1);
System.out.println(r2);
resp.sendRedirect("doctorloginsuccessjsp.jsp");
if(z==0) {
resp.sendRedirect("error.jsp");
}
}catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

Doctorregister.java

```
package servlet;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import dbcon.Connecttodata;
/**
* Servlet implementation class Doctorregister
*/
@WebServlet("/Doctorregister")
public class Doctorregister extends HttpServlet {
Connection con;
private static final long serialVersionUID = 1L;
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
// TODO Auto-generated method stub
String name = req.getParameter("username");
String email = req.getParameter("email");
String mobile = req.getParameter("mobilenumber");
String id = req.getParameter("doctorid");
String hname = req.getParameter("hospitalname");
String hid = req.getParameter("hospitalid");
String specialist = req.getParameter("specialist");
String pass = req.getParameter("password");
int z= 0;
try {
con= Connecttodata.create();
```

```
PreparedStatement ps = con.prepareStatement("INSERT INTO medicalrecord.doctor VALUES(id,?,?,?,?,?,?,?,?,?)");
ps.setString(1, name);

ps.setString(2, email);
ps.setString(3, pass);
ps.setString(4, mobile);
ps.setString(5, id);
ps.setString(6, hname);
ps.setString(7, hid);
ps.setString(8, specialist);
ps.setString(9, "Apply");
int row=ps.eecuteUpdate();
resp.sendRedirect("register.jsp");
}
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

Encryptdata.java

```
package servlet;
import java.security.Key;
import javax.crypto.Cipher;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;
```

```
public class Encryptdata {
public static String encrypt(String Data) throws Exception
{
System.out.println("Encrypted in coming");
Key key = Publickey.generateKey();
Cipher c = Cipher.getInstance("AES");
c.init(Cipher.ENCRYPT_MODE, key);
byte[] encVal = c.doFinal(Data.getBytes());
String encryptedValue = new BASE64Encoder().encode(encVal);
return encryptedValue;
}
public static String decrypt(String encryptedData) throws Exception
{
Key key = Publickey.generateKey();
Cipher c = Cipher.getInstance("AES");
c.init(Cipher.DECRYPT_MODE, key);
byte[] decordedValue = new BASE64Decoder().decodeBuffer(encryptedData);
byte[] decValue = c.doFinal(decordedValue);
String decryptedValue = new String(decValue);
//System.out.println("key is :::"+ABEKey.generateKey());
return decryptedValue;
}
}
```

Encryptdata2.java

```
package servlet;
```

```
import java.security.Key;
import javax.crypto.Cipher;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;
public class Encryptdata2 {
public static String encrypt(String Data) throws Exception
{
System.out.println("Encrypted in coming");
Key key = Publickey.generateKey();
Cipher c = Cipher.getInstance("AES");
c.init(Cipher.ENCRYPT_MODE, key);
byte[] encVal = c.doFinal(Data.getBytes());
String encryptedValue = new BASE64Encoder().encode(encVal);
return encryptedValue;
}
public static String decrypt(String encryptedData) throws Exception
{
Key key = Publickey.generateKey();
Cipher c = Cipher.getInstance("AES");
c.init(Cipher.DECRYPT_MODE, key);
byte[] decordedValue = new BASE64Decoder().decodeBuffer(encryptedData);
byte[] decValue = c.doFinal(decordedValue);
String decryptedValue = new String(decValue);
//System.out.println("key is :::"+ABEKey.generateKey());
return decryptedValue;
}
```

}

1.2 CLIENT-SIDE CODING

Aceept.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@page import="dbcon.Connecttodata"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement" %>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String id = request.getParameter("id").toString();
Connection con;
con=Connecttodata.create();
ps.executeUpdate();
response.sendRedirect("Managementlist.jsp");
%>
<% %>
</body>
</html>
```

Adminlogin.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-
H2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI
Bx" crossorigin="anonymous">
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/js/bootstrap.bundle.min.j
s"
integrity="sha384-
A3rJD856KowSb7dwlZdYEkO39Gagi7vIsF0jrRAoQmDKKtQBHUuLZ9AsSv4j
D4Xa"
crossorigin="anonymous"></script>
<title>Insert title here</title>
</head>
<body>
</head>
<body>
<div class="vh-100 d-flex justify-content-center align-items-center ">
<div class="col-md-5 p-5 shadow-sm border rounded-5 border-primary bg-
white">
<h2 class="text-center mb-4 text-primary">Login Form</h2>
<form action="Adminlogin" method="post">
<div class="mb-3">
<label for="exampleInputEmail1" class="form-label">Email address</label>
<input type="email" name="email" class="form-control border border-primary"
id="exampleInputEmail1" aria-describedby="emailHelp">
</div>
<div class="mb-3">
<label for="exampleInputPassword1" class="form-label">Password</label>
<input type="password" name="password" class="form-control border border-
primary" id="exampleInputPassword1">
</div>
<div class="d-grid">
<button class="btn btn-primary" type="submit">Login</button>
```

```
</div>
</form>
</div>
</div>
</body>
</html>
```

Appointmentsuccess.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>welcome</h1>
</body>
</html>
```

Approvedocument.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@page import="dbcon.Connecttodata"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement" %>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
```

```
<body>
 <%
String id = request.getParameter("id").toString();
Connection con;
con=Connecttodata.create();
PreparedStatement ps=con.prepareStatement("UPDATE
medicalrecord`.`prescription` SET status='sended' where status='Request' and
id='"+id+"' ");
ps.executeUpdate();
response.sendRedirect("documentrequest.jsp");
%>
<% %>
</body>
</html>
```

Doctoraccept.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@page import="dbcon.Connecttodata"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement" %>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String s =session.getAttribute("doctorid").toString();
String s1 =session.getAttribute("specialist").toString();
String id = request.getParameter("id").toString();
Connection con;
```

```
con=Connecttodata.create();
PreparedStatementps=con.prepareStatement("UPDATE
`medicalrecord`.`appointmentform`SET
status='Approved',doctorid='"+s+"',doctorspecialist='"+s1+"'where
status='appointment' and id='"+id+"' ");
ps.executeUpdate();
response.sendRedirect("Appointmentview.jsp");
%>
<% %>
</body>
</html>
```

Doctorregister.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<linkhref='https://fonts.googleapis.com/css?family=Didact+Gothic'
rel='stylesheet' type='text/css'>
<linkhref="//maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-
awesome.min.css" rel="stylesheet">
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<style type="text/css">
```

HTML CSSResult Skip Results Iframe

```
body{
  margin:0 auto;
  width:90%;
  padding:0;
  font-family: 'Didact Gothic', sans-serif;
}
.wrap{
  margin-top:30px;
  width:480px;
}
```

```
fieldset{
  width: 400px;
  border-radius:10px;
  border-right:1px solid transparent;
  border-left:1px solid transparent;
}
legend{
  background:#457b9d;
  width:100px;
  height:100px;
  margin-left:168px;
border-radius:50px;
}
h3{
position:relative;
padding-top:35px;
text-align:center;
color:#FF6D2E;
}
h2{
text-align: center;
font-size: 0.9em;
color:lightgrey;
padding-top: 10px;
}
input{
width:170px;
height: 50px;
margin-left:30px;
margin-top: 15px;
text-align: center;
border-radius: 10px;
border: 1px solid lightgrey;
color: #4361ee;
font-weight: bold;
}
```

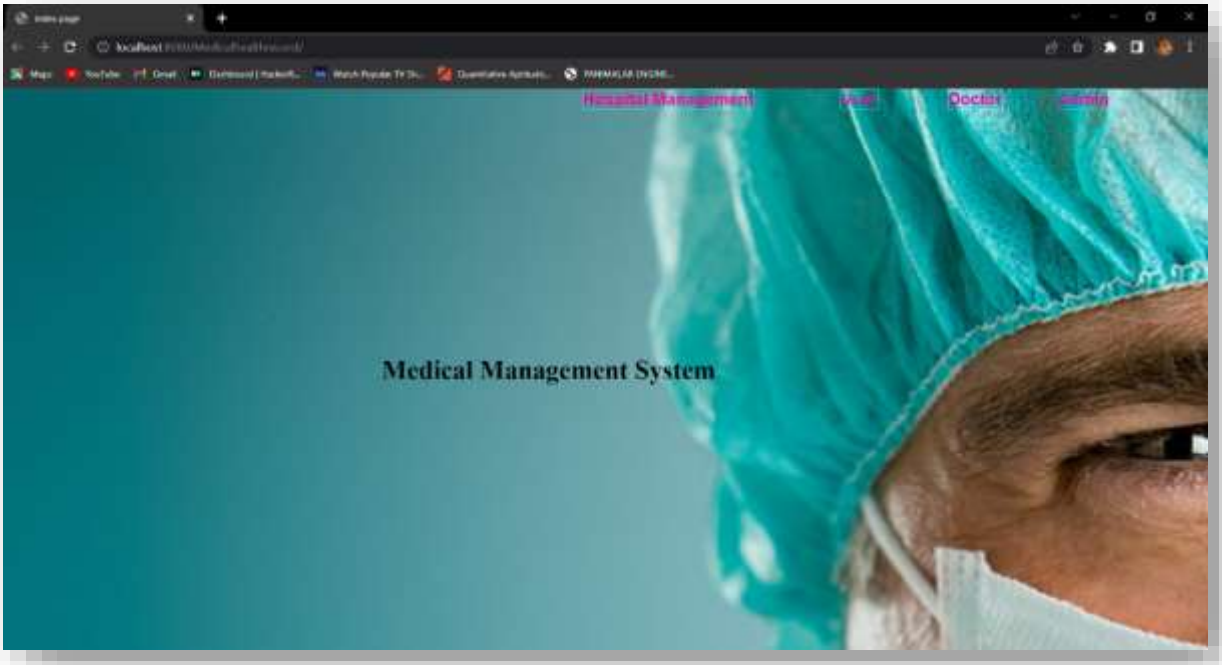
```
input:focus{
border:1px solid rgba(255,109,46,0.6);
box-shadow: 0 0 3px rgba(255,109,46,0.6);
outline: none;
font-size: 1.1em;
}
button{
background: #fff;
border: 1px solid transparent;
border-radius:10px ;
width: 120px;
height: 50px;
margin-left: 20px;
margin-top:15px;
color: #7D7769;
}
button:hover{
background:#FBF7EF ;
border-radius: 10px;
}
button:focus{
border:1px solid rgba(255,109,46,0.6);
box-shadow: 0 0 3px  rgba(255,109,46,0.6);
outline: none;
font-size: 1.1em;
}
.fa-facebook{
color:#4F79A7;
}
.fa-google-plus{
color:#D64E3C;
}
fa-twitter{
color:#4ECDED;
}
.fa-facebook, .fa-google-plus, .fa-twitter{
padding: 8px;
```

```
}
input[type=submit]{
background: #FF9D73;
color: #3A3D52;
margin:20px 140px 30px;
}
input[type=submit]:hover{
background:#FF8652;
}
form{
position: absolute;right: 400px;
}
</style>
</head>
<body>
<div class="wrap">
<form action="Doctorregister" method="post">
<fieldset>
<legend><h3>Sign Up</h3></legend>
<input type="text" name="username" placeholder="Your Name"/>
<input type="text" name="email" placeholder="Email"/>
<input type="password" name="password" placeholder="Password"/>
<input type="text" name="mobilenumber" placeholder="mobile number"/>
<input type="text" name="doctorid" placeholder="doctor id"/>
<input type="text" name="hospitalname" placeholder="hospital name"/>
<input type="text" name="hospitalid" placeholder="hospital id"/>
<input type="text" name="specialist" placeholder="specialist"/>
<input type="submit"  value="Submit">
</fieldset>
</form>
</div>
</body>
</html>
```

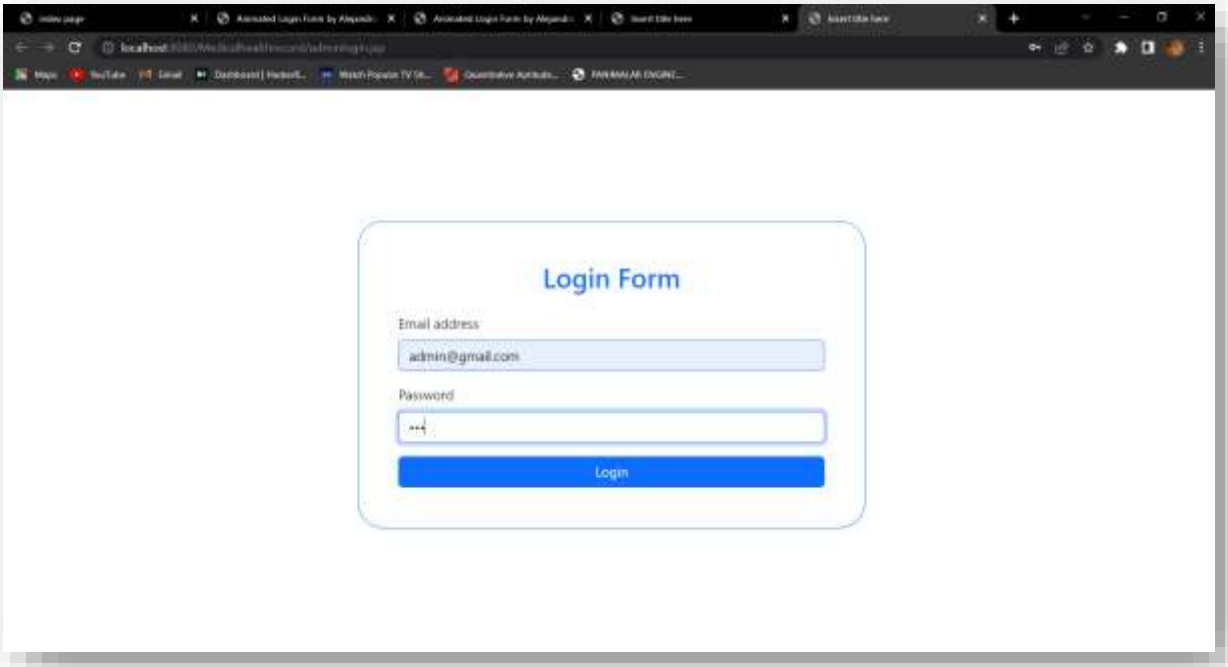
Documentaccept.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@page import="dbcon.Connecttodata"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement" %>
<%@page import="java.sql.*" %>
<%@page import="java.util.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String id = request.getParameter("id").toString();
Connection con;
con=Connecttodata.create();
PreparedStatement ps=con.prepareStatement("UPDATE `medicalrecord`.`user`
SET status='approved' where status='request' and id='"+id+"' ");
ps.executeUpdate();
response.sendRedirect("patientrequest.jsp");
%>
<% %>
</body>
</html>
```

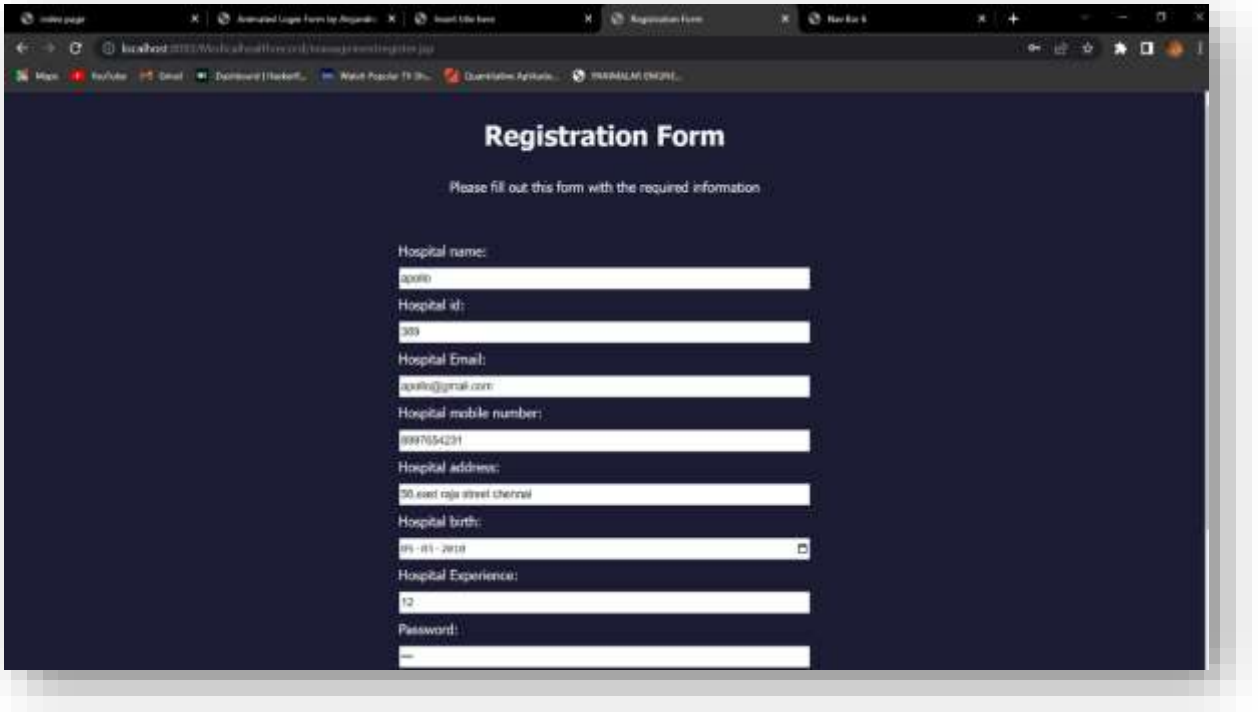
APPENDIX 2. SAMPLE



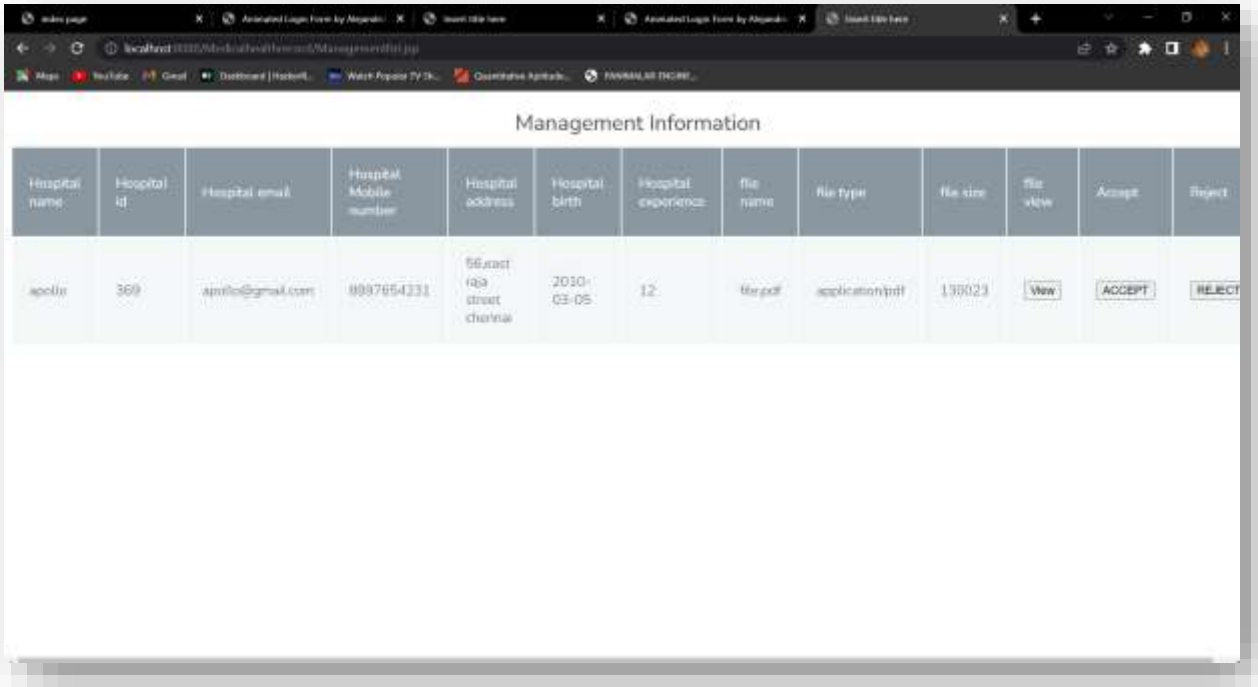
Scr 2.1. Home page



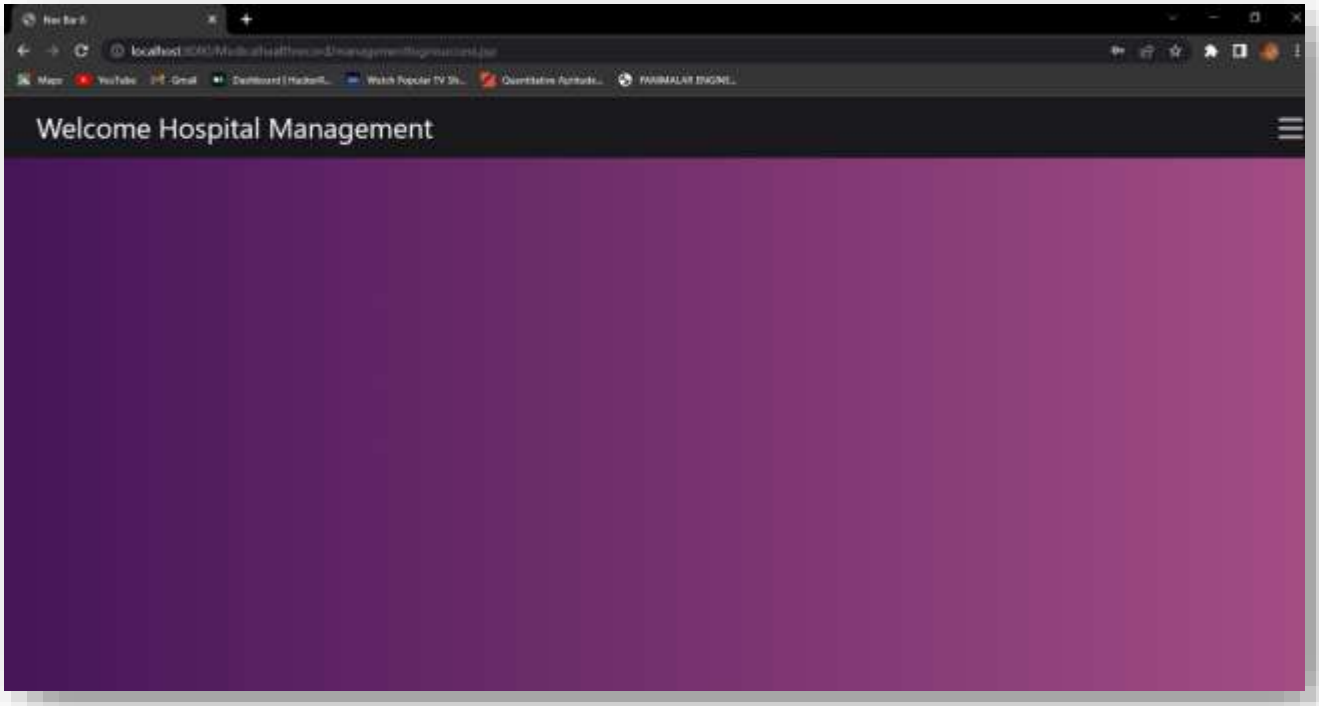
Scr 2.2. ADMIN PAGE



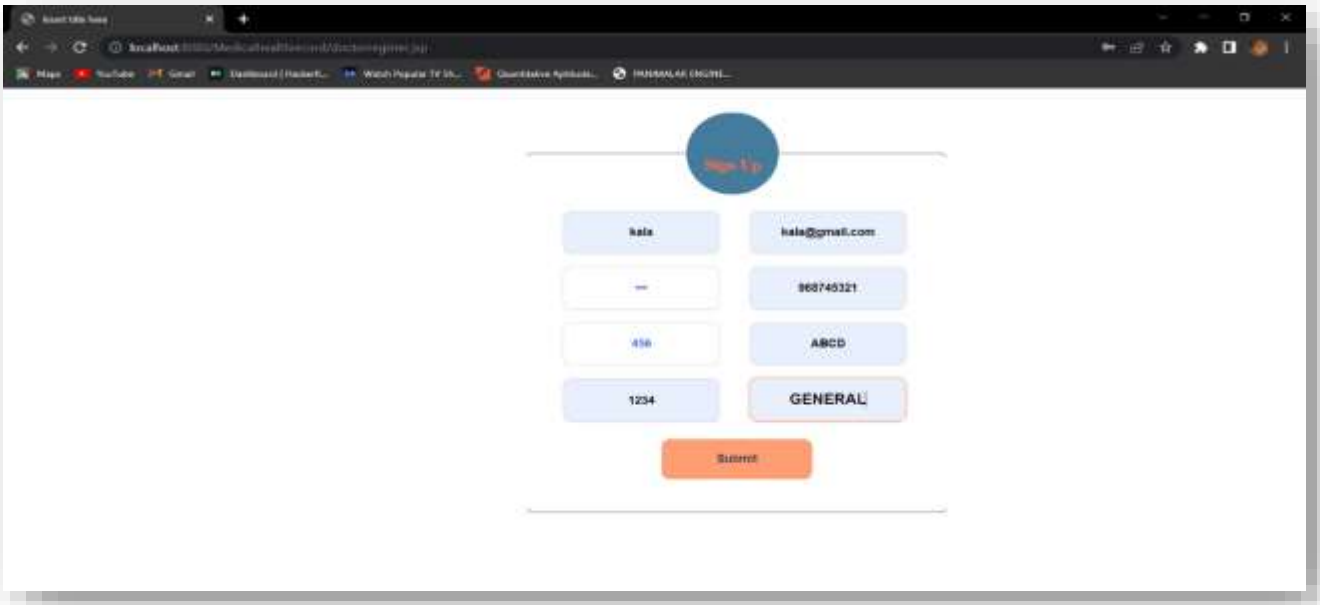
Scr 2.3. Hospital management register page



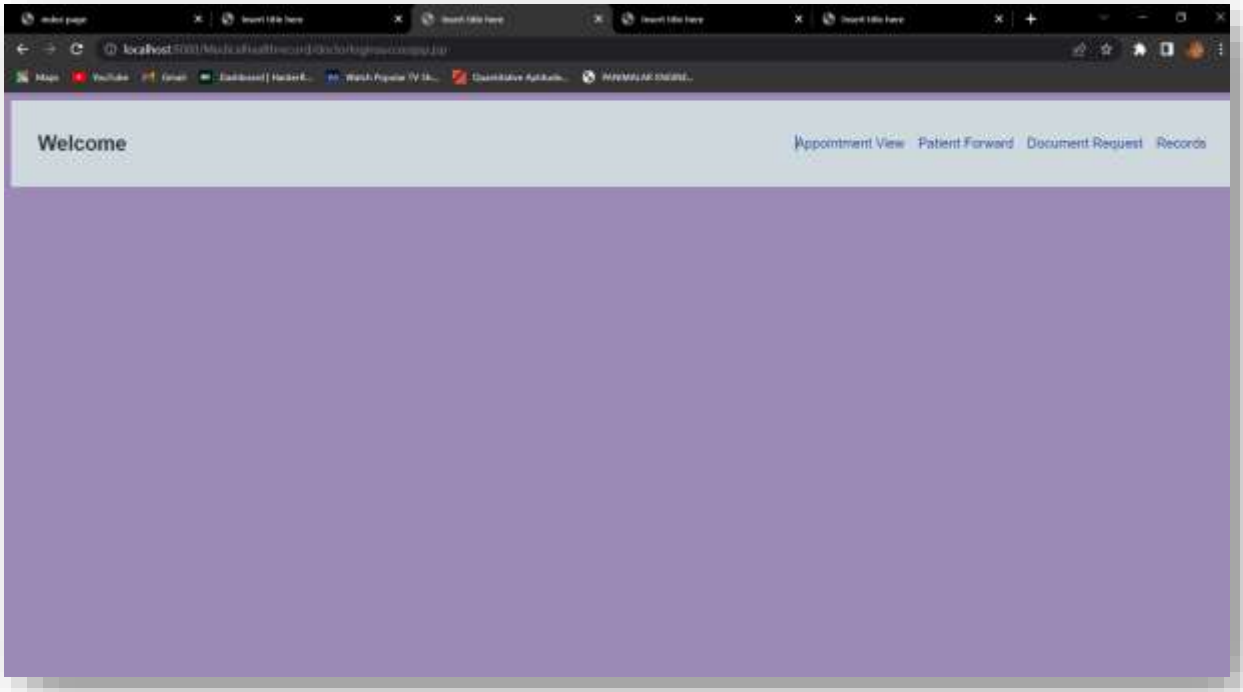
Scr 2.4. Management information



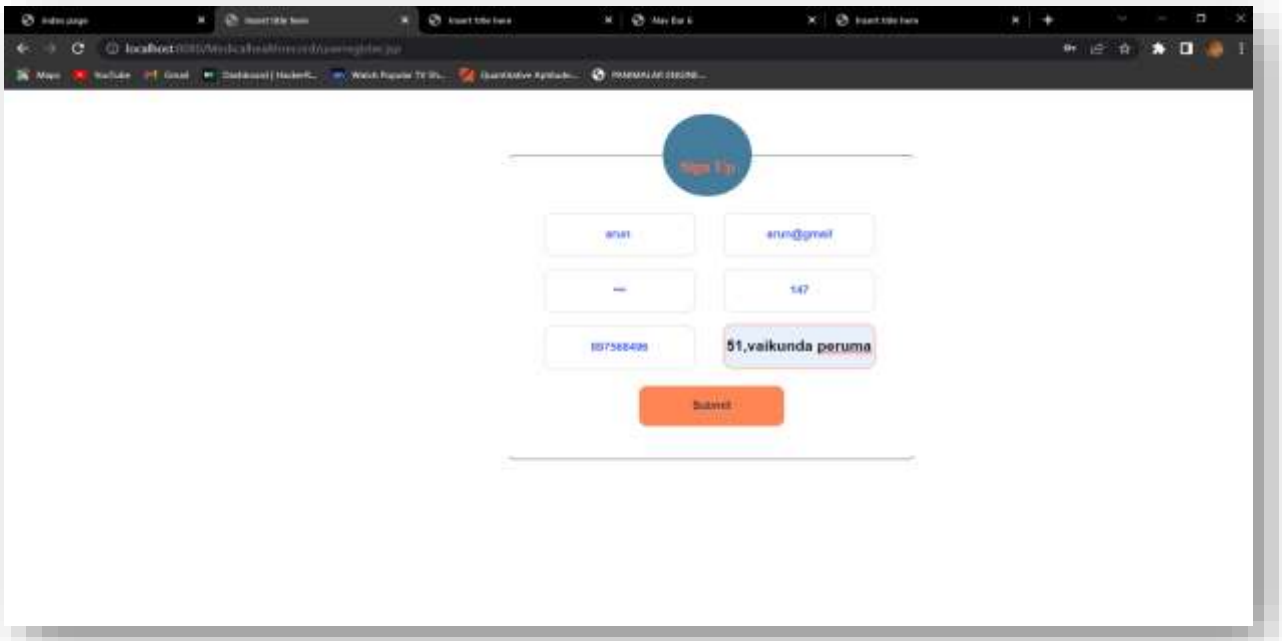
Scr 2.5. Hospital management home page



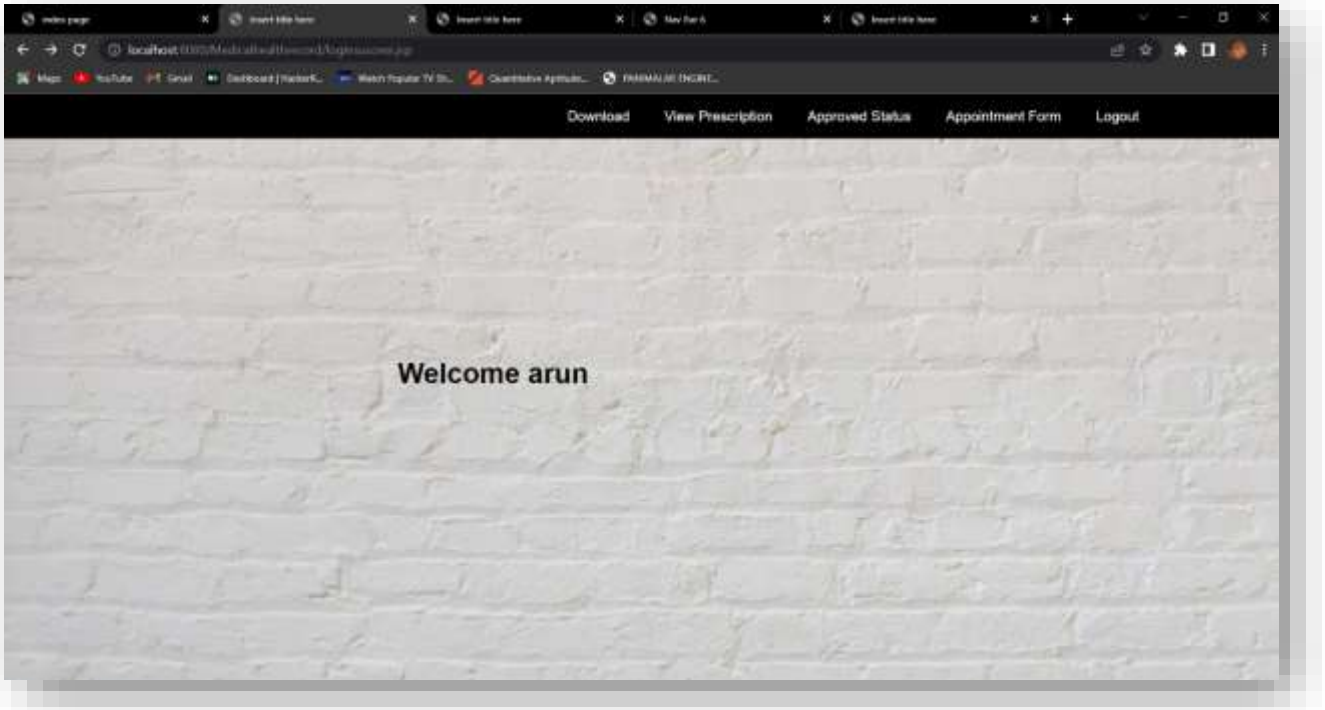
Scr 2.6. Doctor register page



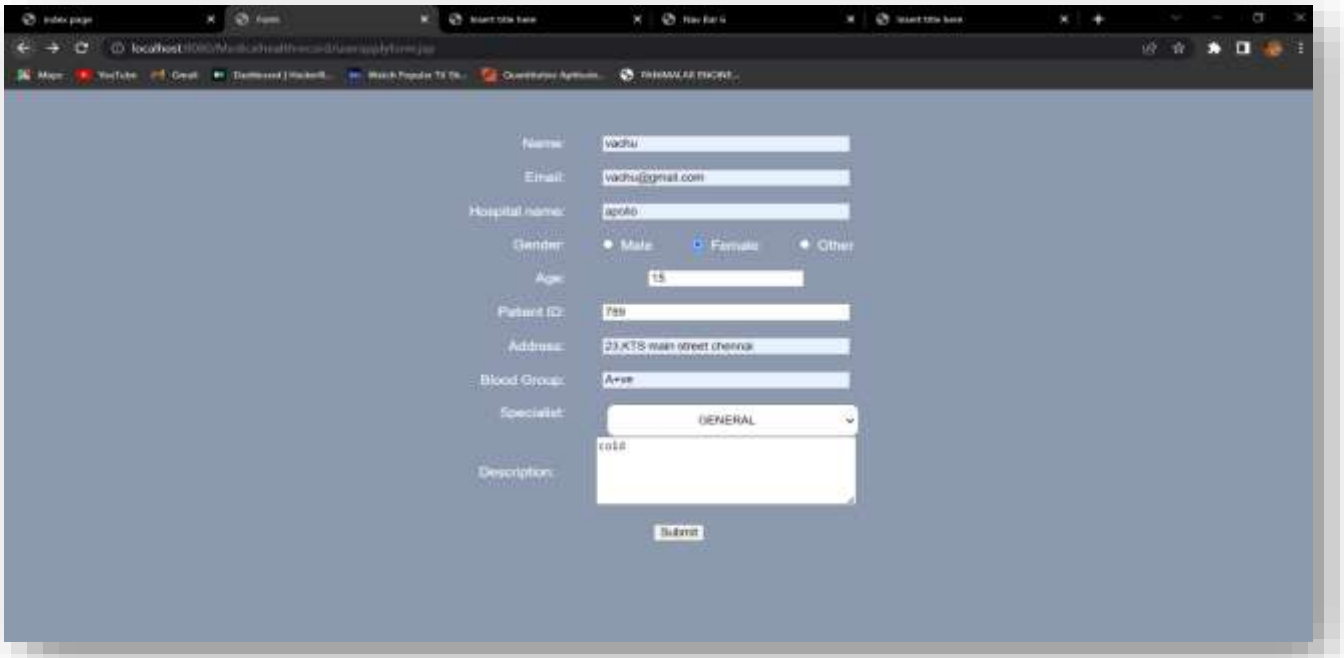
Scr 2.7. Doctor home page



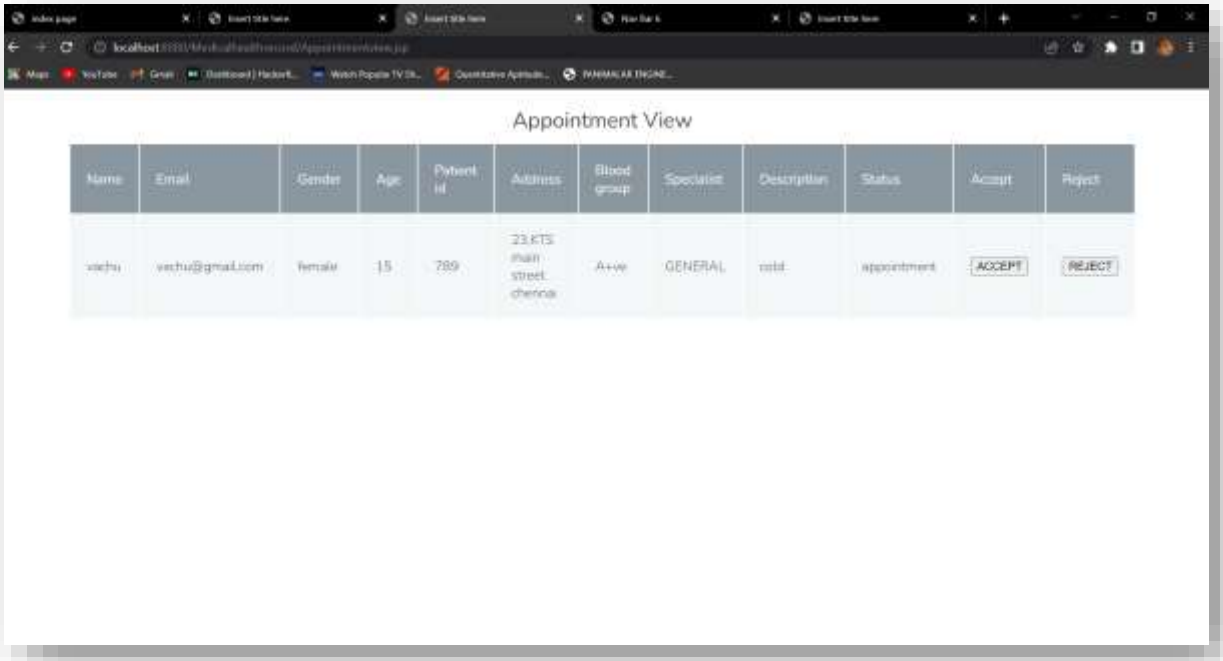
Scr 2.8. Patient register page



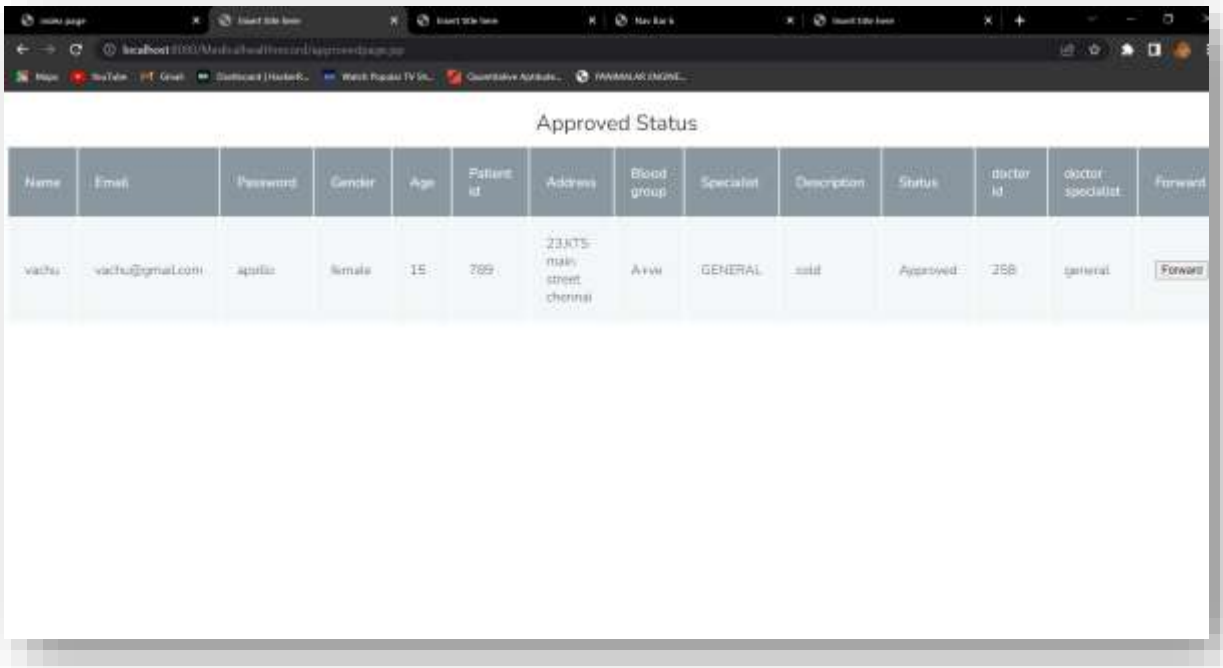
Scr 2.9. Patient home page



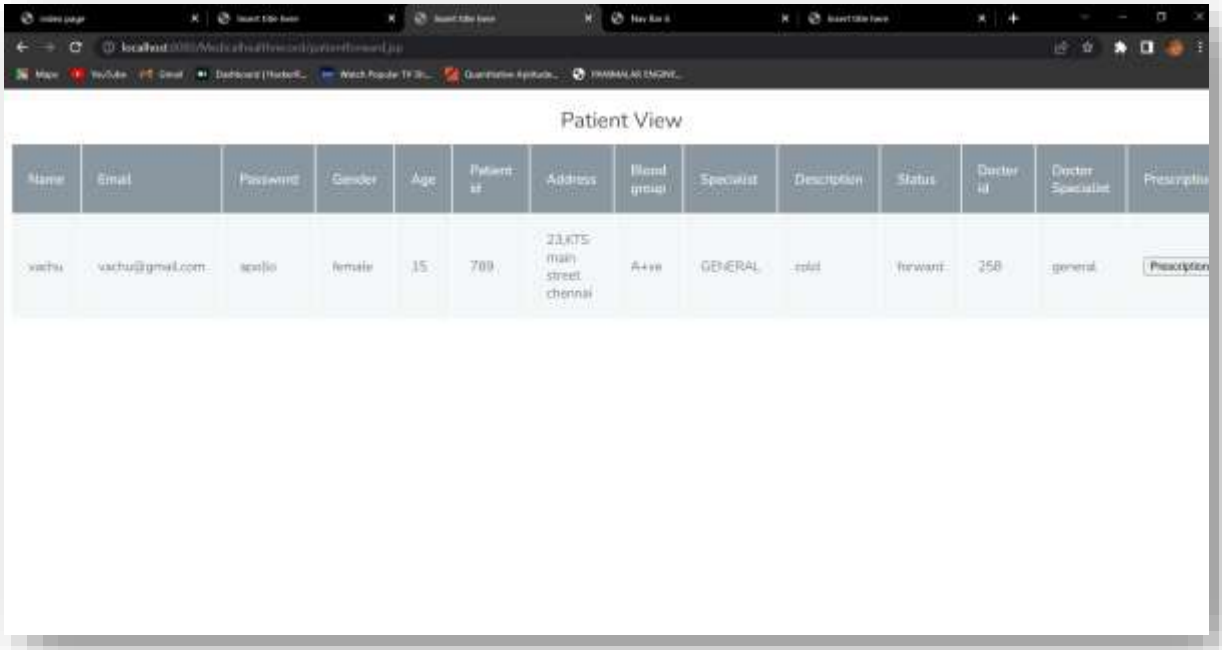
Scr 2.10. Appointment form



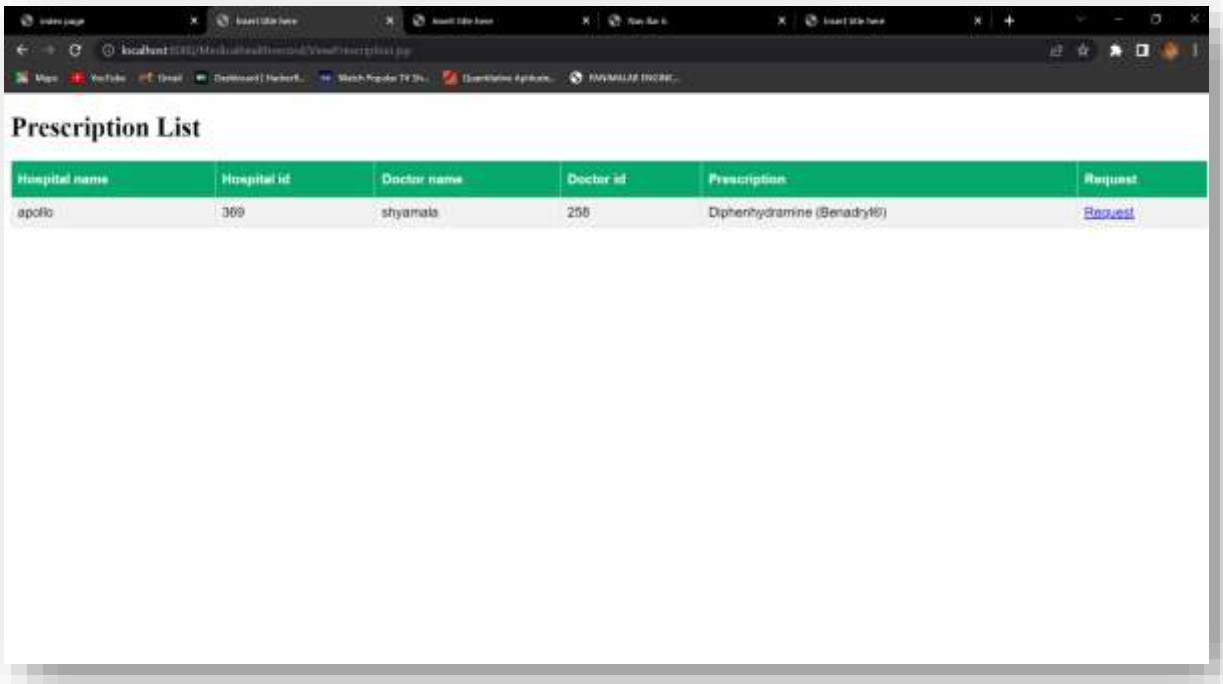
Scr 2.11. View appointment



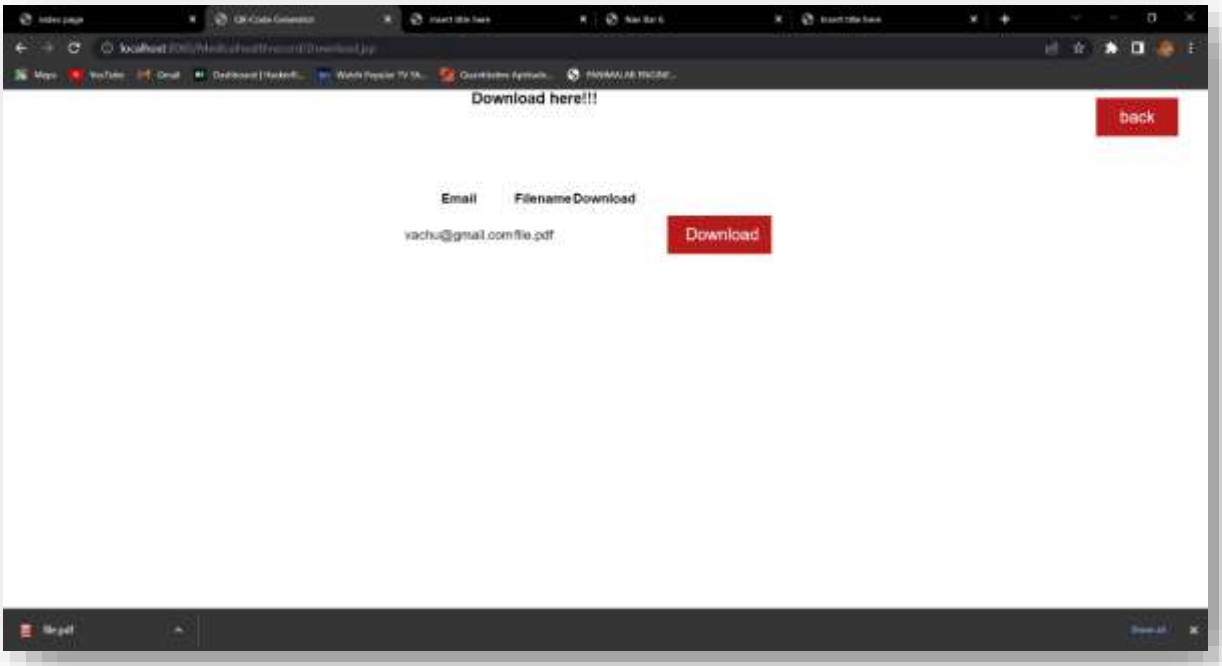
Scr 2.12. Appointment status



Scr 2.13. Prescription page



Scr 2.14. Request prescription



Scr 2.15. Download prescription

REFERENCES

[1] A. Aflaki, B. Feldman, and R. Swinney, “Becoming strategic: Endogenous consumer time preferences and multiperiod pricing,” Oper. Res., vol. 68, no. 4, pp. 1116–1131, 2020.

[2] S. Asian and X. Nie, “Coordination in supply chains with uncertain demand and disruption risks: Existence, analysis, and insight,” IEEE Trans. Syst., Man, Cybern. Syst., vol. 44, no. 9, pp. 1139–1154, Sep. 2014.

[3] C. T.M., J. Zhang, andY. J. Cai, “Consumer-to-consumer digital-productexchange in the sharing economy system with risk considerations: Will digital-product-developers suffer?” IEEE Trans. Syst., Man, Cybern. Syst., vol. 50, no. 12, pp. 5049–5057, Dec. 2020.

[4] N. Boysen, D. Briskorn, and S. Schwerdfeger, “Matching supply and demand in a sharing economy: Classification, computational complexity, and application,” Eur. J. Oper. Res., vol. 278, no. 2, pp. 578–595, 2019.

[5] S. Benjaafar and M. Hu, “Operations management in the age of the sharing economy: What is old and what is new?,” Manuf. Service Oper. Manage., vol. 22, no. 1, pp. 93–101, 2019.

[6] Y. Aviv and A. Pazgal, “Optimal pricing of seasonal products in the presence of forward-looking consumers,” Manuf. Service Oper. Manage., vol. 10, no. 3, pp. 339–359, 2008.

[7] K. Cao, X. Xu,Y. Bian, andY. Sun, “Optimal trade-in strategy of businessto-consumer platform with dual-format retailing model,” Omega, vol. 82, pp. 181–192, 2019.

[8] T. M. Choi and Y. He, “Peer-to-peer collaborative consumption for fashion products in the sharing economy: Platform operations,” Transp. Res. Part E: Logistics Transp. Rev., vol. 126, pp. 49–65, 2019.

[9] S. Benjaafar, G. Kong, X. Li, and C. Courcoubetis, “Peer-to-peer product sharing: Implications for ownership, usage, and social welfare in the sharing economy,” Manage. Sci., vol. 65, no. 2, pp. 477–493, 2018.

[10] G. P. Cachon and R. Swinney, “Purchasing, pricing, and quick response in the presence of strategic consumers,” Manage. Sci., vol. 55, no. 3, pp. 497–511, 2009.

[11] C. H. Chiu, H. L. Chan, and T. M. Choi, “Risk minimizing price-rebatereturn contracts in supply chains with ordering and pricing decisions: A multimethodological analysis,” IEEE Trans. Eng. Manage., vol. 67, no. 2, pp.466–482, 2020.

[12] I. Bellos, M. Ferguson, and L. B. Toktay, “The car sharing economy: Interaction of business model choice and product line design,” Manuf. Service Oper. Manage., vol. 19, no. 2, pp. 185–201, 2017.