

ENHANCING DATA SECURITY IN CLOUD USING BLOCK CHAIN

In the partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

A project submitted by

NURUKURTHI UDAYA SREE

(Regd. No: 321206420034)

Under the guidance of

Dr. BHARATI BIDIKAR

Adjunct Professor



**DEPARTMENT OF INFORMATION TECHNOLOGY AND
COMPUTER APPLICATIONS**

**ANDHRA UNIVERSITY COLLEGE OF ENGINEERING,
ANDHRA UNIVERSITY, VISAKHAPATNAM – 530003**

(2021-2023)

**DEPARTMENT OF INFORMATION TECHNOLOGY AND
COMPUTER APPLICATIONS**

ANDHRA UNIVERSITY COLLEGE OF ENGINEERING

ANDHRA UNIVERSITY, VISAKHAPATNAM - 530003



CERTIFICATE

This is to certify that the project report entitled “**ENHANCING DATA SECURITY IN CLOUD USING BLOCK CHAIN.**”, is the bonafide work carried out by **NURUKURTHI UDAYA SREE** with **REGD. NO: 321206420034**, a student of MCA in AU COLLEGE OF ENGINEERING, ANDHRA UNIVERSITY, VISAKHAPATNAM, during the year 2021-2023, in partial fulfilment of the requirements for the award of degree of **MASTER OF COMPUTER APPLICATIONS**.

Adjunct Prof. DR BHARATI BIDIKAR

PROJECT GUIDE

Prof. KUNJAM NAGESWAR RAO

HEAD OF THE DEPARTMENT

ACKNOWLEDGEMENT

It is with great sense of satisfaction that I present ENHANCING DATA SECURITY IN CLOUD USING BLOCK CHAIN in the form of final project.

I express my deep sense of gratitude to my Project Guide, Dr BHARATI BIDIKAR, Department of Information Technology and Computer Applications (IT&CA), Andhra University College of Engineering, for guiding me all through the project work, giving right direction and shape to my learning by extending her expertise and experience in the education.

Really, I'm indebted for her excellent and enlightened guidance.

I am very thankful to our beloved Head of the Department Prof. KUNJAM NAGESWAR RAO, Department of Information Technology and Computer Applications (ITCA), Andhra University College of Engineering (A), for his valuable suggestions and constant motivation that greatly helped the project to successfully complete.

I would like to thank Prof D. LALITHA BHASKARI, Chairman, BOS and Prof. G. SASIBHUSHAN RAO, Principal AUCE for this valuable cooperation and encouragement. I also extend my heartfelt gratitude to all the teaching and nonteaching staff of the Department of Information Technology and Computer Applications (ITCA) for their support. I gratefully acknowledge the support, encouragement and patience of my parents

DECLARATION

I NURUKURTHI UDAYA SREE, Reg No:321206420034 hereby declare that the project entitled Enhancing Data Security in Cloud Using Block Chain is a record of original work undertaken by me. I have completed this project under the supervision of DR BHARATI BIDIKAR an original work done at ANDHRA UNIVERSITY COLLEGE OF ENGINEERING, VISAKHAPATNAM, submitted in partial fulfillment of the requirements for the award of degree of MASTER OF COMPUTER APPLICATIONS (MCA) in COMPUTER SCIENCE.

I also declared that this project has not prior submitted for the award of any degree, diploma, associateship, fellowship or other title by anyone. I thus confirm the originality of the work and that there is no plagiarism I any part on the project report.

Place: Visakhapatnam
Department of IT & CA
Andhra University
Visakhapatnam.

NURUKURTHI UDAYA SREE
Reg No: 321206420034

ABSTRACT

Daily lots of data is exchanged and loaded on a cloud into different sectors one of which is the health sector. Data exchanged between the patient and doctors need to be secured to gain patients' trust. Blockchain is a mechanism invented to secure data in a more advanced way. Blockchain stores data into chunks that make it hard to decode, which will help provide an extra layer of security. Hash chain is the most reliable part of the blockchain that will help keep the data unreadable. This data can be secured by using a blockchain mechanism at the backend of any hospital website to store the reports of the patients and maintain a two-way authentication for doctor's access to the reports. Using the concepts of dividing data into chunks and establishing an inter-link between each chunk is one of the aspects of blockchain which is implemented on the hospital generated data to inherit blockchain mechanism. In this paper, we have discussed the benefits of using this mechanism to secure patients' reports and how it increases trust in the stored data.

Keywords: - Blockchain, Cloud, Healthcare, Security

TABLE OF CONTENTS

ABSTRACT	
CHAPTER 1: INTRODUCTION	1 - 4
1.1 Motivation and work	2
1.2 Aim & Objectives	3
1.3 Existing System	4
CHAPTER 2: LITERATURE SURVEY	5
CHAPTER 3: SYSTEM ANALYSIS	9 - 10
3.1 Proposed System	9
3.2 Proposed Method	9
3.3 Feasibility Study	10
CHAPTER 4: SYSTEM SPECIFICATIONS	11
4.1 Functional Requirements	11
4.2 Non-functional requirements	11
4.3 Hardware requirements	11
4.4 Software requirements	11
CHAPTER 5: SYSTEM DESIGN	13 - 14
5.1 System Architecture	13
5.2 UML Diagrams	14
CHAPTER 6: SYSTEM IMPLEMENTATION	26 - 45
6.1 Project Modules	26
6.2 Methodology (Algorithms)	26
6.3 Source Code	29
6.4 Results	41
6.4.1 Inputs	41
6.4.2 Outputs	45

CHAPTER 7: SYSTEM TESTING	48 - 49
7.1 Testing Methods	48
CHAPTER 8: CONCLUSION	50
CHAPTER 9: REFERENCE	51

LIST OF FIGURES

FIGURE	NAME OF THE FIGURE	PAGE
3.2.1	Proposed System Architecture	9
5.1	System Architecture	12
5.3.1	Use Case Diagram	16
5.3.2	Class Diagram	17
5.3.3	Sequence Diagram	18
5.3.4	Collaboration Diagram	19
5.3.5	Activity Diagram	20
5.3.6	Component Diagram	21
5.3.7	Deployment Diagram	21
5.3.8	ER Diagram	22
5.3.9	Context Diagram	23
5.3.10	Level -1 Diagram	24
5.3.11	Level - 2 Diagram	25
6.2.2	SHA-256 algorithm for processing a single block	28

CHAPTER 1

INTRODUCTION

Despite using certain frameworks, we have implemented the idea through hard coding. Each report is divided into chunks and these chunks need to be secured to maintain the integrity of data. Each of the chunks is encrypted through AES the key generation algorithm generates the public key (PK), the master key (MK), secret key (SK) of the user. There is a K number of users in group sharing data. Master, the user is the owner of data. So, all the users can access and modify the shared data in the cloud. TPA performs data integrity auditing for modified data of the user. Key Generation as the part of the setup algorithm generates public keys (PK), master keys (MK) of the system, and secret key (SK) of users. In our design each user will have their secret key for data modification Key generation is a technique which is used to store the data in a different methodology and mainly the public key algorithm known as RSA plays a vital role in key generation technologies such as single shared key uses symmetric key algorithm through which data will be stored very secretly Since the public key algorithm uses two keys namely public and a private key and that public key is made as visible to one end user so that they can use that public key to encrypt the data and another end-user can decrypt the data using their private key.

In some conditions, the keys have been generated using the random number generator technique, and it is very efficient that hackers cannot easily guess the keys and provide strong security. the major focus in our paper is about the healthcare data. Hospitals generate a large amount of confidential data and especially every healthcare center needs to maintain HIPAA rules. Here we are using a blockchain mechanism to the data that's been uploaded by the patients regarding their diseases and that data is then secured through blockchain mechanism. This security mechanism will be the backend processing of any hospital website, Recently, a few attempts started considering more realistic scenarios by allowing multiple cloud users to modify data with integrity assurance. Nevertheless, these attempts are still far from practical due to the tremendous computational cost on cloud users, especially when high error detection probability is required by the system. We used Amazon Cloud Storage service S3 for storing data divided into chunks in the form of buckets.

1.1 Motivation of work

Traditional cloud storage and data management systems rely on centralized servers, which can be vulnerable to single points of failure and attacks. Blockchain's decentralized nature offers a distributed network of nodes that collectively validate and secure the data, eliminating the need for a central authority and enhancing trust in the system. Blockchain provides an immutable ledger where data once recorded cannot be altered or tampered with without consensus from the network participants. By storing data in a blockchain, it becomes tamper-proof, ensuring the integrity and authenticity of the information. Cloud services often involve storing sensitive data that needs protection from unauthorized access or breaches. Blockchain technology can offer improved data privacy by employing encryption, access controls, and decentralized identity management, ensuring that only authorized parties can access and interact with the data.

Blockchain's transparent nature allows for increased transparency and auditability of data transactions. Every transaction recorded on the blockchain can be traced and audited, making it easier to detect any unauthorized or suspicious activities and enabling better compliance with data protection regulations. Blockchain platforms support smart contracts, which are self-executing contracts with predefined conditions. Smart contracts enable the automation and enforcement of security protocols, access controls, and permissions, reducing the risk of human error or malicious activities. Data stored in the cloud can be vulnerable to unauthorized modifications or tampering. Blockchain's cryptographic algorithms and consensus mechanisms enable data integrity verification, ensuring that data remains unaltered and trustworthy throughout its lifecycle. Blockchain fosters collaboration among participants within the cloud ecosystem, such as cloud service providers, users, and auditors. By leveraging blockchain technology, stakeholders can work together to enhance data security, share threat intelligence, and establish common standards and best practices.

1.2 Aim & Objectives

Aim:

The aim of enhancing data security in the cloud using blockchain is to leverage the decentralized and immutable nature of blockchain technology to strengthen the security of data stored, transmitted, and processed within cloud environments. By incorporating blockchain into cloud computing, the goal is to provide increased transparency, integrity, and confidentiality of data, reducing the risk of unauthorized access, tampering, or data breaches.

Objectives:

- **Establish a decentralized and tamper-resistant data storage:** Implement a blockchain-based data storage system within the cloud infrastructure, ensuring that data is stored in a decentralized manner across multiple nodes. This reduces the risk of a single point of failure and enhances data resilience against malicious attacks.
- **Strengthen access control mechanisms:** Leverage blockchain's smart contract capabilities to enhance access control in cloud environments. Smart contracts can define and enforce fine-grained access permissions, ensuring that only authorized individuals or entities can access sensitive data stored in the cloud.
- **Implement secure data sharing and collaboration:** Enable secure sharing and collaboration of data within the cloud environment using blockchain-based mechanisms. This includes implementing encryption, key management, and access control protocols that leverage blockchain technology to ensure data confidentiality and privacy.
- **Address scalability and performance challenges:** Explore innovative approaches to overcome the scalability and performance limitations of blockchain technology when applied to cloud environments. This includes optimizing consensus algorithms, network architecture, and storage mechanisms to ensure efficient data processing and minimize latency.
- **Collaborate with industry stakeholders:** Foster collaboration between cloud service providers, blockchain developers, cybersecurity experts, and regulatory bodies to develop standardized best practices, protocols, and frameworks for enhancing data security in the cloud using blockchain. Sharing knowledge and expertise can accelerate the adoption of secure cloud solutions and ensure interoperability between different platforms.

1.3 EXISTING SYSTEM

Existing use of blockchain is for electronic transactions of crypto currencies through public blockchain where these transactions are recorded as blocks of a distributed network and a huge chain is maintained which requires a great amount of computational power. Considering the cloud, the cloud has its own security rules and regulations such as IDS, Encryption methods, etc. But in the continuous increase, the amount of data there lies certain loopholes that may tamper the data security.

Disadvantages:

- Patient's data can be shared.
- Not secured.
- Chance of privacy disclosure.

CHAPTER 2

LITERATURE SURVEY

[1] Suma, V. (2019). SECURITY AND PRIVACY MECHANISM USING BLOCKCHAIN. Journal of Ubiquitous Computing and Communication Technologies (UCCT), 1(01), 45-54

Block chain being a foundational technology impacting and attracting a wide range of applications has become predominant in solving the problem of privacy preserving and security in multitude sectors that is under the control of the government and the private. The paper also presents the security and the privacy mechanism using the block chain to prevent the misuse and the corruption in the sharing of huge set of data generated from the judiciary, security, legislature, commercial code registries etc. The proposed system enables reliability and the trust in the data sharing in the communication channels utilizing the block chain with the RSA digital signature. The proposed system is simulated as a java programming version to evince the enhancement in the latency in the sharing of the information's along with the privacy and the security.

[2] Homoliak, I., Venugopalan, S., Hum, Q., & Szalachowski, P. (2019). A Security Reference Architecture for Blockchains. 2019 IEEE International Conference on Blockchain (Blockchain).

Due to their specific features, blockchains have become popular in recent years. Blockchains are layered systems where security is a critical factor for their success. The main focus of this work is to systematize knowledge about security and privacy issues of blockchains. To this end, we propose a security reference architecture based on models that demonstrate the stacked hierarchy of various threats as well as threat-risk assessment using ISO/IEC 15408. In contrast to the previous surveys, we focus on the categorization of security vulnerabilities based on their origins and using the proposed architecture we present existing prevention and mitigation techniques. The scope of our work mainly covers aspects related to the nature of blockchains, while we mention operational security issues and countermeasures only tangentially.

[3] Killer, C., Rodrigues, B., & Stiller, B. (2019). Security Management and Visualization in a Blockchain-based Collaborative Defense. 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC).

A cooperative network defense is one approach to fend off large-scale Distributed Denial-of-Service (DDoS) attacks. In this regard, the Blockchain Signaling System (BloSS) is a multi-domain, blockchain-based, cooperative DDoS defense system, where each Autonomous System (AS) is taking part in the defense alliance. Each AS can exchange attack information about ongoing attacks via the Ethereum blockchain. However, the currently operational implementation of BloSS is not interactive or visualized, but the DDoS mitigation is automated. In real-world defense systems, a human cybersecurity analyst decides whether a DDoS threat should be mitigated or not. Thus, this work presents the design of a security management dashboard for BloSS, designed for interactive use by cyber security analysts.

[4] Koo, J., Kim, Y.-G., & Lee, S.-H. (2019). Security Requirements for Cloud-based C4I Security Architecture. 2019 International Conference on Platform Technology and Service (PlatCon).

With the development of cloud computing technology, developed countries including the U.S. are performing the efficiency of national defense and public sector, national innovation, and construction of the infrastructure for cloud computing environment through the policies that apply cloud computing. Korea Military is also considering that apply the cloud computing technology into its national defense command control system. However, only existing security requirements for national defense information system cannot solve the problem related security vulnerabilities of cloud computing. In order to solve this problem, it is necessary to design the secure security architecture of national defense command control system considering security requirements related to cloud computing. This study analyze the security requirements needed when the U.S. military apply the cloud computing system. It also analyze existing security requirements for Korea national defense information system and security requirements for cloud computing system and draw the security requirements needed to Korea national defense information system based on cloud computing.

[5] Shen, J., Zhou, T., He, D., Zhang, Y., Sun, X., & Xiang, Y. (2018). Block Design-based Key Agreement for Group Data Sharing in Cloud Computing. IEEE Transactions on Dependable and Secure Computing.

Data sharing in cloud computing enables multiple participants to freely share the group data, which improves the efficiency of work in cooperative environments and has widespread potential applications. However, how to ensure the security of data sharing within a group and how to efficiently share the outsourced data in a group manner are formidable challenges. Note that key agreement protocols have played a very important role in secure and efficient group data sharing in cloud computing. In this paper, by taking advantage of the symmetric balanced incomplete block design (SBIBD), we present a novel block design-based key agreement protocol that supports multiple participants, which can flexibly extend the number of participants in a cloud environment according to the structure of the block design. Based on the proposed group data sharing model, we present general formulas for generating the common conference key IC for multiple participants. Note that by benefiting from the $(v, k + 1, 1)$ -block design, the computational complexity of the proposed protocol linearly increases with the number of participants and the communication complexity is greatly reduced. In addition, the fault tolerance property of our protocol enables the group data sharing in cloud computing to withstand different key attacks, which is similar to Yi's protocol.

[6] Praveena, A., and S. Smys. "Ensuring data security in cloud based social networks." In 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), vol. 2, pp. 289-295. IEEE, 2017.

Nowadays, Online Social Networks is one of the important terms we hear, which allows its users to connect by various link types. Everyday application developers come up with new social networking sites. Consequently, these websites gain huge profit just by providing a platform for the users to communicate. It has already become an important integral part of our daily lives, enabling us to contact our friends and families on time. Since the count of the users of social networks is increasing drastically, storage of such huge amount of data is difficult to accomplish. As a solution, Cloud provides a platform to store this tiny amount of data. More and more social network data has been made publicly available and analyzed in one way or another. However, the issues in securing the data and privacy of users in cloud-based social networks persist. Users are unaware of these issues. They share various pictures, videos and personal data on the networking site which prevail even after deletion. But some of the information revealed is meant to be private hence social network data has led to the risk of leakage of confidential information of individuals. This is because they collect huge personal data and users take risks of trusting them. Since more personalized information is shared with the public, violating the privacy of a target user become much easier. Hence, security of the

social networking data stored in the cloud is one of the major issues in cloud-based social networks. In this paper, we propose a framework for secure storing of data on the cloud-based social networks. The framework encrypts the data before storing it in the cloud, and the data is decrypted only with the private key of the user, making the data secure in the cloud. The proxy re-encryption scheme is used to re-encrypt the data to make it more secure.

CHAPTER 3

SYSTEM ANALYSIS

3.1 PROPOSED SYSTEM

The overall working of the proposed system will have a UI for any hospital through which the patients can book the slot for checkup and doctors can view those requests. Doctors need to request the patients for downloading the report which requires an OTP that's been sent on to the patient's email. As for storage once the patient uploads the report is divided into chunks of the file and these chunks are encrypted using AES, these chunks of data are distributed among 2 buckets. These buckets store the chunks of file randomly distributed among them and have an interlink to form a chain within the chunks.

Advantages:

- No chance of data loss.
- Highly secured
- More trustable.

3.2 PROPOSED METHOD

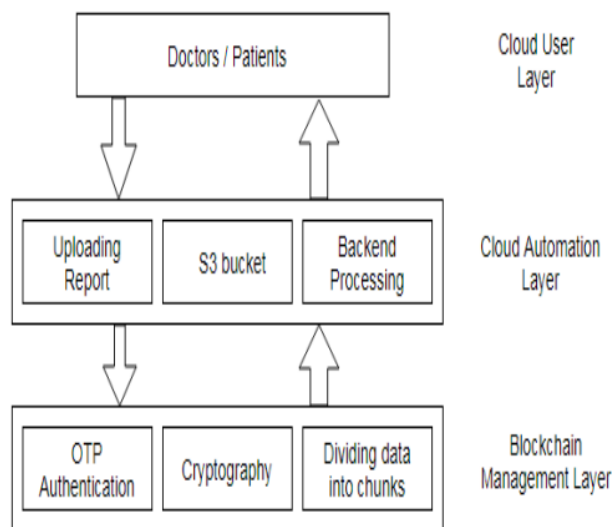


Fig: 3.2.1 Proposed Method Architecture

3.3 Feasibility Study:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ☐ **Economical Feasibility**
- ☐ **Technical Feasibility**
- ☐ **Social Feasibility**

Economical Feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available.

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 on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social 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. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional Requirements:

In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behaviour, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

Functional software requirements help you to capture the intended behaviour of the system. This behaviour may be expressed as functions, services or tasks or which system is required to perform

4.2 Non-functional requirements:

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

4.3 Hardware requirements:

For developing the application, the following are the Hardware Requirements:

- 1. Processor : I3/Intel Processor
- Hard Disk : 160GB

4.4 Software requirements (SRS):

For developing the application, the following are the Software Requirements:

- Operating System : Windows 7/8/10
- IDE : Pycharm
- Server side scripts : HTML, CSS, Js
- Libraries Used : Numpy, IO, OS, Random, Flask
- Technology : Python 3.6+

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The system architecture gives an overview of the working of the system.

The working of this system is described as follows:

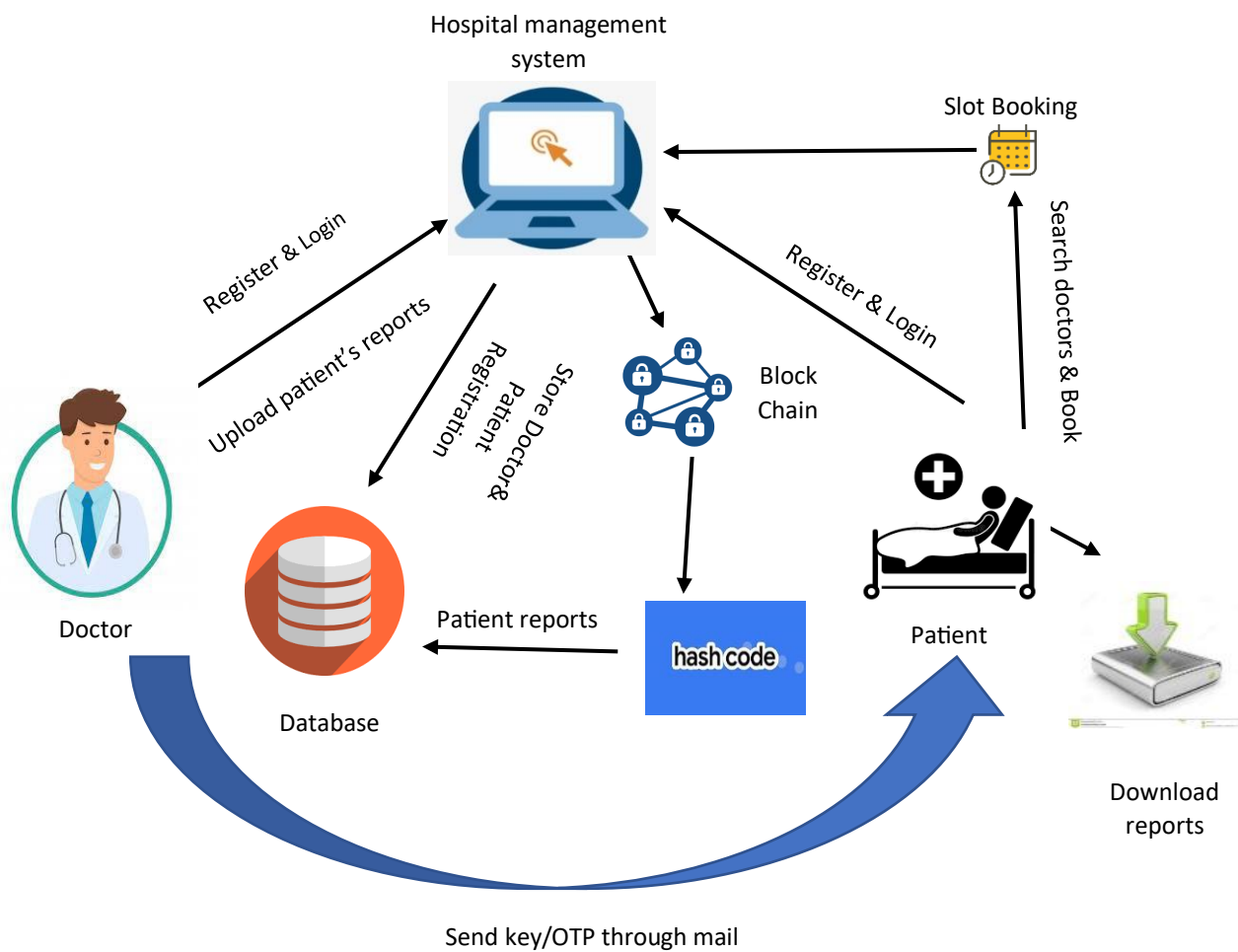


Fig 5.1 System Architecture

5.2 PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python class and objects

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g., a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

Here’s a brief list of Python OOP ideas:

- The class statement creates a class object and gives it a name. This creates a new namespace.
- Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: `ClassName.Attribute`.
- Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.
- Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copy’s part comes in.

- Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.
- Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

5.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

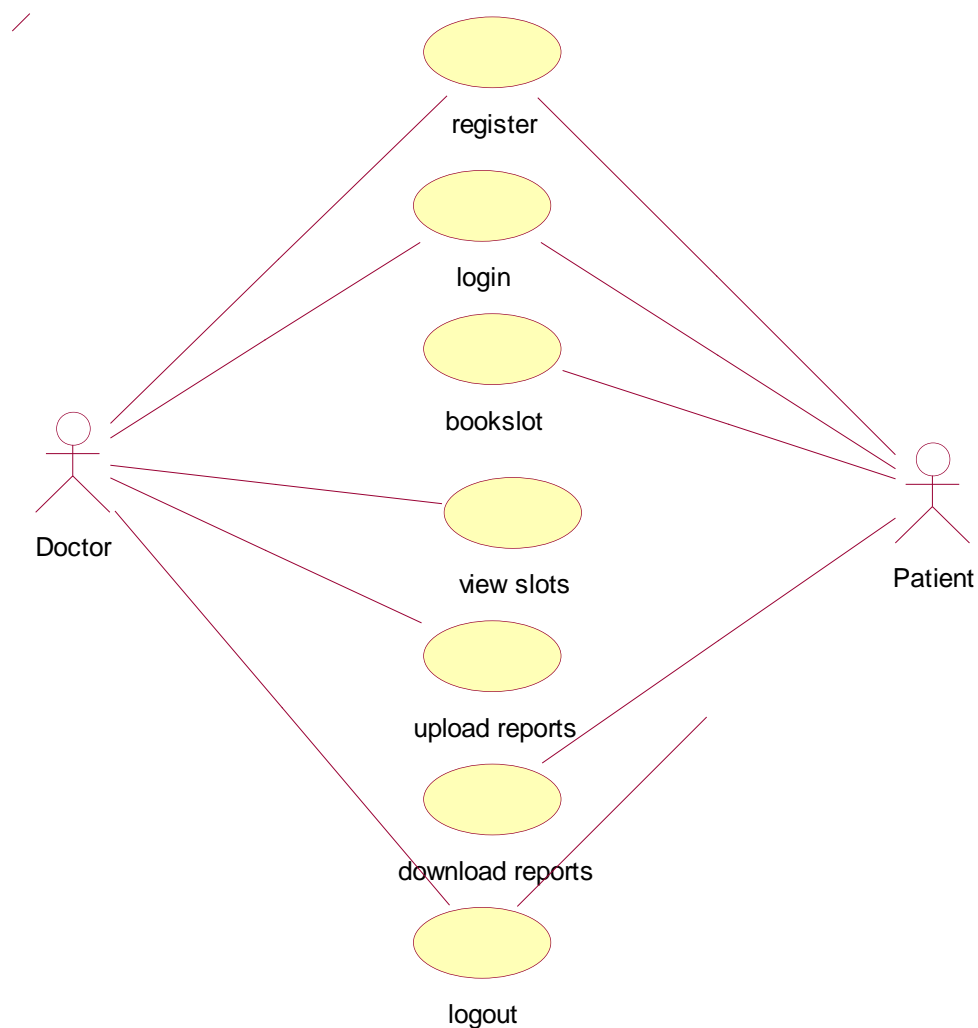


Fig 5.3.1 Use Case Diagram

CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

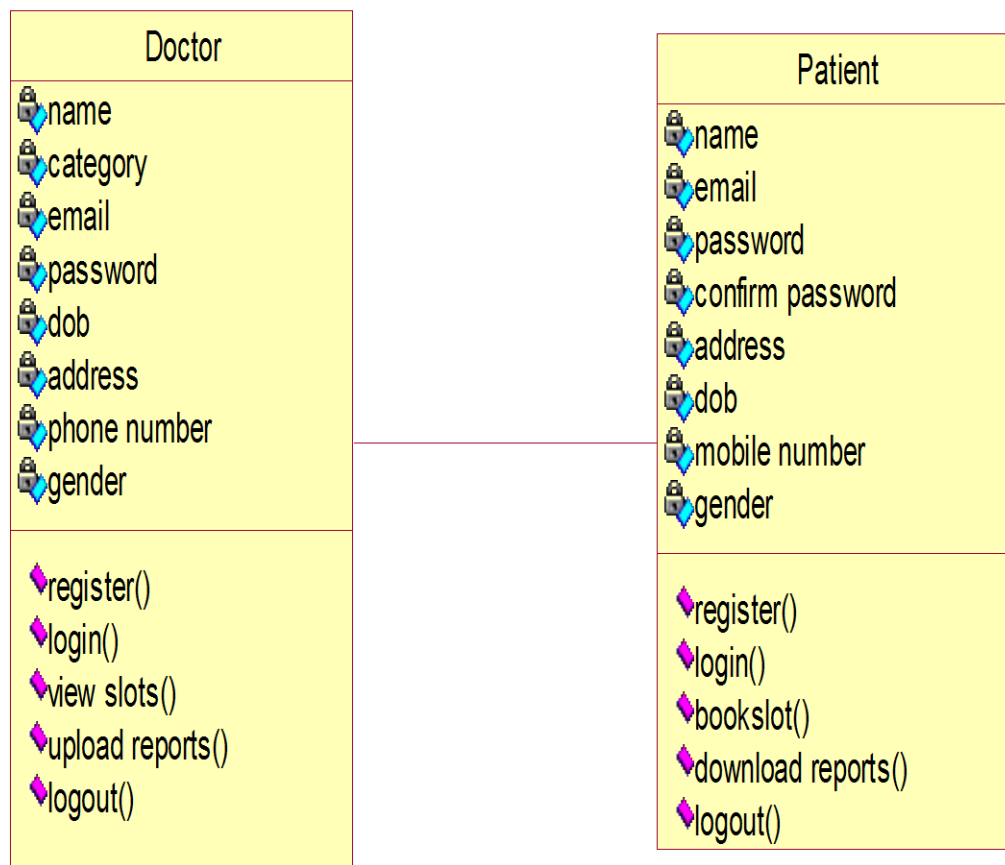


Fig 5.3.2 Class Diagram

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

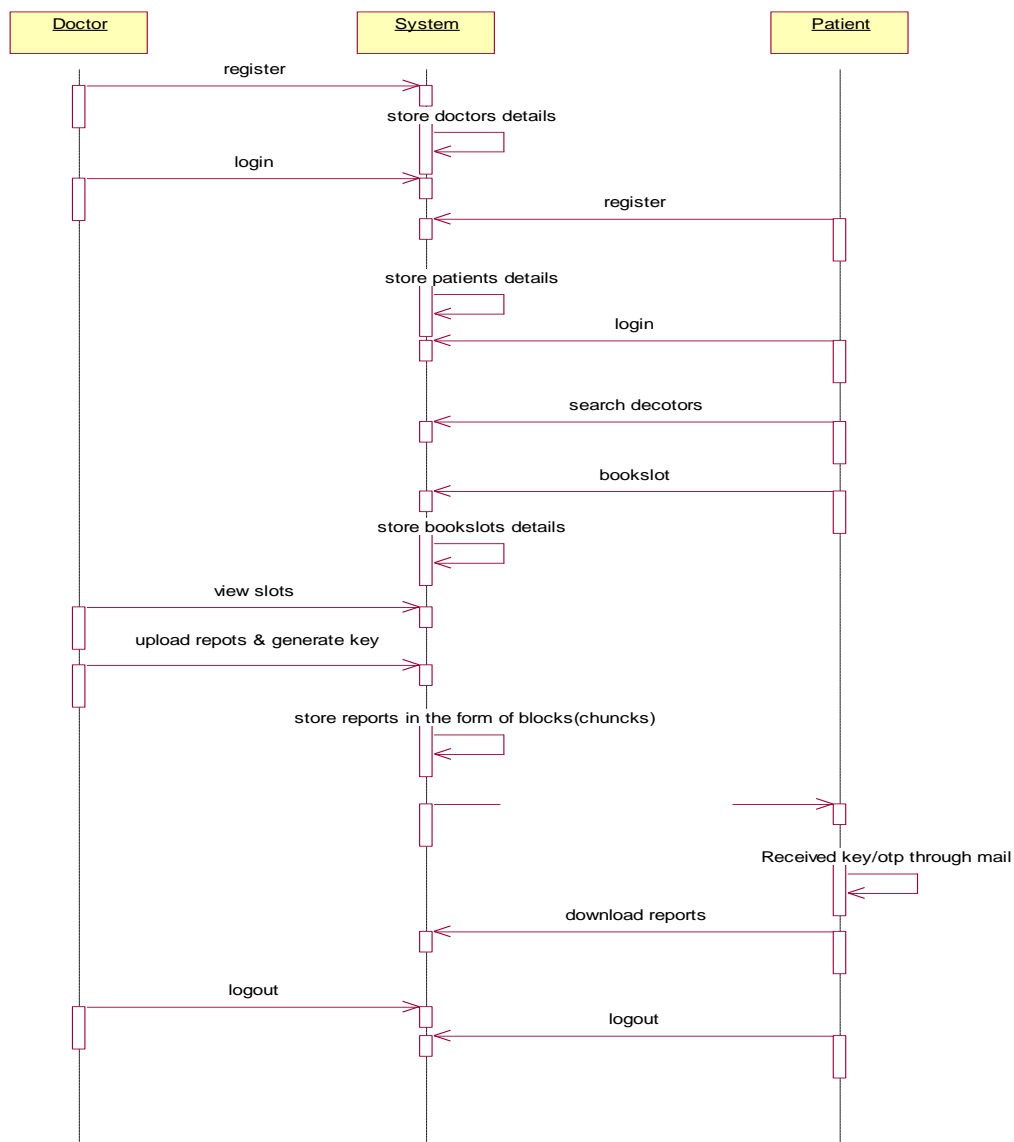


Fig: 5.3.3. Sequence Diagram

COLLABORATION DIAGRAM:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.

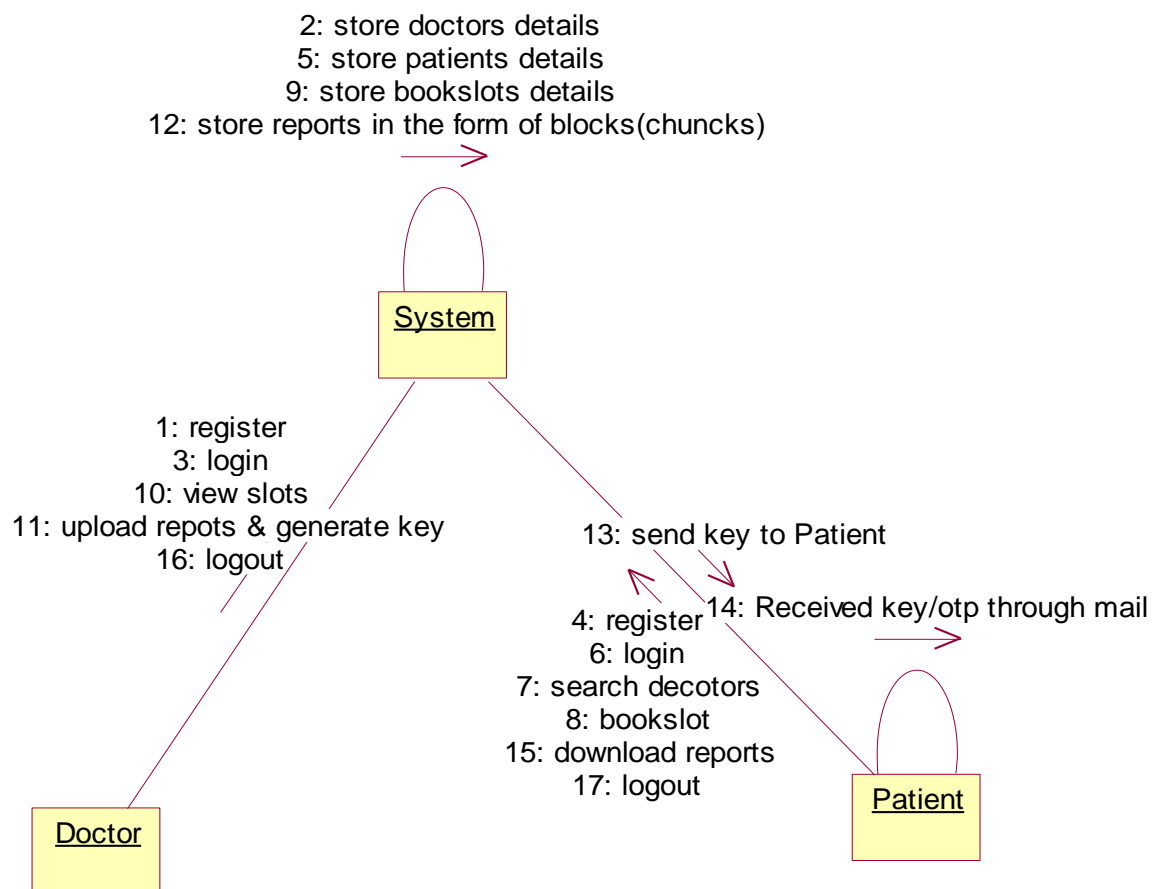


Fig 5.3.4. Collaboration Diagram

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

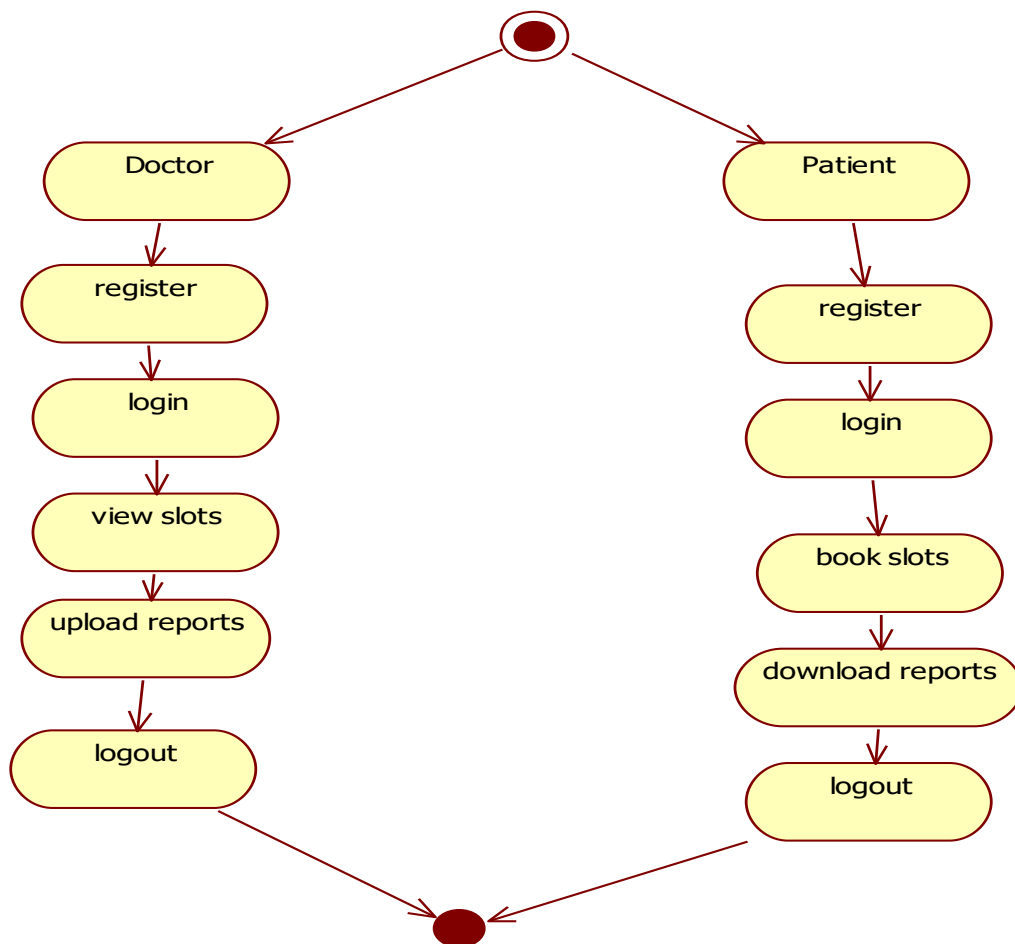


Fig: 5.3.5 Activity Diagram

COMPONENT DIAGRAM

Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executable, libraries etc. So, the purpose of this diagram is different, Component diagrams are used during the implementation phase of an application. But it is prepared well in advance to visualize the implementation details. Initially the system is designed using different UML diagrams and then when the artifacts are ready component diagrams are used to get an idea of the implementation.

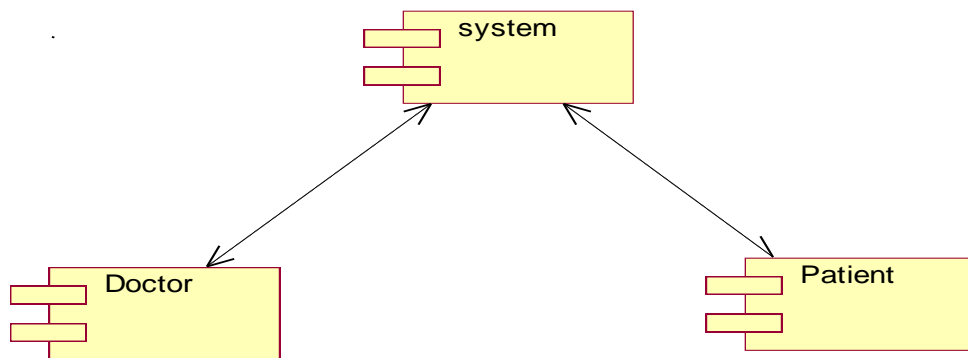


Fig: 5.3.6 Component Diagram

DEPLOYMENT DIAGRAM

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hard wares used to deploy the application.

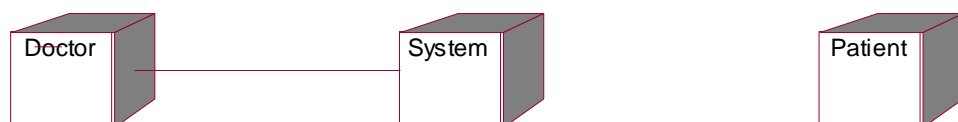


Fig 5.3.7 Deployment Diagram

ER DIAGRAM:

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.

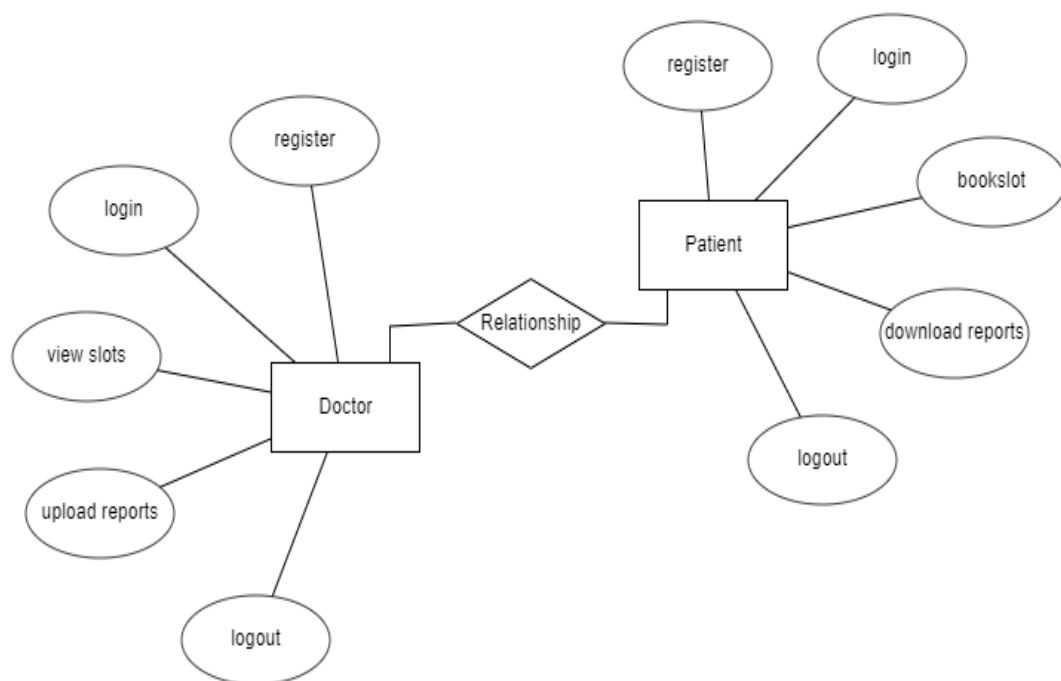


Fig: 5.3.8 ER Diagram

DFD DIAGRAM:

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

CONTEXT DIAGRAM:

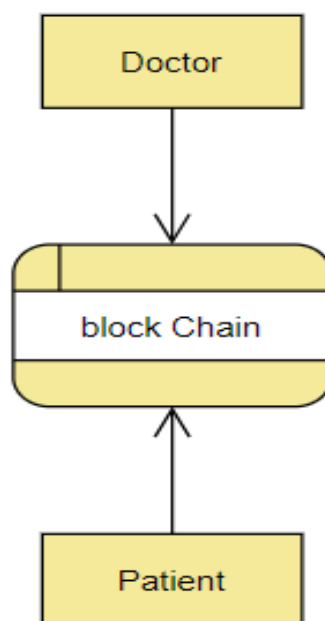


Fig 5.3.9 Context Diagram

LEVEL -1 DIAGRAM:

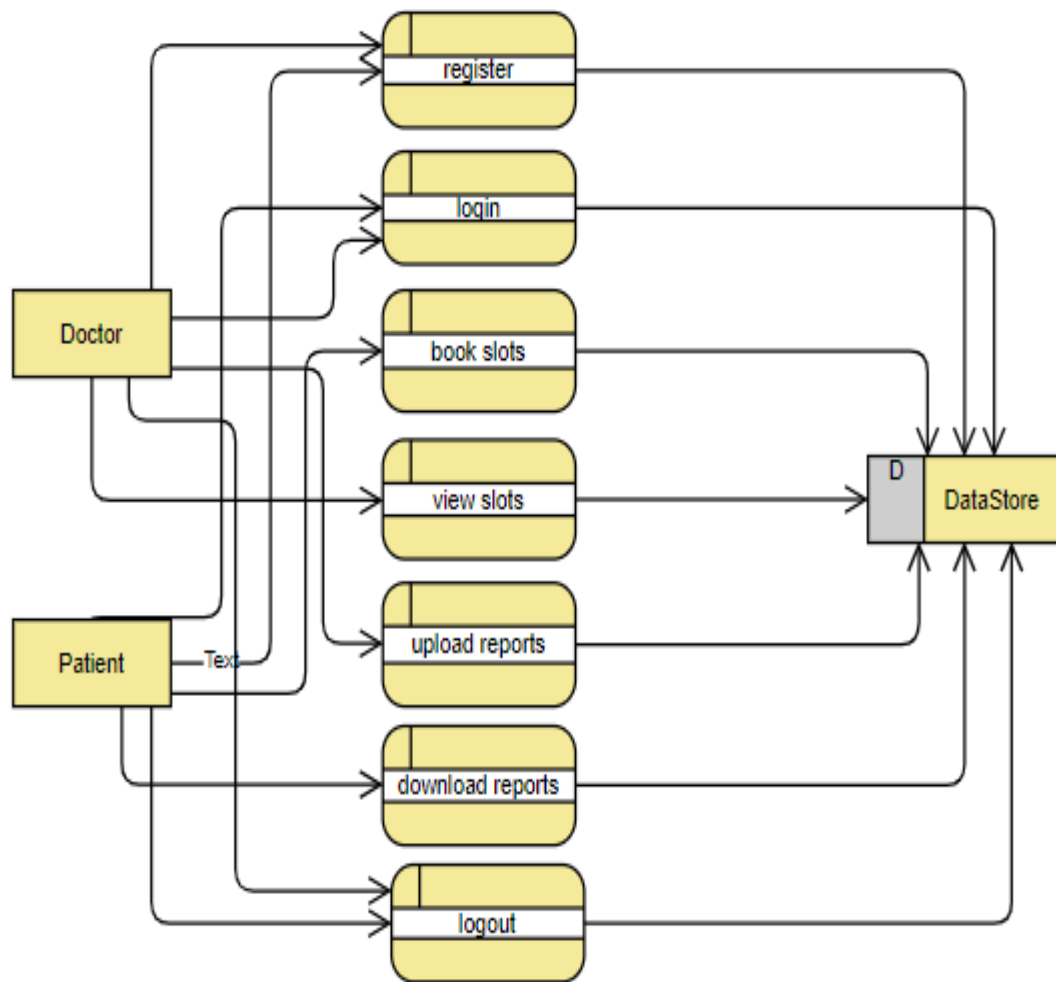


Fig 5.3.10 Level - 1 Diagram

LEVEL - 2 DIAGRAM:

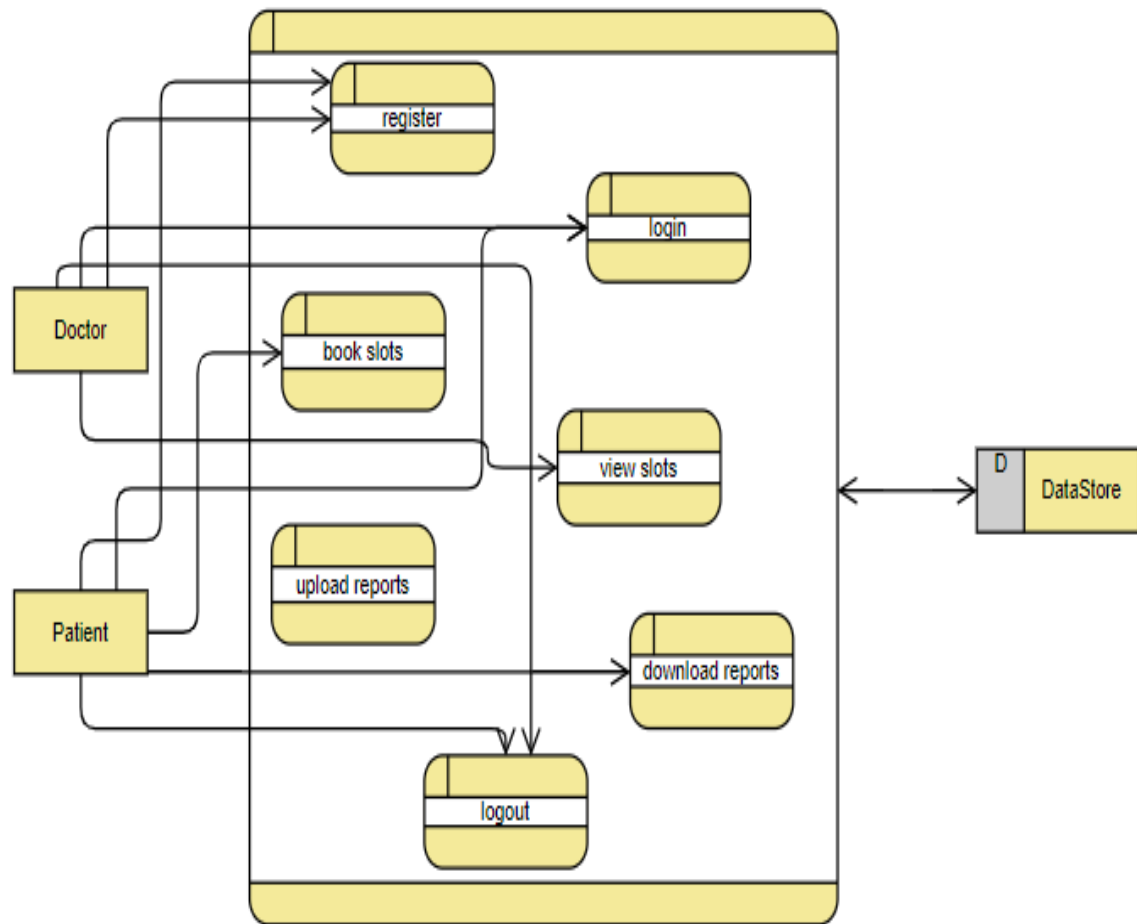


Fig 5.3.11 Level - 2 Diagram

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Project Modules:

6.1.1 Doctor Module:

Doctor needs to be register and then login in to the website. After login they can view the all slots are booked by patients. Based on the following dates booked by the patients, doctor will provide the checkup. After completion of the checkup, they can upload the patients reports. Those reports are stored in the database in the form of blocks. The uploaded reports will be divided in to chunks and each chunk is assigned with a hash code.

6.1.2 Patient Module: Here patient can register and login to the website. After login he/she can search the doctors based on their specialization. If the required doctors are available then the patient will book the slot. After the completion of patient's checkup by the concerned doctor, the doctor will send the reports in an email which consists of a key/ OTP, the patient will download the reports.

6.2 Methodology (Algorithms):

The encryption process uses a set of specially derived keys called round keys. These are applied, along with other operations, on an array of data that holds exactly one block of data. The data to be encrypted. This array we call the state array.

You take the following AES steps of encryption for a 128-bit block:

- Derive the set of round keys from the cipher key.
- Initialize the state array with the block data (plaintext).
- Add the initial round key to the starting state array.
- Perform nine rounds of state manipulation.
- Perform the tenth and final round of state manipulation.
- Copy the final state array out as the encrypted data (ciphertext).
- The reason that the rounds have been listed as "nine followed by a final tenth round" is because the tenth round involves a slightly different manipulation from the others.

The block to be encrypted is just a sequence of 128 bits. AES works with byte quantities so we first convert the 128 bits into 16 bytes. We say "convert," but, in reality, it is almost certainly stored this way already. Operations in RSN/AES are performed on a two-dimensional byte array of four rows and four columns. At the start of the encryption, the 16 bytes of data

6.2.1 Advanced Encryption Standard Algorithm (AES):

The Advanced Encryption Standard (AES) is a widely used symmetric encryption algorithm that employs a key-based approach to encrypt and decrypt data. The AES key generation algorithm follows a specific process to generate the encryption keys. Here's a simplified overview of the AES key generation algorithm:

1. **Key Size Selection:** Determine the desired key size for AES encryption. AES supports three key sizes: 128 bits, 192 bits, and 256 bits. The larger the key size, the stronger the encryption.
2. **Random Key Generation:** Generate a random key of the chosen size. The key should be generated using a cryptographically secure random number generator (CSPRNG) to ensure sufficient randomness and unpredictability.
3. **Key Expansion:** The AES key expansion algorithm takes the initial key and expands it into a set of round keys, which are used in the AES encryption and decryption process. The number of round keys generated depends on the key size:
 - For a 128-bit key, 10 round keys are generated.
 - For a 192-bit key, 12 round keys are generated.
 - For a 256-bit key, 14 round keys are generated.

The key expansion process involves applying various transformations to the initial key, such as substitution, permutation, and XOR operations.

4. **Final Round Key:** The last round key generated during the key expansion process is used in the final round of the AES encryption algorithm. It is not used in the previous rounds.

6.2.2 Secure Hash Algorithm:

The secure hash algorithm is family of cryptographic hash function. Sha 2 is a family of two similar hash functions with different block sizes known as SHA-256 and SHA- 512. They differ in the word size sha-256 uses 32-bit words.

The goal of any hash function is to produce digests that appear to be random to be considered cryptographically secure the hash function should meet two requirements first that it is impossible for an attacker to generate a message that matches a specific hash

value and second it should be impossible for an attacker to create two messages producing the exactly same hash value even slight change in the plain text should trigger a drastic difference in the two digests.

- The length of the clear text should be less than 2 to the power 64 bits in case of Sha-1 and Sha-256.
- The length of the hash digest should be 256 bits in the Sha-256 Algorithm.

SHA-256 Technical parameters:

- Block size indicator: 64 bytes
- Maximum Allowed message length: 33 bytes
- Characteristics of the message digest size: 32 bytes
- The standard word size: 4 bytes
- Internal position length parameter: 32 bytes
- The number of iterations in one cycle: 64

The internal state size of SHA-256 is 256. It is more secure than SHA-1. The output size of SHA-256 is 256 bits. It is hash function commonly used in block chain and it has 256 bits so it has proved security.

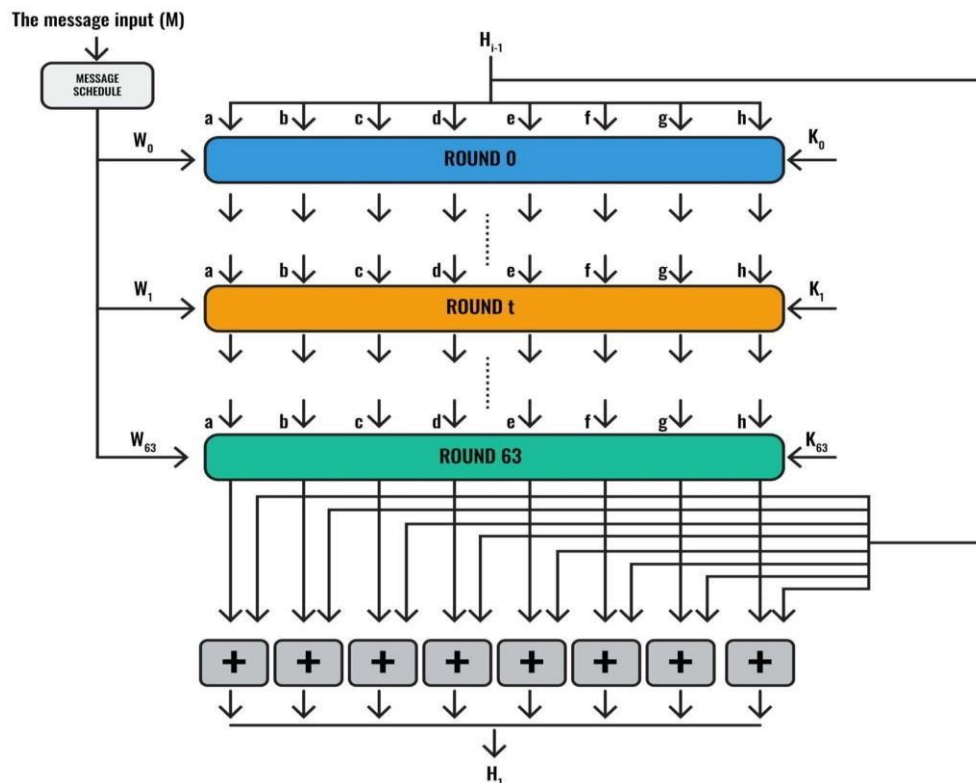


Fig 6.2.2 The SHA-256 algorithm for processing a single block

6.3 Sample Code:

- **Python code**

```
from flask import Flask, request, render_template, redirect,url_for,session,flash
import pandas as pd
import mysql.connector
import numpy as np
from flask_mail import *
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
app=Flask(__name__)
app.secret_key='Lakshmi'
import hashlib
import datetime
from datetime import datetime

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="",
    database="block_chain"
)
mycursor = mydb.cursor()
@app.route('/')
def index():
    return render_template("index.html")
@app.route('/doctor')
def doctor():
    return render_template("doctor.html")
@app.route('/doctorback',methods=['POST','GET'])
def doctorback():
    if request.method=='POST':
        print("gekjhiuth")
        name=request.form['name']
        gen=request.form['gen']
        email=request.form['email']
        pwd=request.form['pwd']
        dob=request.form['dob']
        addr=request.form['addr']
        cpwd=request.form['cpwd']
        pno=request.form['pno']
```

```

        dtype=request.form['dtype']
    else:
        flash("Invalid Email/Password ", "danger")
        return render_template('uolog.html', msg="invalid value")

    return render_template('uolog.html')

@app.route('/doctorhome')
def doctorhome():
    return render_template("doctorhome.html")

@app.route('/patient')
def patient():
    return render_template("register.html")

@app.route('/regback',methods=['POST','GET'])
def regback():
    if request.method=='POST':
        print("gekjhiuth")
        name=request.form['name']
        gen=request.form['gen']
        email=request.form['email']
        pwd=request.form['pwd']
        addr=request.form['addr']
        cpwd=request.form['cpwd']
        pno=request.form['pno']
        dob=request.form['dob']
        print(addr)

        sql="select * from patient"
        result=pd.read_sql_query(sql,mydb)
        email1=result['email'].values
        print(email1)
        if email in email1:
            flash("email already existed","success")
            return render_template('register.html')
        if(pwd==cpwd):
            sql = "INSERT INTO patient (name,email,pwd,gen,addr,dob,pno) VALUES"
            (%s,%s,%s,%s,%s,%s,%s,%s)"
            val = (name,email,pwd,gen,addr,dob,gen)
            mycursor.execute(sql, val)
            mydb.commit()
            flash("Successfully Registered","warning")
            return render_template('login.html')
        else:
            flash("Password and Confirm Password not same")

```

```

    return render_template('register.html')

@app.route('/login')
def login():
    return render_template("login.html")

@app.route('/loginback',methods=['POST', 'GET'])
def loginback():
    if request.method == "POST":

        email = request.form['email']

        password1 = request.form['pwd']
        print('p')

        sql = "select * from patient where email='%s' and pwd='%s' " % (email, password1)
        print('q')
        x = mycursor.execute(sql)
        print(x)
        results = mycursor.fetchall()

        print(results)
        global name
        session['email'] = email

        if len(results) > 0:
            flash("Welcome ", "primary")
            return render_template('patienthome.html', msg=results[0][1])
        else:
            flash("Invalid Email/Password ", "primary")
            return render_template('login.html', msg="invalid value")

    return render_template('login.html')

@app.route('/patienthome')
def patienthome():
    return render_template("patienthome.html")

@app.route("/search")
def search():
    return render_template("search.html")

@app.route("/searchback",methods=['POST','GET'])
def searchback():
    print("dfhlksokhso")
    if request.method == 'POST':

```

```

print("gekjhiuth")
#fname = request.form['fname']
dtype = request.form['dtype']

print("Reading BLOB data from python_employee table")

sql = "select * from doctor where dtype LIke '%" + dtype + "%' "
x = pd.read_sql_query(sql, mydb)
print("^^^^^^^^^^^^^^^^")
print(type(x))
print(x)
x = x.drop(['pwd'], axis=1)
x = x.drop(['dob'], axis=1)
x = x.drop(['addr'], axis=1)
return render_template("searchback.html", col_name=x.columns.values,
row_val=x.values.tolist())
return render_template("searchback.html")

@app.route("/bookslot/<s1>/<s2>/<s3>/<s4>/<s5>")
def bookslot(s1=0,s2="",s3="",s4="",s5=""):
    global g,f1,a1,a2,a3
    g=s1
    f1=s2
    a1=s3
    a2=s4
    a3=s5
    return render_template("bookslot.html",g=g,f1=f1,a1=a1,a2=a2,a3=a3)

@app.route('/bookslotback',methods=['POST','GET'])
def bookslotback():
    if request.method=='POST':
        name = request.form['name']
        id = request.form['id']
        email = request.form['email']
        pname = request.form['pname']
        sym = request.form['sym']
        age = request.form['age']
        pno = request.form['pno']
        dtype = request.form['dtype']
        date = request.form['date']
        print(date)
        date2 = datetime.now().strftime('%Y-%m-%d')
        mycursor = mydb.cursor()
        pemail = session.get('email')
        if date >= date2:

```



```

        sql = "insert into bookslot(dname,pname,demail,pemail,sym,age,dtype,dno,date)
values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
        print("dfhlksokhso")
        print("Reading BLOB data from python_employee table")
        email = session.get('email')
        print(email)
        sql = "select * from bookslot where status='Completed' and demail='%s' and
action='waiting' "%(email)
        x = pd.read_sql_query(sql, mydb)
        print("^^^^^^^^^^^^^^^^")
        print(type(x))
        print(x)
        x = x.drop(['dname'], axis=1)
        x = x.drop(['demail'], axis=1)
        x = x.drop(['dtype'], axis=1)
        x = x.drop(['status'], axis=1)
        x = x.drop(['dno'], axis=1)
        x = x.drop(['action'], axis=1)

```

```

        return render_template("upload.html", col_name=x.columns.values,
row_val=x.values.tolist())

```

```

@app.route("/upback/<s1>/<s2>/<s3>/<a>")
def upback(s1="",s2="",s3="",a=0):
    global g,fl,al

```

```

        return render_template("upback.html",g=s1,fl=s2,al=s3,a=a)

```

```

@app.route("/upback1",methods=["POST","GET"])
def upback1():
    if request.method=="POST":
        pname = request.form['pname']
        id = request.form['id']
        pemail = request.form['pemail']
        sym = request.form['sym']
        file =request.form['file']
        # email = session.get('email')

        dd = "text_files/" + file
        f = open(dd, "r")
        data = f.read()

        # print(data)
        dataleee=len(data)

```

```

datalen=int(len(data)/3)
print(datalen,len(data))
g=0
a = "
b = "
c = "
for i in range(0,2):
    if i==0:
        a=data[g: datalen:1]
        print(a)
        print("=====")
        result = hashlib.sha1(a.encode())
        hash1 = result.hexdigest()
        print(hash1)
        print("+++++")

    print(g)
    print(len(data))
    c=data[g: len(data):1]
    print(c)
    print("=====")

    print("*****")
    result = hashlib.sha1(c.encode())
    hash2 = result.hexdigest()
    print(hash2)

    from datetime import datetime
    now = datetime.now()
    currentDay = datetime.now().strftime('%Y-%m-%d')
    sql = "INSERT INTO reports
(fid,block1,block2,hash1,hash2,pname,pemail,sym,date) VALUES
(%s,%s,%s,%s,%s,%s,%s,%s,%s)"
    val = (id, a, c, hash1, hash2, pname,pemail,sym,now)
    mycursor.execute(sql, val)
    mydb.commit()
    # otp = "Dear "
    # msg='Here I am sending your reports pls refer it.'
    # m1="You can download your health report now."
    # m2='Hash values for downloading the file.'
    #
    # mail_content = otp + pname+ ','+msg + m1 + currentDay+'\n'+m2+hash1+' and '+
hash2
    # sender_address = 'cse.takeoff@gmail.com'
    # sender_pass = 'Takeoff@123'

```

```

# receiver_address = pemail
# message = MIMEMultipart()
# message['From'] = sender_address
# message['To'] = receiver_address
# message['Subject'] = 'Enhancing the Data Security in Cloud using Block Chain'
#
# message.attach(MIMEText(mail_content, 'plain'))
# session = smtplib.SMTP('smtp.gmail.com', 587)
# session.starttls()
# session.login(sender_address, sender_pass)
# text = message.as_string()
# session.sendmail(sender_address, receiver_address, text)
# session.quit()

status = 'Success'
sql = "update bookslot set action='%s' where id='%s' " % (status, id)
mycursor.execute(sql)
mydb.commit()

sql = "select * from reports where fid='%s' " % (id)
x = pd.read_sql_query(sql, mydb)
print("^^^^^^^^^^^^^^^^^^^^")
print(type(x))
print(x)
x = x.drop(['pname'], axis=1)
# x = x.drop(['demail'], axis=1)
x = x.drop(['pemail'], axis=1)
x = x.drop(['block1'], axis=1)
x = x.drop(['block2'], axis=1)
x = x.drop(['date'], axis=1)
x = x.drop(['fid'], axis=1)
x = x.drop(['sym'], axis=1)
flash("file uploaded successfully", "success")
return render_template("upback1.html", col_name=x.columns.values,
row_val=x.values.tolist())
@app.route('/viewreport<s1>/<s2>/<s3>')
def viewreport(s1=0,s2="",s3=""):
    return render_template('viewreport.html',s1=s1,s2=s2,s3=s3)
@app.route("/reportback",methods=['POST','GET'])
def reportback():
    print("dfhlksokhso")
    if request.method == 'POST':
        print("gekjhiuth")
        hash1 = request.form['hash1']
        id = request.form['id']
        hash2 = request.form['hash2']

```

```

        sql = "select count(*),CONCAT(block1,block2,) from reports where
hash1='"+hash1+"' and hash2='"+hash2+"'
        x = pd.read_sql_query(sql, mydb)
        count=x.values[0][0]
        print(count)
        assss=x.values[0][1]
        assss=assss.decode('utf-8')
        print("^^^^^^^^^^^^^^^^")
        return render_template("reportback.html", msg=assss)

```

```

@app.route("/down")
def down():
    email = session.get('email')
    sql = "select * from reports where pemail='%s'" % (email)
    x = pd.read_sql_query(sql, mydb)

```

```

    x = x.drop(['block1'], axis=1)
    x = x.drop(['fid'], axis=1)
    x = x.drop(['block2'], axis=1)
    x = x.drop(['pemail'], axis=1)
    x = x.drop(['hash1'], axis=1)
    x = x.drop(['hash2'], axis=1)
    x = x.drop(['pname'], axis=1)
    # x = x.drop(['date'], axis=1)

```

```

    # x["View Data"] = " "
    # x["Send Request"] = ""

```

```

    return render_template("down.html", col_name=x.columns.values,
row_val=x.values.tolist())

```

```

@app.route("/download/<s1>/<s2>/<s3>")
def download(s1=0,s2="",s3=""):
    global g,fl,a1
    g=s1
    fl=s2
    a1=s3
    return render_template("download.html",g=g,fl=fl,a1=a1)

```

```

@app.route("/downfile",methods=['POST','GET'])
def downfile():
    print("dfhlksokhso")
    if request.method == 'POST':
        print("gekjhiuth")

```

```

hash1 = request.form['hash1']
id = request.form['id']
hash2 = request.form['hash2']

sql = "select count(*),CONCAT(block1,block2,) from reports where
hash1='"+hash1+"' and hash2='"+hash2+"'
x = pd.read_sql_query(sql, mydb)
count=x.values[0][0]
print(count)
asss=x.values[0][1]
# asss=asss.decode('utf-8')

# print("^^^^^^^^^^^^^^^^")
# if count==0:
#     flash("Enter Valid Hash Values","danger")
#     return render_template("downfile.html")
# if count==1:
return render_template("downfile.html", msg=asss)

return render_template("downfile.html")

if __name__ == '__main__':
    app.run(debug=True)

```

- **Database Code**

```

DROP TABLE IF EXISTS `bookslot`;

CREATE TABLE `bookslot` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `dname` varchar(100) DEFAULT NULL,
  `pname` varchar(100) DEFAULT NULL,
  `demail` varchar(100) DEFAULT NULL,
  `pemail` varchar(100) DEFAULT NULL,
  `sym` varchar(100) DEFAULT NULL,
  `age` varchar(100) DEFAULT NULL,
  `dtype` varchar(100) DEFAULT NULL,
  `dno` varchar(100) DEFAULT NULL,
  `date` varchar(100) DEFAULT NULL,
  `status` varchar(100) DEFAULT 'pending',
  `action` varchar(100) DEFAULT 'waiting',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;

/*Data for the table `bookslot` */

insert into
`bookslot`(`id`,`dname`,`pname`,`demail`,`pemail`,`sym`,`age`,`dtype`,`dno`,`date`,`status`,`a
ction`) values (1,'Lakshmi','Keerthana','lakshmi@gmail.com','cse.@gmail.com','Heartburn
pain','26','Cardiologist','5678909876','2021-06-12','Completed','Success');

/*Table structure for table `doctor` */

DROP TABLE IF EXISTS `doctor`;

CREATE TABLE `doctor` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) DEFAULT NULL,
  `dtype` varchar(100) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `pwd` varchar(100) DEFAULT NULL,
  `dob` varchar(100) DEFAULT NULL,
  `addr` varchar(100) DEFAULT NULL,

```

```

`pno` varchar(100) DEFAULT NULL,
`gen` varchar(100) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;

/*Data for the table `doctor` */

insert into `doctor`(`id`,`name`,`dtype`,`email`,`pwd`,`dob`,`addr`,`pno`,`gen`) values
(1,'lakshmi','Cardiologist','lakshmi@gmail.com','1','2021-06-06','ongole','5678909876','female');

/*Table structure for table `patient` */

DROP TABLE IF EXISTS `patient`;
CREATE TABLE `patient` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `pwd` varchar(100) DEFAULT NULL,
  `gen` varchar(100) DEFAULT NULL,
  `addr` varchar(100) DEFAULT NULL,
  `dob` varchar(100) DEFAULT NULL,
  `pno` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;

/*Data for the table `patient` */

insert into `patient`(`id`,`name`,`email`,`pwd`,`gen`,`addr`,`dob`,`pno`) values
(1,'keerthana','cse.takeoff@gmail.com','1','ms','tirupati','2021-06-09','ms');

```

```

/*Table structure for table `reports` */

```

```

DROP TABLE IF EXISTS `reports`;

```

```
CREATE TABLE `reports` (  
  `id` int(10) NOT NULL AUTO_INCREMENT,  
  `fid` int(10) DEFAULT NULL,  
  `block1` longblob,  
  `block2` longblob,  
  `hash1` varchar(200) DEFAULT NULL,  
  `hash2` varchar(200) DEFAULT NULL,  
  `pname` varchar(100) DEFAULT NULL,  
  `pemail` varchar(100) DEFAULT NULL,  
  `sym` varchar(100) DEFAULT NULL,  
  `date` varchar(100) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
```


6.4 Results

6.4.1. INPUTS

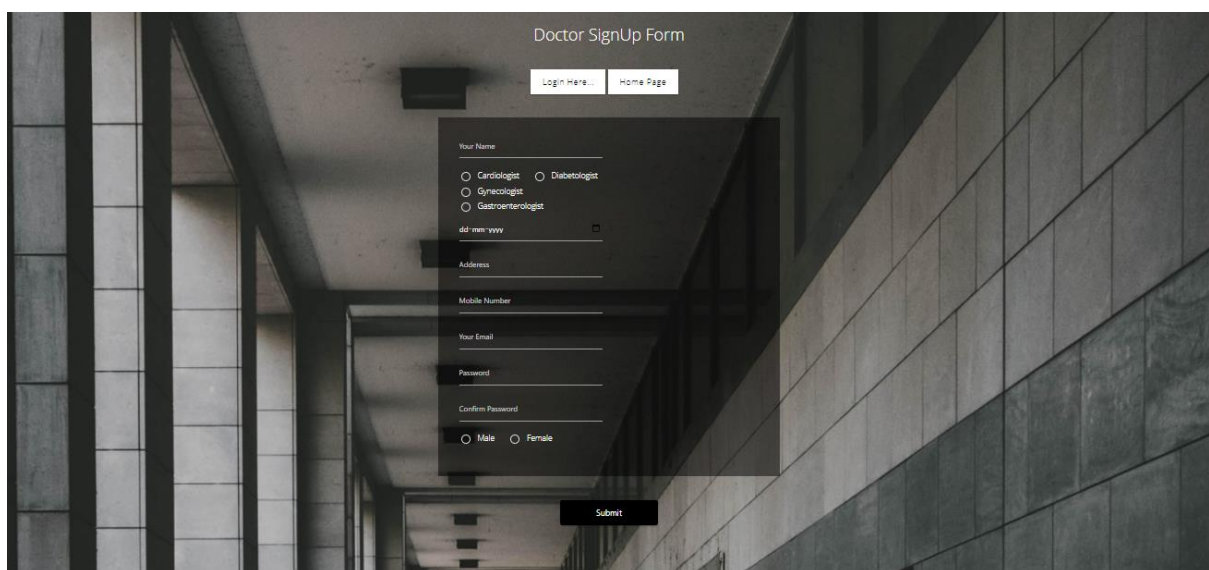
HOME PAGE:

This is the home page of the “Enhancing the data security in cloud using Block Chain. It consists of 3 modules namely Home, Doctor, Patient.



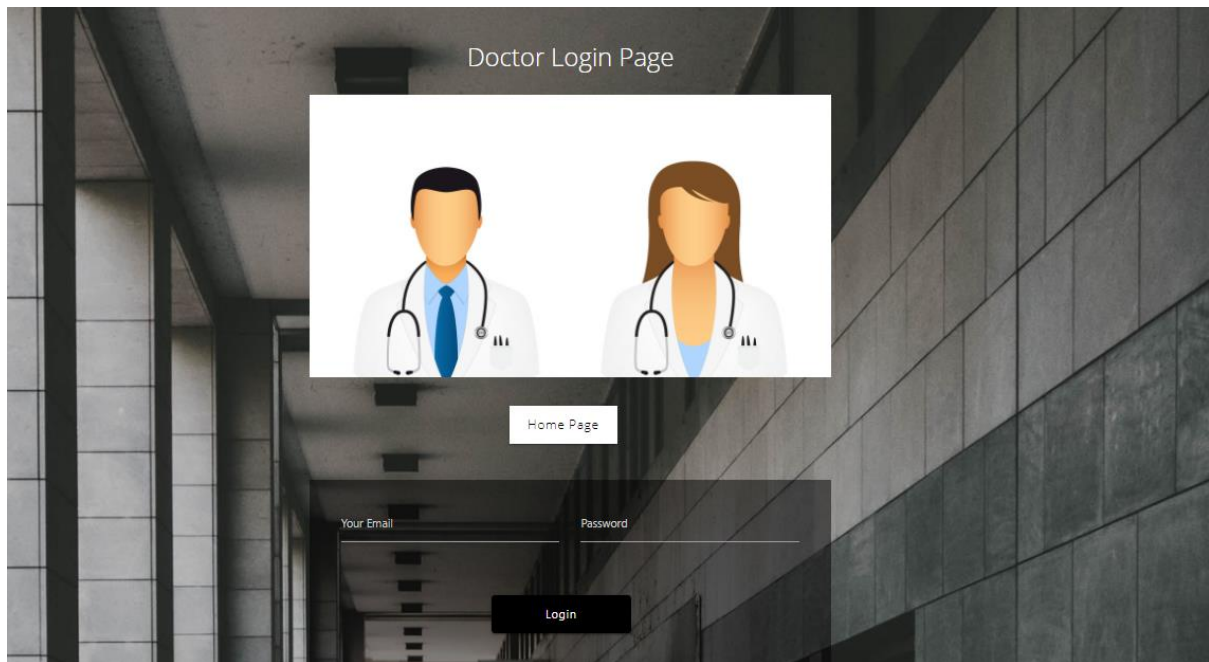
DOCTOR REGISTRATION PAGE:

This is the doctor's registration page. Here the doctor needs to fill their particular details and set password to their account.



DOCTOR LOGIN PAGE:

This is the doctor login page. In this page the doctor will login using their email and password according to their availability.



DOCTOR HOME PAGE:

This is the doctor home page in this page the doctor will view the slots and upload the patients reports



CHECKUP STATUS PAGE:

This is the checkup status page. In this it tells the status of the checkup whether completed or not.



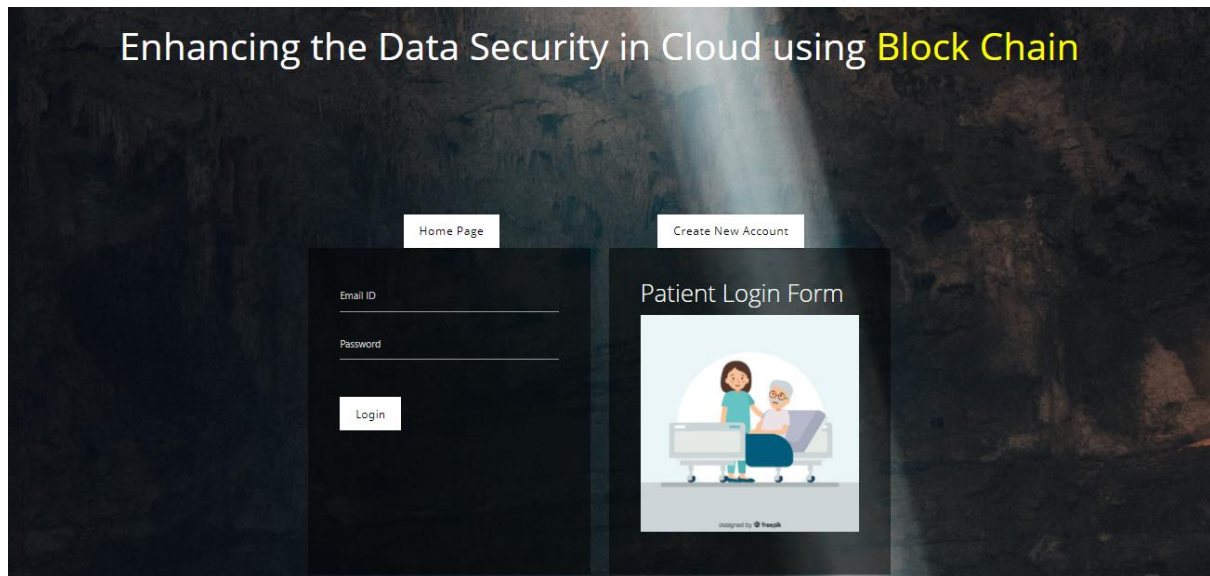
PATIENT REGISTRATION PAGE:

This is the patient registration page. Where the patient needs to fill their required details.

A screenshot of a web page titled "Patient Registration Page" with a dark, textured background. On the left side, there is a registration form with the following fields: a gender selection row with radio buttons for "Mr.", "Ms.", and "Mrs."; "First Name", "Email", "Password", "Confirm Password", "Mobile", "Address", and "dd-mm-yyyy" (with a calendar icon). A white "REGISTER" button is at the bottom of the form. On the right side, there is a circular profile picture placeholder showing a person with dark hair and a blue shirt. Below the profile picture are two buttons: "Login Here..." and "Home Page".

PATIENT LOGIN PAGE:

This is the patient login page. Here the patient will be logged in using their email ID and password.



PATIENT HOME PAGE:

This is the patient's home page. Here the patient can book their slots and download the reports.



6.4.2 OUTPUTS

SEARCHING DOCTORS PAGE:

This is the doctor's search page. Patient can search the doctors of their convenience to get the appointment.



Enhancing the Data Security in Cloud using Block Chain

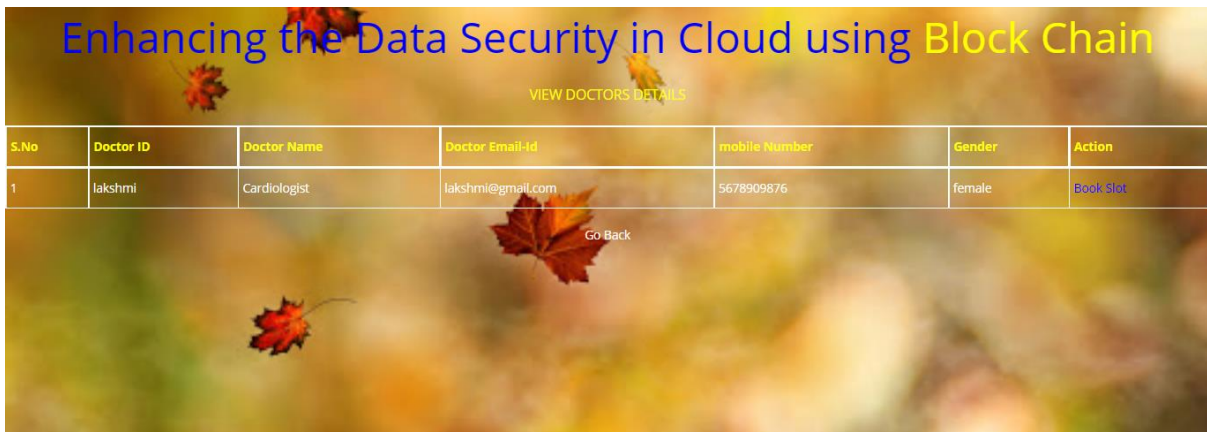
Search Doctors here...

Cardiologist

Search Reset

VIEW DOCTORS DETAILS:

This the page where we can find the details of the available doctors to book slot. The details of the doctor will be given with the specialization.



Enhancing the Data Security in Cloud using Block Chain

VIEW DOCTORS DETAILS

S.No	Doctor ID	Doctor Name	Doctor Email-Id	mobile Number	Gender	Action
1	lakshmi	Cardiologist	lakshmi@gmail.com	5678909876	female	Book Slot

Go Back

SLOT BOOKING PAGE:

This is the slot booking page. Here the patient can book their slot according to the availability of the doctor.

Enhancing the Data Security in Cloud using **Block Chain**

Book Slot here...

Keerthana

28

Heartburn pain

18-06-2021

Submit

FILE DOWNLOAD PAGE:

This is the file download page. Where the symptoms of the patients and reports can be downloaded.

Enhancing the Data Security in Cloud using **Block Chain**

VIEW DOCTORS DETAILS

S.No	Hash1	Hash2	Symptoms	Action
3	9a611ae3732b13d72ae219f89459089819a87580	37fecf3e19620e14a9fe45a3f763a3077efb6835	Heartburn pain	Download

Go Back

ENTER HASH VALUES:

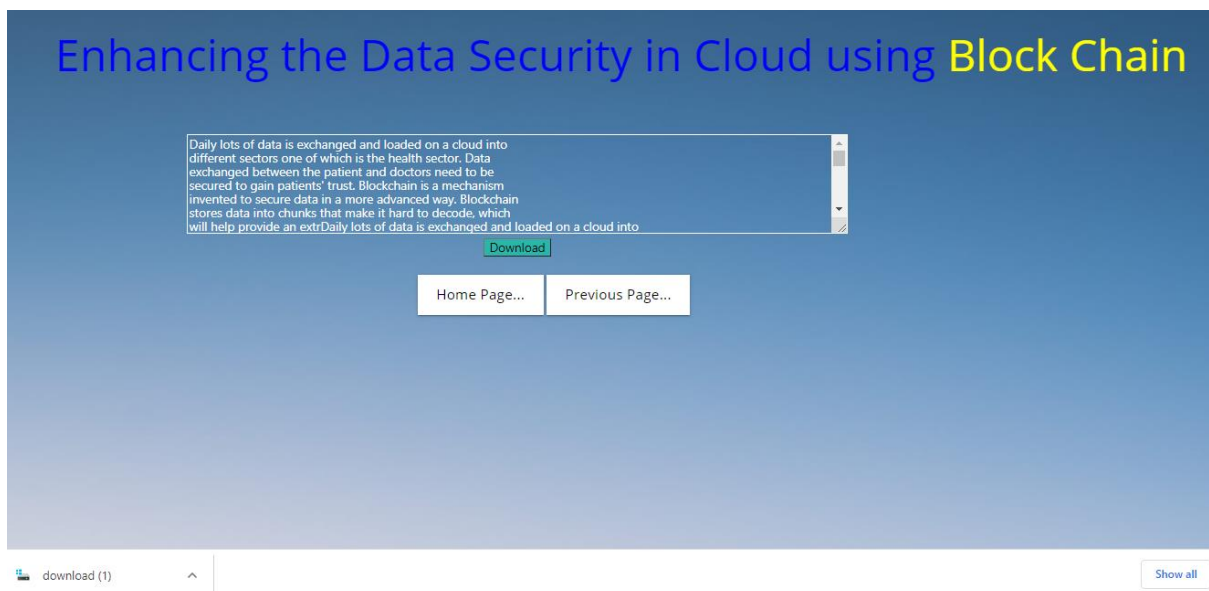
This is the authentication page. Here both the values of patient's device and doctor's device need to be entered.



The screenshot shows a web page with a pink background. At the top, the title "Enhancing the Data Security in Cloud using Block Chain" is displayed in blue and yellow text. Below the title, the instruction "Enter Hash Values here..." is shown in green. There are two input fields: the first contains the hash "9a611ae3732b13d72ae21" and the second contains "37fecf3e19620e14a9fe45". A "Submit" button is located below the second input field.

FILE DOWNLOAD PAGE:

This is the file download page. Here the patient can download the reports.



The screenshot shows a web page with a blue gradient background. At the top, the title "Enhancing the Data Security in Cloud using Block Chain" is displayed in blue and yellow text. Below the title, there is a text box containing a paragraph about data security and blockchain. A green "Download" button is positioned below the text box. Below the "Download" button, there are two buttons: "Home Page..." and "Previous Page...". At the bottom of the page, there is a "download (1)" link and a "Show all" button.

CHAPTER 7

SYSTEM TESTING

Testing Methods:

It's important to note that while blockchain can enhance data security in the cloud, it is not a silver bullet solution. It should be complemented with other security measures such as strong access controls, encryption, and regular security updates to provide a comprehensive security framework.

1. Penetration Testing:

Conduct regular penetration testing to identify vulnerabilities in the cloud infrastructure, applications, and smart contracts deployed on the blockchain. This helps ensure that potential security weaknesses are identified and remediated.

2. Blockchain Consensus Mechanism Testing:

Blockchain networks rely on consensus mechanisms to validate and secure transactions. It is essential to thoroughly test the consensus algorithm to identify any potential weaknesses or vulnerabilities that could compromise the security of the data stored in the cloud.

3. Privacy and Encryption Testing:

Blockchain technology does not inherently provide privacy for the data stored in the blockchain. Additional encryption and privacy measures must be implemented to protect sensitive information. Testing should include verifying the encryption algorithms, key management, and data access controls to ensure that data privacy is maintained.

4. Performance and Scalability Testing:

Evaluate the performance and scalability of the blockchain network to handle a growing number of transactions and users. Stress testing and load testing should be performed to identify any bottlenecks or limitations in the system that could impact data security.

5. Security Incident Response Testing:

Prepare and test an incident response plan specifically designed for blockchain-based cloud environments. Simulate security incidents and evaluate the response capabilities to ensure effective detection, containment, and recovery procedures are in place.

6. Smart Contract Audits:

Smart contracts are integral to blockchain-based applications. Performing audits on smart contracts can help identify coding errors, vulnerabilities, and potential security flaws. Audits can be conducted by specialized security firms to assess the smart contract code for potential risks and ensure the security of transactions and data.

CHAPTER 8

CONCLUSION

The main motive of our work is to create a trustworthy, efficient and real-time system for storing patients report in more efficient way, healthcare sectors rarely focus onto storing data into the cloud and maintain the security of data and mostly prefer sticking to have copies of the report, Our system not only lowered the task of maintaining hard copy records but also provided a security mechanism adapted through blockchain mechanism which will benefit a lot in maintain HIPAA Act more efficiently and help in achieving growth in the way of working of health sectors.

We propose SecNet, a new network paradigm focused on secure data, to utilize AI and blockchain to solve data abuse and enable AI to reliably manage data in an insecure environment through the blockchain. Store, share, and compute instead of exchanging data. SecNet is a blockchain technology and AI-based secure computing platform to ensure data ownership, a blockchain-based incentive mechanism to provide the paradigm and incentives for data convergence, and provide stronger AI for, ultimately better networking Provides security. It also discusses, a common use case for SecNet in healthcare systems, and provides an alternative way to use the SecNet storage capabilities.

It also evaluates the improvement of network vulnerabilities in response to DDoS attacks, and analyzes the creative aspects that encourage users to share security rules for a more secure network. Future work on will look at using the blockchain to grant access to data requests and design secure and detailed smart contracts for data exchange and AI-powered computing services on SecNet. It also simulates SecNet and analyzes its performance through extensive experimentation on advanced platforms.

CHAPTER 9

REFERENCES

- [1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. Consulted, 2008.
- [2] 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) Amita Kashyap, G. Sravan Kumar, Sunita Jangir, Emmanuel S. Pilli, Preeti Mishra "IHIDS: Introspection- Based Hybrid Intrusion Detection System in Cloud Environment".
- [3] Watanabe, H., Fujimura, S., Nakadaira, A., Miyazaki, Y., Akutsu, A. & Kishigami, J. J. (2015). Blockchain contract: A complete consensus using blockchain. 2015 IEEE 4th Global Conference on Consumer Electronics (GCCEP).
- [4] Matousek, K. (2008). Security and reliability considerations for distributed healthcare systems. 2008 42nd Annual IEEE International Carnahan Conference on Security Technology.
- [5] Shen, J., Zhou, T., He, D., Zhang, Y., Sun, X., & Xiang, Y. (2018). Block Design-based Key Agreement for Group Data Sharing in Cloud Computing. IEEE Transactions on Dependable and Secure Computing.
- [6] Zhe, D., Qinghong, W ., Naizheng, S., & Yuhan Z. (2017). Study on Data Security Policy Based on Cloud Storage. 2017 IEEE 3rd International Conference on Big Data Security on Cloud (Big Data Security), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS).
- [7] Bharadwaj, D. R., Bhattacharya, A., & Chakkaravarthy, M. (2018). Cloud Threat Defense- A Threat Protection and Security Compliance Solution. 2018 IEEE International Conference on Cloud Computing in Emerging Markets.
- [8] Suma, V. (2019). SECURITY AND PRIVACY MECHANISM USING BLOCKCHAIN. Journal of Ubiquitous Computing and Communication Technologies (UCCT), 1(01), 45-54
- [9] Praveena, A., and S. Smys. "Ensuring data security in cloud based social networks." In 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), vol. 2, pp. 289-295. IEEE, 2017.

- [10] Homoliak, I., Venugopalan, S., Hum, Q., & Szalachowski, P. (2019). A Security Reference Architecture for Blockchains. 2019 IEEE International Conference on Blockchain (Blockchain).
- [11] Killer, C., Rodrigues, B., & Stiller, B. (2019). Security Management and Visualization in a Blockchain-based Collaborative Defense. 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC).
- [12] Hendre, A., & Joshi, K. P. (2015). A Semantic Approach to Cloud Security and Compliance. 2015 IEEE 8th International Conference on Cloud Computing.
- [13] Koo, J., Kim, Y.-G., & Lee, S.-H. (2019). Security Requirements for Cloud-based C4I Security Architecture. 2019 International Conference on Platform Technology and Service (PlatCon).