VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018, KARNATAKA.

A PROJECT REPORT

On

"Blockchain-Driven Police Complaint System with Access Control Technique"

Submitted in Partial Fulfillment for the Award of the Degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By

1EE21CS019 HARSHITHA S

Under the Guidance of

Prof. Gnanamani H
Assistant Professor

Department of Computer Science and Engineering
EAST WEST COLLEGE OF ENGINEERING
(Affiliated to Visvesvaraya Technological University, Belagavi& Approved by AICTE, New Delhi.)
(CA Site, 13, Sector A, Yelahanka New Town, Bengaluru, Karnataka 560064)
Bengaluru – 560064.

2024-2025

EAST WEST COLLEGE OF ENGINEERING

(Affiliated to Visvesvaraya Technological University, Belagavi& Approved by AICTE, New Delhi.)

(CA Site, 13, Sector A, Yelahanka New Town, Bengaluru, Karnataka 560064)

Bengaluru-560064

Department of Computer Science and Engineering

Certificate

Certified that the Project Work entitled **"BLOCKCHAIN-DRIVEN POLICE COMPLAINT SYSTEM WITH ACCESS CONTROL TECHNIQUE"** carried out by **HARSHITHA S (1EE21CS019),** Bonafide students of **East West College of Engineering**, in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University**, **Belagavi** during the academic year 2024-2025. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the Department Library. The project report has been approved as it satisfies the academic requirements in respect of **Project Work (21CSP75)** prescribed for the said degree.

Signature of the Guide	**Signature of the HOD**	**Signature of the Principal**
Prof. Gnanamani H	**Dr. Lavanya N L**	**Dr. Santhosh Kumar G**
Assistant Professor	**Professor & Head**	**Principal**

EXTERNAL EXAMINERS:

Name of the Examiners	**Signature with Date**
1._____	_____
2._____	_____

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement ground my efforts with success.

I consider it is a privilege to express my gratitude and respect to all those who guided me in completion of project work.

I am, grateful to thank our Principal **Dr. Santhosh Kumar G**, East West College of Engineering, who patronized throughout our career & for the facilities provided to carry out this work successfully.

It's a great privilege to place on record my deep sense of gratitude to our beloved HOD **Dr. Lavanya N L** of Computer Science & Engineering, who patronized throughout our career & for the facilities provided to carry out this work successfully.

I am also grateful to thank my project guide **Prof**. **Gnanamani H,** Assistant Professor of CSE department for her invaluable support and guidance.

I would also like to thank the teaching and non-teaching staff members who have helped me directly or indirectly during the project.

Lastly but most importantly I would like thank my family and friends for their co-operation and motivation to complete this project successfully.

HARSHITHA S (1EE21CS019)

ABSTRACT

With the increase in criminal activities and prevalence of unreported incidents across India, the traditional methods of lodging FIRs remain intact despite the availability of online portals. Furthermore, the centralized nature of the Crime and Criminal Tracking Network and System (CCTNS) limits its effectiveness to specific states. To overcome these challenges, a blockchain system is essential to ensure safe and transparent handling of complaints. Our proposed solution uses blockchain technology to manage cognizable and non-cognizable crimes, encrypt FIRs and securely store them by adding hashes to the blockchain network. A blockchain-based police complaint management system enhances accountability, transparency and efficiency, ultimately promoting trust and credibility in law enforcement. Employing a public cloud network ensures transparency while protecting data privacy through encryption. Users access complaint details securely, with only authorized police officers having authorized access. This solution increases the consistency of complaint handling, promoting a more responsive and reliable police department. By leveraging blockchain, our system aims to enhance the integrity and accountability of complaint management, promoting a safer and more transparent environment for citizens and law enforcement.

EAST WEST COLLEGE OF ENGINEERING

(Affiliated to Visvesvaraya Technological University, Belagavi& Approved by AICTE, New Delhi.)

(CA Site, 13, Sector A, Yelahanka New Town, Bengaluru, Karnataka 560064)

Bengaluru-560064

Department of Computer Science and Engineering

DECLARATION

I **HARSHITHA S (1EE21CS019),** Bonafide students of **East West College of Engineering,** hereby declare that the project entitled "**BLOCKCHAIN DRIVEN POLICE COMPLAINT SYSTEM WITH ACCESS CONTROL TECHNIQUE**" submitted in partial fulfilment for the award of Bachelor of Engineering in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2024-2025 is our original work and the project has not formed the basis for the award of any other degree, fellowship or any other similar titles.

Name & Signature of the Student with date

HARSHITHA S (1EE21CS019)

ABBREVIATIONS

- BC : Blockchain
- P2P : Peer-to-Peer
- PoW : Proof of Work
- PoS : Proof of Stake
- PBFT : Practical Byzantine Fault Tolerance
- DApp : Decentralized Application
- DLT : Distributed Ledger Technology
- RSA : Rivest-Shamir-Adleman (Encryption Algorithm)
- ECC : Elliptic Curve Cryptography
- API : Application Programming Interface
- JSON : JavaScript Object Notation
- SQL : Structured Query Language
- DBMS : Database Management System
- UI/UX : User Interface / User Experience
- PKI : Public Key Infrastructure
- KYC : Know Your Customer
- IoT : Internet of Things
- ID : Identification
- HTTP : HyperText Transfer Protocol
- HTTPS: HyperText Transfer Protocol Secure
- IP : Internet Protocol
- TCP/IP: Transmission Control Protocol / Internet Protocol
- FIR : First Information Report
- RTI : Right to Information Act

TABLE OF CONTENTS

Sl No	CHAPTERS	Page no

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

In India, complaints regarding offenses have to be registered under the law. There are two types of offenses i.e. cognizable and non-cognizable offenses. **Cognizable offenses** include serious types of crimes like murder, theft, kidnapping, and rape, etc. According to the Criminal Procedure Code 1973, mentioned in section 2C in case of a cognizable offense, police can arrest the suspect without any warrant. The assigned inspector can start the investigation process without any orders from the court. In the commission of any cognizable offense, the First information report aka FIR. Is registered at the police station. Any individual can file an FIR. If he/she is a victim or has seen the offense being committed. FIR. Details include the complainant's name and address, date and time of location and facts of the incident, etc. Once the FIR is registered, chargesheet report is filed by the police officer. The complainant can apply for acquiring the chargesheet by submitting a letter under the Right to Information Act (RTI) and by paying a certain amount of fees to the court.

Non-cognizable offenses include criminal activities like cheating or forgery etc. A non-cognizable complaint can be registered at the police station. It has a structure similar to FIR. As defined in Section 2(l) of Criminal Procedure court 1973, in case of non-cognizable offense, a police officer has no authority to arrest/investigate without a warrant. The police officer has to obtain permission from the court/magistrate in order to start the investigation process. The crime rate i.e. crime per lakh is increasing at a rapid rate. More than 50 lakh cognizable crimes were registered in the year 2018. Due to increased criminal activities as well as the presence of corrupt police officials, they tend to refuse, avoid or detain the registration of FIR/NCR/complaints which are the obstacles for the complainants to seek justice at the very beginning. According to a survey, 24% of people were unable to register their complaints and 9% of people said that the non-registration was because they were demanded to pay a bribe. Among the people who were able to register their complaints, 30% of the complainants didn't receive an FIR copy. There is a need for a transparent system to eradicate corruption from the public systems. We aim to propose an online police complaint management system using blockchain technology for managing FIR's and NCR's in a decentralized manner in order to cater to problems involving denial of police officers to file complaints.

1.1 Overview of the Present Work

The increasing prevalence of criminal activities and unreported incidents across India underscores the pressing need for a robust and efficient complaint management system. Despite advancements in technology, traditional methods of filing First Information Reports (FIRs) remain largely unchanged, and the existing online portals have not fully addressed the challenges of accessibility and transparency. Furthermore, the centralized

architecture of the Crime and Criminal Tracking Network and System (CCTNS) restricts its effectiveness to certain states, creating significant disparities in law enforcement processes. To bridge these gaps, a blockchain based complaint management system is proposed, offering a secure, decentralized, and transparent framework for handling both cognizable and non-cognizable crimes. This system leverages blockchain technology to encrypt FIRs and securely store them by adding cryptographic hashes to the blockchain network, ensuring data immutability and preventing unauthorized modifications.

The proposed solution enhances accountability and transparency by providing an immutable audit trail of all complaint-related activities. It employs a public cloud network to maintain transparency while ensuring data privacy through advanced encryption mechanisms. Citizens can securely access the details of their complaints, and access to sensitive information is strictly restricted to authorized police officers, promoting trust and minimizing opportunities for corruption. By automating key processes and enabling real-time data validation, the system improves the consistency and reliability of complaint handling across jurisdictions, addressing the inefficiencies and delays associated with manual and centralized systems.

Moreover, the blockchain system fosters greater public confidence in law enforcement by ensuring that complaints are handled with integrity and without bias. The decentralized nature of the platform eliminates single points of failure, making it resilient against data breaches and manipulation. This approach not only streamlines the process of lodging and managing complaints but also equips law enforcement agencies with tools to deliver faster and more reliable services. By integrating blockchain technology into the core of police complaint management, the system aims to create a safer, more transparent, and responsive environment for both citizens and law enforcement, ultimately strengthening the justice system and promoting accountability.

1.2 Problem Statement

The criminal activities in India are increasing at an alarming rate, and it is becoming increasingly difficult for the police to manage and track complaints related to these crimes. Despite the introduction of an online portal for filing First Information Reports (FIRs) and Non-Cognizable Reports (NCRs), most complaints are still filed in a traditional handwritten format, leading to errors and delays in processing. Moreover, the current centralized system, Crime and Criminal Tracking Network and Systems (CCTNS), is limited to a specific state, making it difficult to track and manage complaints across the country.

1.3 Objectives

Ensure Data Integrity:

To create a tamper-proof system that guarantees the immutability of police complaint records.

Blockchain technology ensures that once data is stored, it cannot be altered or deleted, preserving the original state of complaints and evidence.

Enhance Transparency and Accountability:
To provide a transparent audit trail for all activities related to complaint handling and evidence management.
The system ensures that all actions are traceable, promoting accountability within law enforcement agencies.

Improve Data Security:
To safeguard sensitive complaint and evidence data from unauthorized access and potential breaches.
The integration of cryptographic techniques, such as SHA-256 hashing and encryption protocols, protects data at rest and in transit.

Automate Processes for Efficiency:
To reduce manual intervention in complaint management through smart contracts.
Automation speeds up processes such as complaint verification, evidence validation, and data transfers, ensuring timely resolution of cases.

Facilitate Decentralized Data Sharing:
To enable secure and efficient sharing of complaint data across police stations and jurisdictions.
The use of IPFS and blockchain ensures seamless data transfer without central authority dependency.

Build Public Trust:
To instill confidence in the legal system by ensuring fairness, transparency, and security in the complaint management process.
A robust system reduces opportunities for corruption and misconduct, encouraging citizens to report crimes without fear.

Develop a User-Friendly System:
To design an accessible platform for users, including the public and law enforcement, regardless of technical expertise. The system prioritizes usability through intuitive interfaces for filing complaints, tracking progress, and interacting with authorities.

Each of these objectives is designed to address the challenges and gaps in existing police complaint management systems while leveraging the advantages of blockchain technology. By achieving these objectives, the proposed work aims to revolutionize law enforcement data management, ensuring a fair and efficient process for all stakeholders.

1.4 Scope of the Project

The scope of this project encompasses the development, implementation, and evaluation of a blockchain-based police complaint management system. Key elements of the scope include:

System Development:

Designing and developing a decentralized platform using blockchain technology.

Incorporating smart contracts to automate key processes such as complaint registration, verification, and transfer.

Integrating IPFS for secure storage of complaint-related documents and evidence.

User Accessibility:

Creating a user-friendly interface for filing complaints, tracking progress, and accessing case updates.

Ensuring accessibility for both citizens and law enforcement personnel through web-based and mobile applications.

Security Measures:

Employing cryptographic techniques to protect data integrity and confidentiality.

Implementing secure data transfer protocols to prevent unauthorized access during data sharing.

Scalability:

Designing the system to handle large volumes of complaints and data across multiple police stations and jurisdictions.

Exploring lightweight blockchain protocols to improve performance in resource-constrained environments.

Transparency and Accountability:

Providing transparent audit trails for all complaint-related activities.

Enabling real-time updates and status tracking for complaints to build public trust in the system.

Pilot Implementation:

Conducting a pilot program to evaluate the system's performance under real-world conditions.

Gathering feedback from users and stakeholders to refine the system for large-scale deployment.

Chapter 2

LITERATURE SURVEY

Ensuring the Integrity of the Police Complaint files using Blockchain Technology

The paper "Ensuring the Integrity of the Police Complaint Files Using Blockchain Technology" proposes a blockchain-based solution to enhance transparency, security, and accountability in police complaint management. It utilizes the Go Ethereum blockchain platform with a Proof-of-Authority consensus mechanism, smart contracts, and IPFS for decentralized, immutable data storage. The system automates processes, ensures data integrity, and restricts access to authorized personnel, addressing vulnerabilities in traditional systems. While promising, it requires real-world testing, scalability analysis, and integration with existing infrastructure to ensure practical implementation.

Drawbacks: The system lacks real-world testing and scalability analysis, limiting its practical applicability. It does not address vulnerabilities in smart contracts or provide a cost-benefit analysis for implementation. Integration with existing police infrastructure and training strategies for stakeholders are also not considered

Research Gaps The system lacks real-world testing and scalability analysis, limiting its practical applicability. It does not address vulnerabilities in smart contracts or provide a cost-benefit analysis for implementation. Integration with existing police infrastructure and training strategies for stakeholders are also not considered.

Police Complaint Management System Using Blockchain Technology

This paper presents a decentralized system to manage police complaints using blockchain and IPFS technologies. It emphasizes data integrity through tamper-proof ledgers and secure document encryption, employing the Diffie-Hellman key swap method. The system aims to enhance trust, transparency, and efficiency in managing sensitive police records by decentralizing storage and ensuring immutable complaint logs.

Drawbacks: The paper lacks an evaluation of scalability and performance under high data loads. It does not address integration with existing police IT systems, which may pose challenges for adoption.

Research Gaps: The study does not explore potential legal or ethical implications, such as compliance with data privacy regulations like GDPR or ensuring usability for non-technical stakeholders.

Smart FIR: Securing e-FIR Data through Blockchain within Smart Cities

This study focuses on maintaining the authenticity and integrity of electronic First Information Reports (eFIRs) in smart city environments. By using a distributed ledger and identity verification, it mitigates risks such as fraudulent FIR registrations. The proposed system offers real-time data access for law enforcement and addresses limitations in centralized systems, including single points of failure.

Drawbacks: The paper does not analyze the costs and infrastructure requirements for implementing blockchain in resource-constrained police stations. It also lacks discussions on the potential resistance to technology adoption among police personnel

Research Gaps: Privacy-preserving technologies, such as zero-knowledge proofs, are not considered, which could enhance security in sensitive complaint data handling. The paper also does not address how blockchain scalability can be achieved for large smart city networks.

A Platform for Submitting and Handling Crimes Online

This paper highlights an online crime reporting system designed to bridge the gap between the public and law enforcement. It focuses on secure and efficient reporting, allowing users to remain anonymous while ensuring that complaints are recorded transparently. The system promises to enhance responsiveness and accountability in complaint handling through web-based platforms.

Drawbacks: The lack of blockchain or decentralized storage solutions makes the proposed system vulnerable to data tampering and unauthorized access. Additionally, the system's scalability and performance under realworld conditions are not addressed.

Research Gaps: The study does not investigate the integration of advanced security mechanisms, such as blockchain or cryptographic techniques, which could significantly improve data integrity. It also overlooks the challenges of adoption in less technologically advanced regions.

Security Enhancement of Forensic Evidence Using Blockchain

This paper explores how blockchain technology can ensure the immutability of forensic evidence, providing cryptographic hashes and transparent audit trails. It uses smart contracts to automate data verification and transfer, reducing human intervention and ensuring accountability in forensic evidence management.

Drawbacks: The paper does not address the computational overhead of blockchain implementation, which may affect the processing time for large forensic datasets. Furthermore, it lacks detailed real-world case studies to validate the proposed system.

Research Gaps: The study does not explore methods to enhance the scalability of blockchain for large-scale forensic investigations.

Using Blockchain Technology to Manage Police Complaints

This paper discusses a blockchain-based solution for managing police complaints, using IPFS for decentralized storage and a security module for encrypting documents. It highlights the benefits of transparency, data integrity, and public trust through immutable complaint records and consensus mechanisms.

Drawbacks: The paper lacks concrete implementation details and real-world performance evaluations. Additionally, it does not address the interoperability of blockchain systems with existing police workflows.

Research Gaps: The study does not investigate the potential challenges of legal compliance or the ethical implications of using immutable records. It also overlooks user experience and accessibility considerations, which are crucial for wide-scale adoption.

2.1 Proposed Work:

The proposed system aims to leverage blockchain technology to enhance the integrity, transparency, and security of police complaint records. By utilizing a decentralized blockchain network and smart contracts, the system will provide an immutable ledger for storing police complaints and related evidence. The following components are central to the proposed work:

Decentralized Ledger: Implementing a blockchain network to store complaint records, ensuring that all data is immutable and tamper-proof.

Smart Contracts: Using Ethereum-based smart contracts to automate complaint submission, verification, and evidence handling processes. These contracts will also manage access control and ensure transparency.

InterPlanetary File System (IPFS): Integrating IPFS for secure, decentralized storage of supporting documents and evidence. Each document will be associated with a unique hash for verification.

Cryptographic Security: Employing SHA-256 hashing algorithms to ensure the integrity of complaint records and supporting files.

User-Friendly Interface: Developing a web-based and mobile-friendly platform for users to file complaints, track complaint status, and interact with law enforcement transparently.

Secure Data Transfers: Implementing encrypted data transmission protocols for transferring complaints between police stations or jurisdictions.

Chapter 3

SYSTEM REQUIREMENTS

3.1 Functional Requirements

- The system will be developed as a web-based application utilizing advanced Java technology, with the Tomcat web server and MySQL database server as its foundation.
- The system will incorporate three primary actors such as Admin, User, and Police.
- The Admin also referred to as the Trusted Authority, will implement user privileges, responsible for creating cryptography keys and also managing user and police attributes.
- The Admin will have the ability to create and maintain police details within the system.
- User Registration and Authentication: The Admin will be empowered to create user accounts within the system, employing robust authentication mechanisms to ensure the security and privacy of user accounts.
- Users will have the capability to submit complaints through the system's interface.
- Stringent measures will be implemented within the system to safeguard the privacy and security of user data and complaint information.
- Access controls, encryption, and secure communication protocols will be employed to provide robust protection for sensitive information.

3.2 Non-Functional Requirements

Performance:

The system must handle a high volume of complaint submissions and evidence uploads efficiently.

Response times for data retrieval and updates must not exceed 2 seconds under normal conditions.

Scalability:

The platform must support increasing user loads, including millions of complaints and evidence records.

The blockchain infrastructure must allow seamless addition of nodes to expand the network capacity.

Reliability:

The system must ensure 99.9% uptime to avoid interruptions in complaint submission and processing.

Backup and recovery mechanisms must restore operations within 1 hour in case of failures.

Usability:

Interfaces must be intuitive and user-friendly, requiring minimal training for users.

The platform must provide clear error messages and help options for troubleshooting.

Security:

Data must be encrypted during storage and transmission.

The system must enforce strict role-based access control and multi-factor authentication.

Compatibility:

The system must integrate with existing police IT infrastructures and databases.

It must support cross-platform functionality on web browsers and mobile devices.

Maintainability:

The codebase must be modular and well-documented to facilitate easy updates and troubleshooting.

Regular updates and patches must be applied without disrupting operations.

Legal Compliance:

The system must adhere to data protection laws, such as GDPR and applicable regional regulations.

3.3 Hardware and Software Used

3.3.1 Hardware Requirements:

- System : i3 core processor or above
- Hard Disk : 500 GB
- RAM : 8 GB
- Any desktop / Laptop system with above configuration or higher level

3.3.2 Software Requirements:

- Operating system : Windows 8 or above
- Coding Language : Java (Jdk 1.7)
- Web Technology : Servlet, JSP
- Web Server : TomCAT 6.0
- IDE : Eclipse Indigo
- Database : My-SQL 5.0
- UGI for DB : SQLyog
- JDBC Connection : Type 4 Driver

Chapter 4

SYSTEM DESIGN

4.1 Architecture

The blockchain-based police complaint management system is built on a robust and layered architecture designed to ensure efficiency, security, and scalability. The **High-Level Architecture** consists of three main layers. The **Presentation Layer** provides user interfaces accessible via web and mobile platforms, enabling citizens to file complaints, track their progress, and submit evidence seamlessly. This layer also includes dashboards tailored for law enforcement officers, allowing them to efficiently manage complaints and access relevant data. The **Application Layer** houses the business logic essential for complaint processing, incorporating smart contracts that automate core operations. This layer also includes mechanisms for authentication and role-based access control to ensure that only authorized personnel have access to sensitive information. The **Data Layer** is the backbone of secure and immutable record-keeping, featuring a blockchain ledger for logging all transactions, IPFS (InterPlanetary File System) for decentralized storage of complaint documents and evidence, and secure databases for storing metadata to allow quick access without compromising security.

The **Security Design** prioritizes the protection of sensitive data through multiple measures. All data, both in transit and at rest, is safeguarded using end-to-end encryption. A robust role-based access control system ensures that users can only access information relevant to their roles, minimizing risks associated with data breaches or unauthorized access. Additionally, cryptographic hashing mechanisms are employed to ensure tamper detection, maintaining the integrity and authenticity of all recorded data.

To handle varying loads and ensure consistent performance, the system incorporates a thoughtful **Scalability Design**. Its modular architecture allows for the seamless addition of blockchain nodes, enhancing scalability as the system grows. Lightweight protocols are utilized to optimize performance, particularly in resource-constrained environments, ensuring that the system remains efficient and responsive.

Finally, the **Integration Design** facilitates compatibility with existing police IT infrastructures. APIs enable the system to integrate seamlessly with pre-existing platforms, enhancing its adaptability and usability. Moreover, the design ensures compatibility with cross-border complaint systems, enabling international collaboration and extending the system’s applicability to global law enforcement scenarios. This comprehensive and well-structured design ensures that the system is both robust and adaptable, capable of addressing the challenges of modern police complaint management.

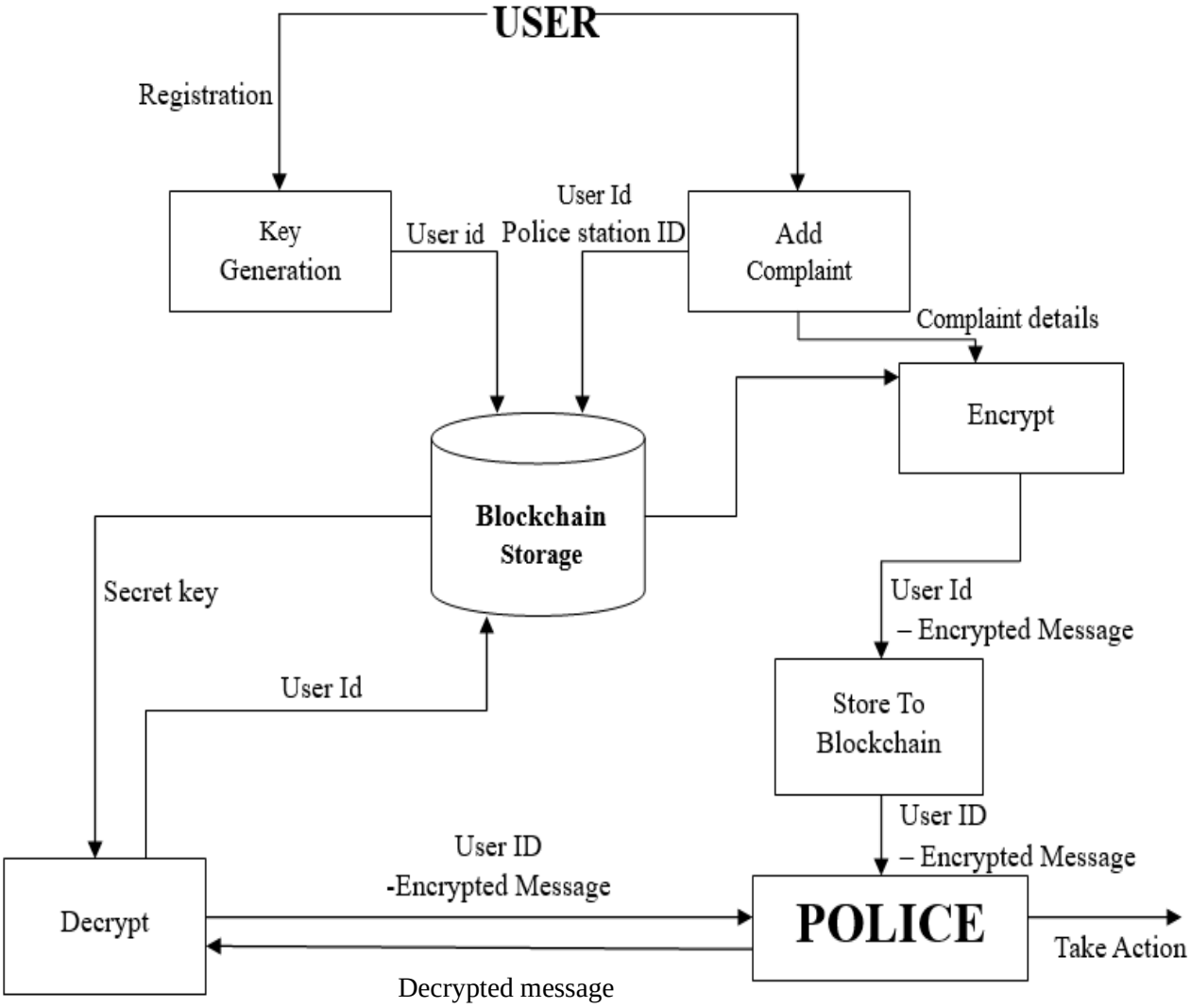

Fig. 4.1 System Architecture

This **System Architecture** illustrates a **Police Complaint Management System** using **Blockchain Technology**.

Police Action:

The police take the necessary actions based on the complaint.

- **User Registration**:

Citizens and officers register using secure authentication methods.

Multi-factor authentication ensures secure access.

- **Complaint Filing**:

Users submit complaints through the interface.

Complaints are hashed and stored on IPFS, with hashes recorded on the blockchain.

- **Verification**:

Law enforcement verifies complaints and updates their status.

All actions are logged immutably on the blockchain.

- **Evidence Submission**:

Users upload evidence, which is stored securely on IPFS.

Blockchain ensures traceability and authenticity of the evidence.

- **Tracking and Updates**:

Users track complaint progress in real time.

Notifications are sent automatically via smart contracts.

4.2 Methodology

4.2.1 Blockchain

Blockchain is a decentralized digital ledger that records transactions across multiple computers or nodes. It serves as the underlying technology for cryptocurrencies like Bitcoin, but its potential applications extend beyond digital currencies. At its core, a blockchain consists of a chain of blocks, where each block contains a list of transactions. These blocks are linked together using cryptographic hashes, creating an immutable and transparent record of all the transactions. Once a block is added to the chain, it becomes extremely difficult to alter or remove the information stored within it, providing a high level of security and integrity. This decentralization eliminates the need for intermediaries, reduces the risk of fraud, and increases trust among participants.

However, blockchain is not without its challenges. Issues such as scalability, energy consumption, and regulatory frameworks need to be addressed for widespread adoption. Nonetheless, the potential benefits offered by blockchain technology continue to drive innovation and exploration in various fields, paving the way for a more decentralized and secure future.

4.2.2 Hashing

Clients may employ cryptosystem to encrypt the files in the cloud to lowering the threat associated with cloud storing. By maintaining a brief hashing in internal space, the use of a hash value is a suitable method for maintaining data fidelity. By re-tallying up the hashes of the information that was obtained and comparing it to locally preserved information, the server answers are authenticated in this way. Here, we are using MD5 hashing function for security purpose when file uploading and downloading is performed.

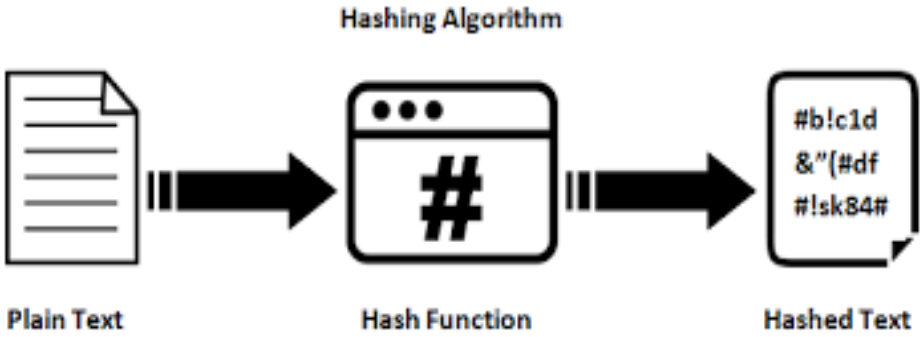

Fig. 4.2 Hashing Algorithm

We face issues with information protection, system files, recoveries, network congestion, and hosting safety in CCT.

The block verification process in blockchain involves using hashing techniques to ensure the integrity and authenticity of data within a block. Here's a brief description of the process:

- **Data Compilation**: A block contains a collection of data, such as transactions or records, that need to be verified. This data is compiled together before the verification process begins.
- **Hashing:** The block's data is then fed through a cryptographic hash function, which generates a unique fixed-length string of characters called a hash. The hash is essentially a digital fingerprint of the block's data.
- **Hash Linking:** Each block also contains the hash of the previous block in the blockchain. This creates a chain-like structure where each block is linked to the one before it, forming a tamper-evident sequence.
- **Verification:** The generated hash is compared to the hash stored in the previous block. If they match, it confirms that the data within the block has not been altered. Any modification to the data would result in a completely different hash, thereby alerting the network to potential tampering.
- **Block Addition**: Once a block is verified and accepted by the network, it is added to the blockchain. The new block then serves as the reference point for verifying the next block in the chain.

By employing hashing techniques and linking blocks through hashes, the block verification process ensures the immutability and security of data in a blockchain network.

4.2.3 Hybrid Cloud

A hybrid cloud, also known as a cloud hybrid (CH), is a CCE which integrates on-premise private with a public CCE, enabling information & apps to be exchanged among them. Building a uniform, automatic, & good handling CCE is the main objective of merging such private and public cloud systems. The private cloud platform is used in a CH to handle crucial tasks, while the public clouds platform is used to handle non-critical operations. Several clouds arrangements, in which a company employs more than a single public cloud in conjunction to its on-premises data centers are included in certain criteria of cloud hybrid systems

as per some individuals. Advantages of CH systems includes the ability for enterprises to immediately ramp capacities upward or downward to manage surplus capacities while computational & handling demands exceeds the limitations of an in-house infrastructure. They can also save both expenditure & time by not having to buy, set up, & manage additional servers which they may not necessarily require.

In CCT, fault-tolerance entails drawing up a plan for continuous operations while certain components are inaccessible or broken. It assists businesses in determining its infrastructural demands & expectations plus offers solutions in the event that a certain instrument is unreachable for whatever occasion. Numerous copies for every operation are executed simultaneously in FTS. As a result, additional copies could be employed to continue the system functioning in its place if any component fails. It's crucial to maintain a recovery mechanism in case a component of the system malfunctions or breakdowns. The server employs multiple contingency backups featuring multiple applications

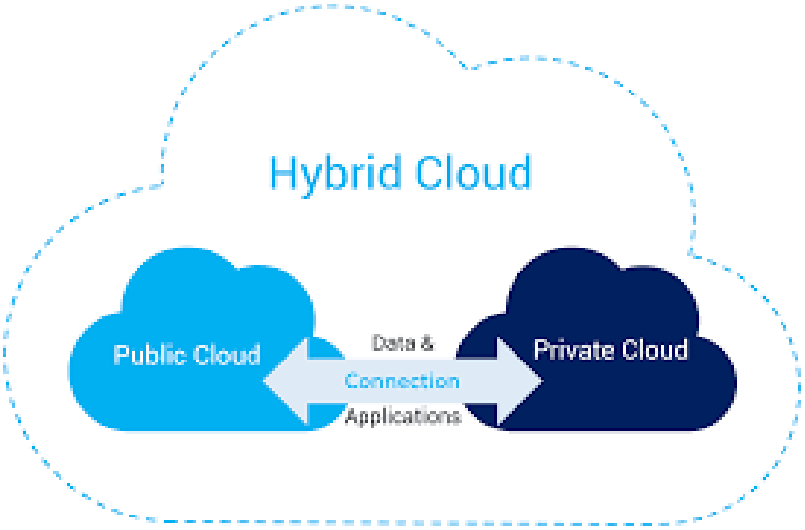

Fig. 4.3 Hybrid Cloud Architecture

The following are some methods for FTS in CCT:

- Every operation must be prioritized when creating a fault-tolerant mechanism. The repository deserves special consideration since it supports several other instances.
- The organization will need to engage on sample testing once the prioritized outlining is done. For instance, organization has a web-site with discussions where people may sign up & leave feedbacks. Customers will be unable to log-in if authenticating facilities stop working as a result of a malfunction.

4.2.4 RSA Algorithm

The RSA algorithm, named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman, is a widely used cryptographic algorithm for secure communication and data encryption. RSA is an asymmetric

encryption algorithm, which means it uses a pair of keys: a public key for encryption and a private key for decryption.

Here's a high-level overview of how the RSA algorithm works:

Key Generation:

Select two large prime numbers, p and q.

Compute their product, n = p * q, which becomes the modulus for both the public and private keys.

Calculate Euler's totient function, φ(n) = (p - 1) * (q - 1).

Choose an integer e (usually a small prime) such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$. This is the public exponent.

Compute the private exponent d, which is the modular multiplicative inverse of e modulo φ(n). In other words, $(e * d) \% \varphi(n) = 1$. This is the private key.

Encryption:

To encrypt a message m, the sender uses the recipient's public key (n, e).

The sender represents the message as an integer between 0 and n-1.

The sender computes the ciphertext $c = m^e \bmod n$.

Decryption:

To decrypt the ciphertext c, the recipient uses their private key (n, d).

The recipient computes the plaintext message $m = c^d \bmod n$.

The security of RSA is based on the difficulty of factoring large composite numbers into their prime factors. Breaking RSA encryption requires factoring the modulus n, which is computationally difficult for sufficiently large prime numbers.

RSA is widely used for secure communication, digital signatures, and key exchange protocols. However, it can be computationally expensive for encrypting large amounts of data, so it is often used in combination with symmetric encryption algorithms, where the data is encrypted with a symmetric key, and then the symmetric key is encrypted using RSA.

FTP Protocol

File Transfer Protocol (FTP) is a widely used network protocol for transferring files between a client and a server over a network. It facilitates the uploading, downloading, and management of files, making it a straightforward choice for systems that require bulk data transfers. In the context of the uploaded document, FTP could be considered as a traditional method for transferring police case files or evidence between

different departments or secure servers. For example, police stations could use FTP to share General Diary (GD) entries, investigation reports, or evidence files with other stations or legal entities. However, FTP has significant security limitations. Standard FTP transmits data in plaintext, making it susceptible to interception and unauthorized access during transmission. This poses a considerable risk for sensitive data, such as police records, where a breach could compromise investigations or endanger victims. To mitigate these vulnerabilities, secure alternatives like SFTP (SSH File Transfer Protocol) or FTPS (FTP Secure) could be used. These protocols introduce encryption to the data transfer process, ensuring that files remain confidential and tamper-proof.

While FTP or its secure versions might still be relevant for certain use cases, the blockchain-based system described in the document emphasizes the use of IPFS (Inter Planetary File System). IPFS eliminates the need for traditional FTP by providing a decentralized, secure, and tamper-proof method for storing and sharing files. Files stored in IPFS are hashed, and any change to the file results in a new hash, making unauthorized modifications easily detectable. This approach aligns with the document's goal of enhancing data security and integrity.

SMTP:

Simple Mail Transfer Protocol (SMTP) is the standard protocol for sending emails across networks. It governs the format, transmission, and delivery of email messages. In the context of the uploaded document, SMTP could be used to send notifications to users and stakeholders regarding updates on police cases. For instance, when a General Diary (GD) is registered, verified, or transferred, SMTP could be utilized to notify users via email, including critical information such as IPFS hashes or blockchain transaction IDs for transparency and traceability.

However, SMTP, in its traditional form, lacks encryption, leaving email messages vulnerable to interception and spoofing. This is a significant concern when transmitting sensitive information, such as complaint details or evidence links. To address these vulnerabilities, the system can implement secure email communication using SMTPS (SMTP Secure) or encryption technologies like PGP (Pretty Good Privacy). These measures ensure that emails are encrypted during transmission, preventing unauthorized access or tampering.

By integrating SMTP securely, the system can maintain effective communication without compromising data confidentiality or integrity. Notifications sent via SMTP can complement the blockchain-based system, providing real-time updates to users while ensuring that sensitive information remains protected. This integration supports the document's overarching objective of transparency and security in managing police case files.

Chapter 5

IMPLEMENTATION

5.1 Use Case Diagrams

A use case is a set of scenarios that describing an interaction between a source and a destination. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors. Shows the use case diagram.

5.1.1 Use Case Diagram Admin

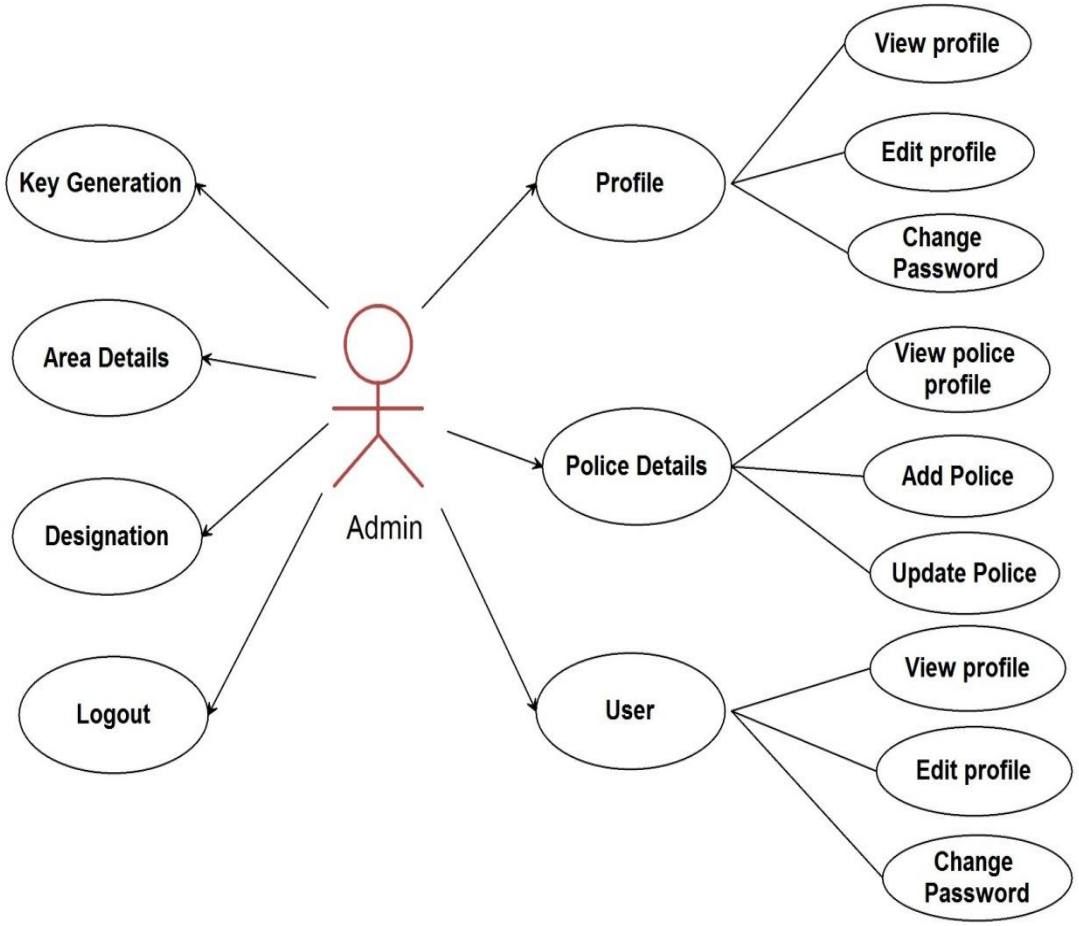

Fig. 5.1 Use Case Diagram for Admin

The diagram illustrates the various functionalities available to an **Admin** in a system. The Admin has control over multiple areas, including **Key Generation**, managing **Area Details**, **Designation**, and the ability to **Logout**. For **Profile Management**, the Admin can view, edit, and change their password. In terms of **Police Details Management**, the Admin can view police profiles, add new police entries, and update existing police details. Additionally, the Admin can manage **User Profiles** by viewing, editing, and allowing

password changes. Overall, the Admin serves as the central authority, handling system configurations, user profiles, and police-related information effectively.

5.1.2 Use Case Diagram User

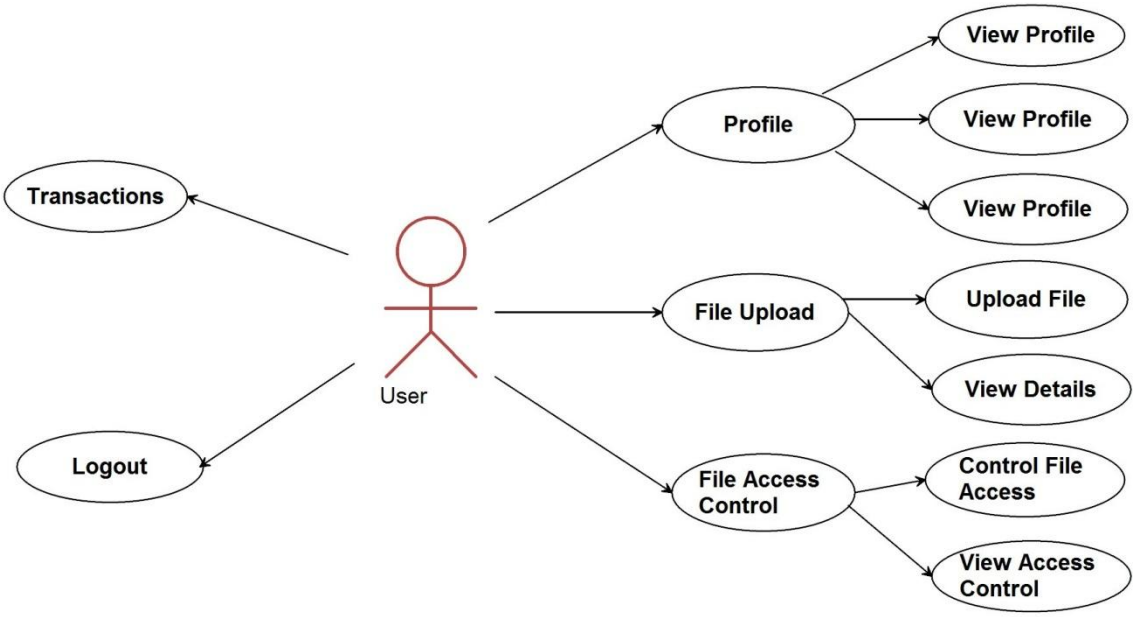

Fig. 5.2 Use Case Diagram for user

The diagram outlines the functionalities available to a User in the system. Users can manage their Profile, which includes multiple options to View Profile. For managing data, users can perform File Upload tasks, including uploading files and viewing file details. Additionally, users have access to File Access Control, allowing them to control and view file access permissions. Other features include managing Transactions and logging out via the Logout option. These functionalities provide users with comprehensive control over their profiles, file management, and access settings.

5.1.3 Use Case Diagram Police

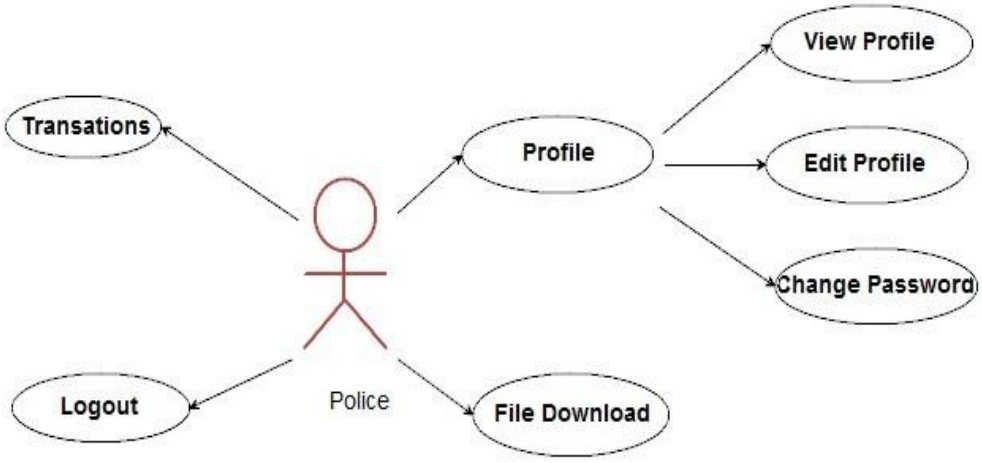

Fig. 5.3 Use Case Diagram for Police

The diagram outlines the functionalities available to a Police in the system. Police can manage their Profile, which includes multiple options to View Profile, Edit Profile, Change Profile. For managing data, users can perform File Download tasks and viewing file details. Other features include managing Transactions and logging out via the Logout option. These functionalities provide Police with comprehensive control over their profiles and file management.

5.2 Data Flow Diagram:

A data flow diagram (DFD) is graphic representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context level DFD first which shows the interaction between the system and outside entities. DFD's show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage. There are only four symbols:

- Squares representing external entities, which are sources and destinations of information entering and leaving the system.
- Rounded rectangles representing processes, in other methodologies, may be called 'Activities', 'Actions', 'Procedures', 'Subsystems' etc. which take data as input, do processing to it, and output it.
- Arrows representing the data flows, which can either, be electronic data or physical items. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly.
- The flat three-sided rectangle is representing data stores should both receive information for storing and provide it for further processing.

5.2.1 Level 0 Data Flow Diagram:

The Level0 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole.

A **Level 0 Data Flow Diagram (DFD)** is a high-level representation of a system's major processes, data flows, external entities, and data stores. It's also called a *context diagram* because it depicts the system as a single process and shows how it interacts with external entities.

Level 0 DFD typically includes:

- **External Entities**: These are sources or destinations of data outside the system (e.g., users, other systems, or organizations).
- **Processes**: At this level, the entire system is represented as a single process (often labeled "System" or "Process 0").
- **Data Flows**: Arrows that indicate the flow of data between external entities and the system.
- **Data Stores**: Optional at this level, but they are sometimes shown to give context.

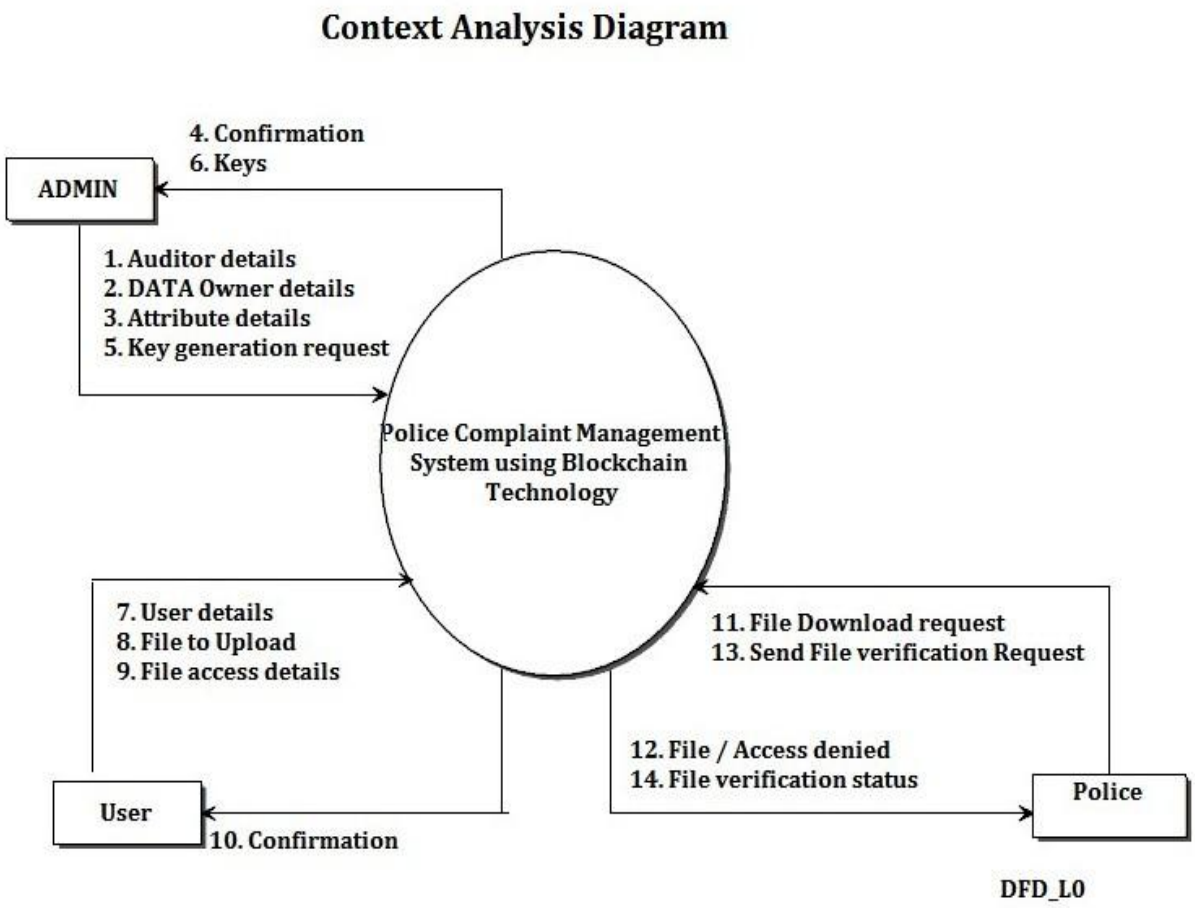

Fig. 5.4 Level 0 Data Flow diagram or Context Analysis Diagram

The process will do by the Intermediate nodes. Finally the requested data will be delivered to the requesting node. It represents the overall process in the simple and short procedure. Here there are only to nodes Source which used transmit the data packet to the respective node and destination which are being used to receive the packet and gain the required data.

5.2.2 Level 1 Data Flow Diagram:

In the level1Data Flow Diagram, select a file and transfer that file to server. Server receives all the details and generates a Message Digest ,once MD file is generated it retrieves all the public key belongs to the user group i.e.(MD + Publickey) generates a secure MD , Encrypt secure MD with user Private keys and generates Ring-Signature and send a mail to all users.

Key Elements of a Level 1 DFD:

- **Sub-Processes**: Processes are numbered and represent specific functional areas or activities within the system.
- **External Entities**: External entities from Level 0 remain unchanged.
- **Data Stores**: Data stores are introduced to show where data is held.
- **Data Flows**: More detailed data flows illustrate the interactions between sub-processes, data stores, and external entities.

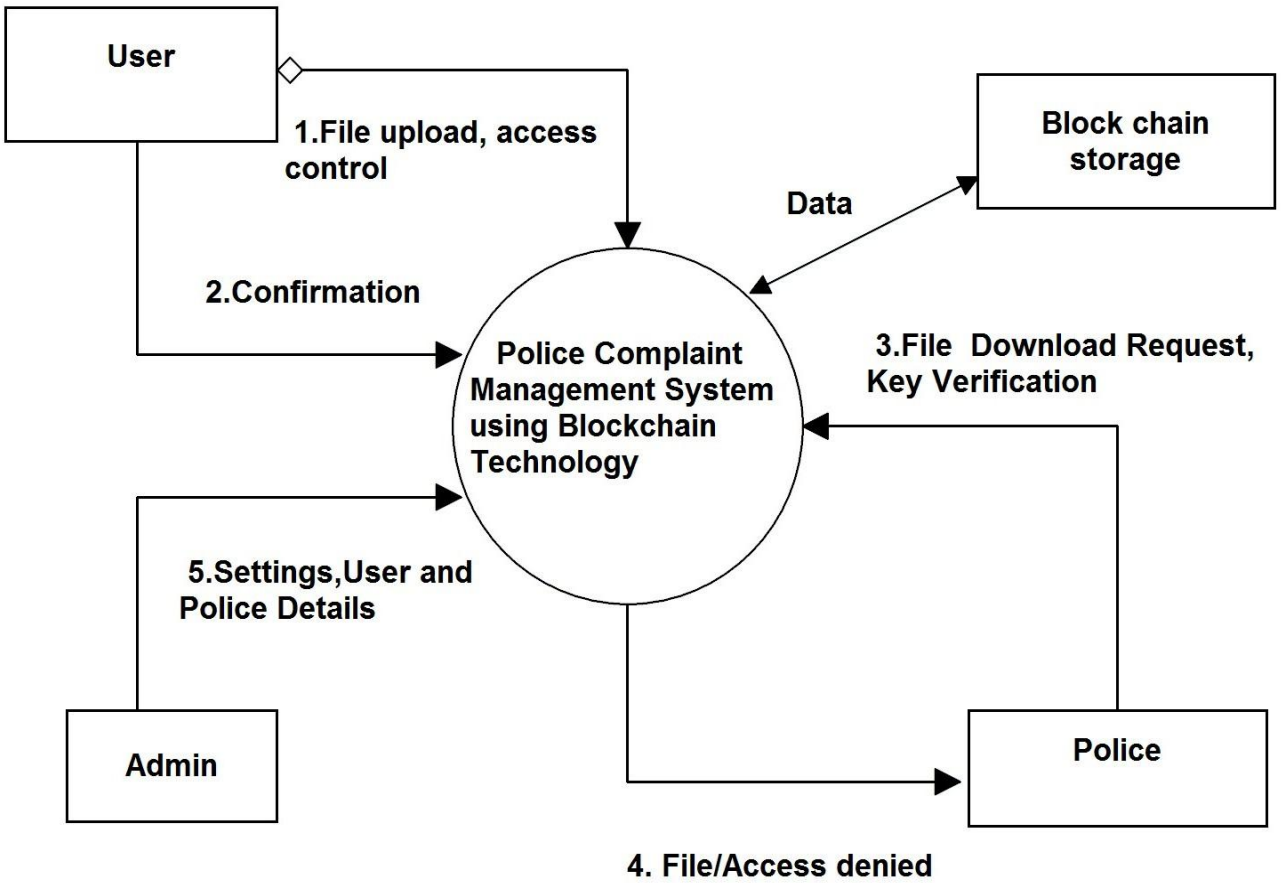

Fig. 5.5 Level 1 Data Flow Diagram

This Data Flow Diagram (DFD) illustrates the working of a Police Complaint Management System using Blockchain Technology.

- **User Interaction**: The User uploads a file and provides access control permissions. The system processes the upload and sends a confirmation back to the user.
- **Blockchain Storage**: The uploaded data is securely stored in the Blockchain Storage, ensuring tamper-proof and transparent data handling.
- **Police Interaction**: The Police can send a File Download Request to the system and perform Key Verification to access the stored data. Based on access permissions, the system can either provide the file or deny access (Step 4).
- **Admin Role**: The Admin manages settings, user details, and police details (Step 5) within the system. This ensures proper control and monitoring of the system.
- **System Central Functionality:** The Police Complaint Management System acts as the central hub that interacts with the User, Police, Admin, and Blockchain Storage, managing data flow, access control, and security.

5.2.3 Level 2 Data Flow Diagram:

A **Level 2 Data Flow Diagram (DFD)** is a more detailed breakdown of a single **Level 1 process**. It decomposes the process into smaller, more specific sub-processes to provide better insight into the system's inner workings and how data flows between components.

This Data Flow Diagram (DFD) explains the File Upload Process.

- **User:** The process begins with the user initiating the file upload.
- **Select File**: The user selects the file they want to upload.
- **Transfer to Web Server**: The selected file is transferred to a web server for further processing.
- **Block Creation Process**: The file is processed, and a block is created (e.g., for blockchain storage or system verification).
- **Store Block**: The created block is stored securely in the system.
- **Update Metadata**: Metadata related to the file and block is updated in the system for tracking purposes.
- **Upload Confirmation**: A confirmation message is sent back to the user, indicating that the file upload process is complete.
- **User:** The process ends, and the user receives confirmation.

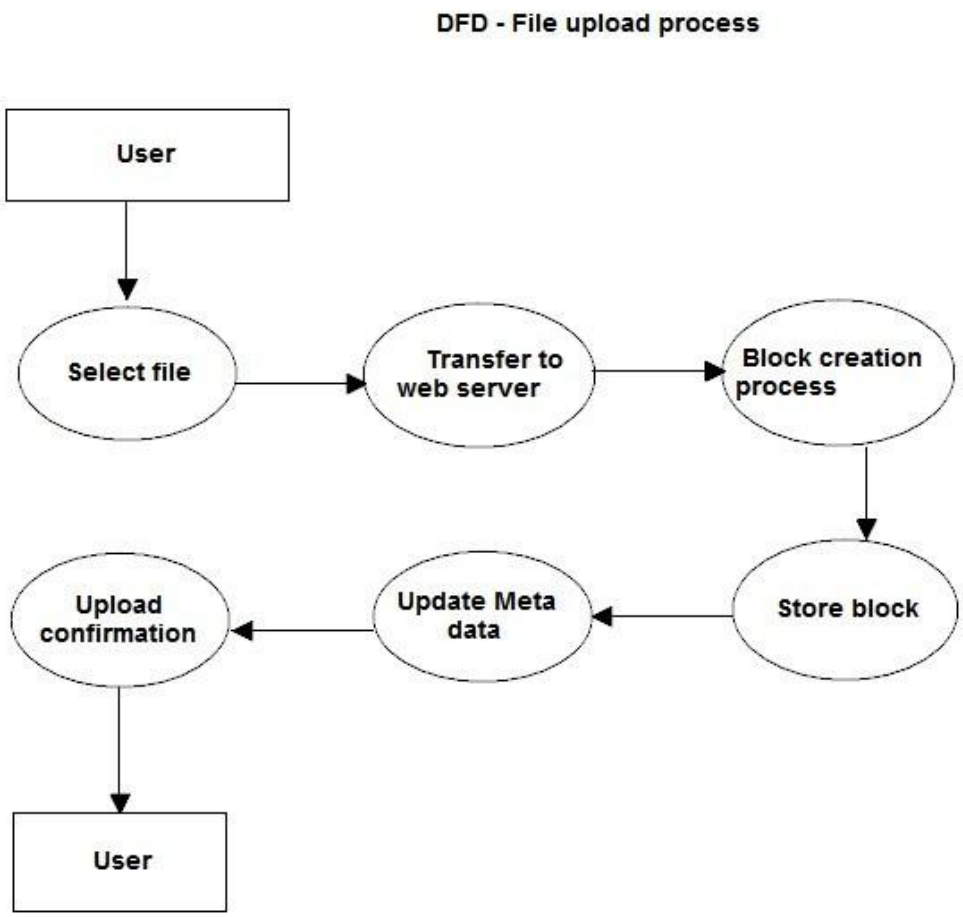

Fig. 5.6 Level 2 Data Flow Diagram

5.2.4 Level 3 Data Flow Diagram:

A **Level 3 Data Flow Diagram (DFD)** is the most detailed layer of a DFD hierarchy. It decomposes a **Level 2 process** into smaller, granular sub-processes, giving an in-depth view of how data flows and is processed within that part of the system. Here's how it is structured:

- **Read Data**: Data is first read as input for block creation.
- **Generate Root Hash (MD5)**: A root hash is generated to represent the input data uniquely.
- **Pick Previous Block Hash Tag**: Metadata from the previous block hash is fetched to maintain chain continuity.
- **Generate Block Header**: A block header is created using the previous hash, root hash, timestamp, and other details.
- **Encrypt Data and Create Block Body**: Data is encrypted to ensure security, forming the block body.
- **Compress and Generate Block**: The block header and body are compressed to finalize the block.
- **Store Block**: The completed block is stored securely.
- **Update Metadata Table**: Metadata information is updated to reflect changes, ensuring proper tracking.

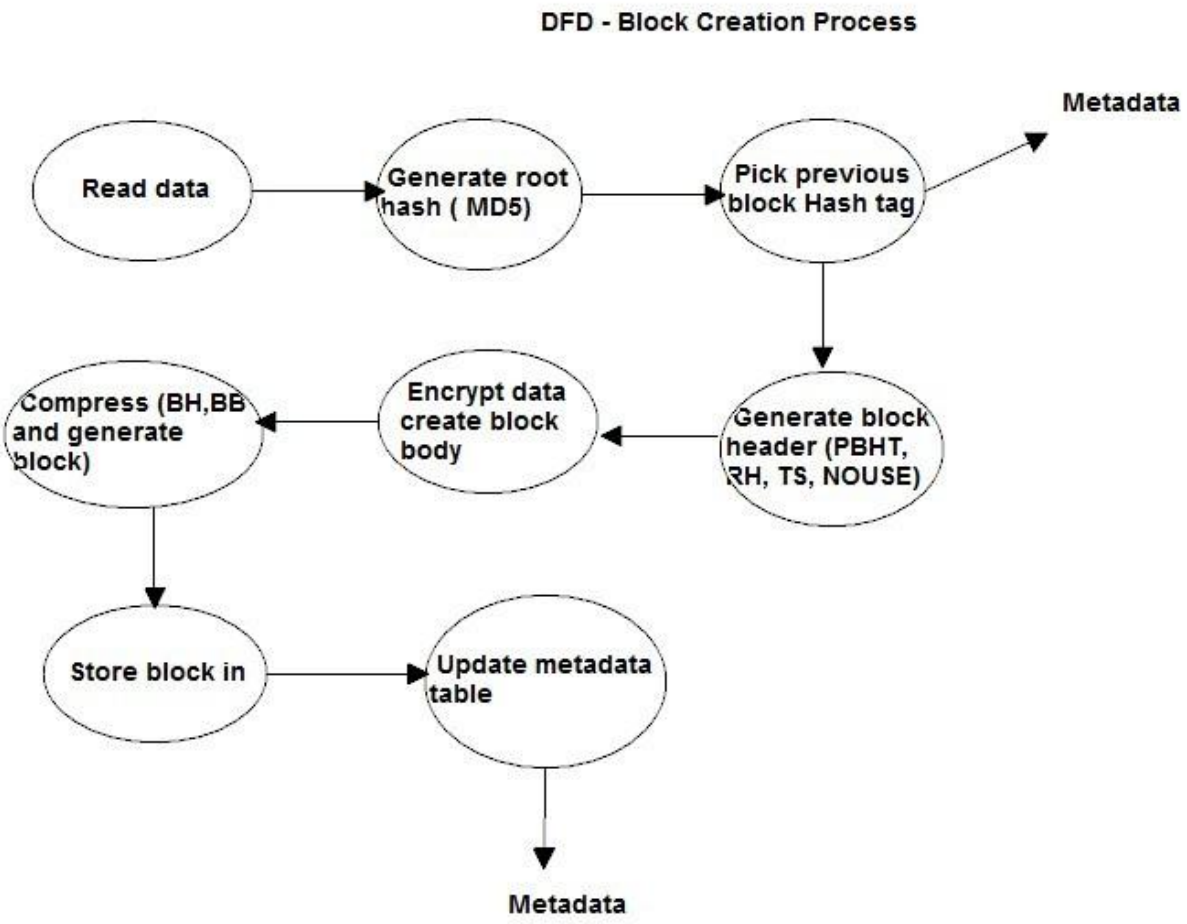

Fig. 5.7 Level 3 Data Flow Diagram

5.2.5 Level 4 Data Flow Diagram:

A **Level 4 Data Flow Diagram (DFD)** is the most detailed layer of a DFD hierarchy. It decomposes a **Level 3 process** into smaller, granular sub-processes, giving an in-depth view of how data flows and is processed within that part of the system. Here's how it is structured:

- **Select file**: File is first selected from the database
- **Input the Private Key :** Private key text document must be given as key for verification.
- **Access Control Verification**: the private key is verified with the access control given by the user.
- **Block ID & Metadata**: A block ID is used to find the Hash value from the meta data.
- **Download Block from Cloud**: decrypted complaint file is downloaded from the cloud.
- **Uncompress the Block**: body are Uncompressed to open the block.
- **Decrypt**: the text file is converted into readable form.
- **Download File in User System** : The converted file is stored to Recived_complaints in D-drive.
- **Display Conformation message**: the downloaded message is popped up.

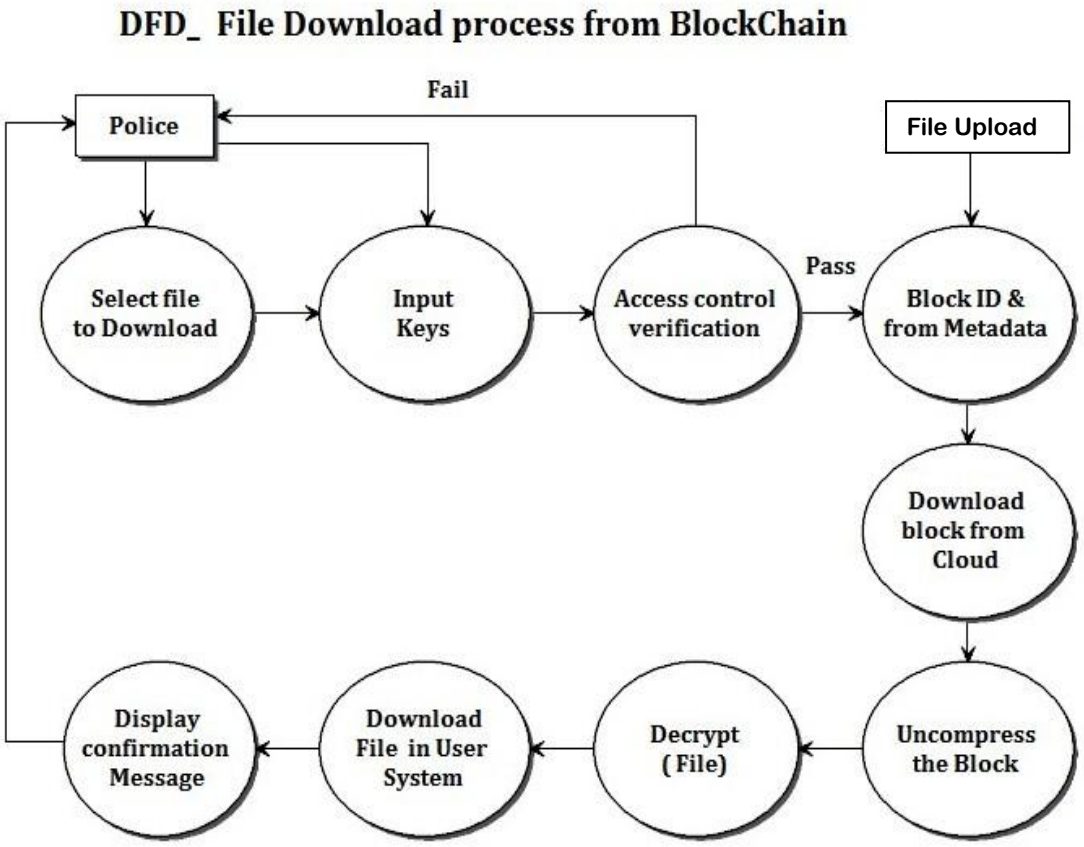

Fig. 5.8 Level 4 Data Flow Diagram

5.3 Sequence Diagram

The sequence diagram illustrates the workflow of the blockchain-based system designed for securely managing and storing Electronic Health Records (HER) files. The process ensures secure handling, transparency, and immutability of data by leveraging blockchain technology. Below is a detailed explanation of the sequence of events and their significance:

Process Flow:

- **File Submission:** the process begins with data owner uploading an HER file via the interface.
- **Data Processing :** the Webserver processes the received file and appends additional information, such as storage details .
- **Block creation :** Hash code Process, block body creation, Block header creation and compression.
- **Storage :** once the block is finalized, it is sent to the storage system. This ensures that the block is securely and permanently stored in the blockchain, where it remains tamper-proof and accessible to authorized users.
- **Acknowledgment:** after successful storage, an ACK signal is sent back to the Webserver, which relays it through the interface to the Data Owner.

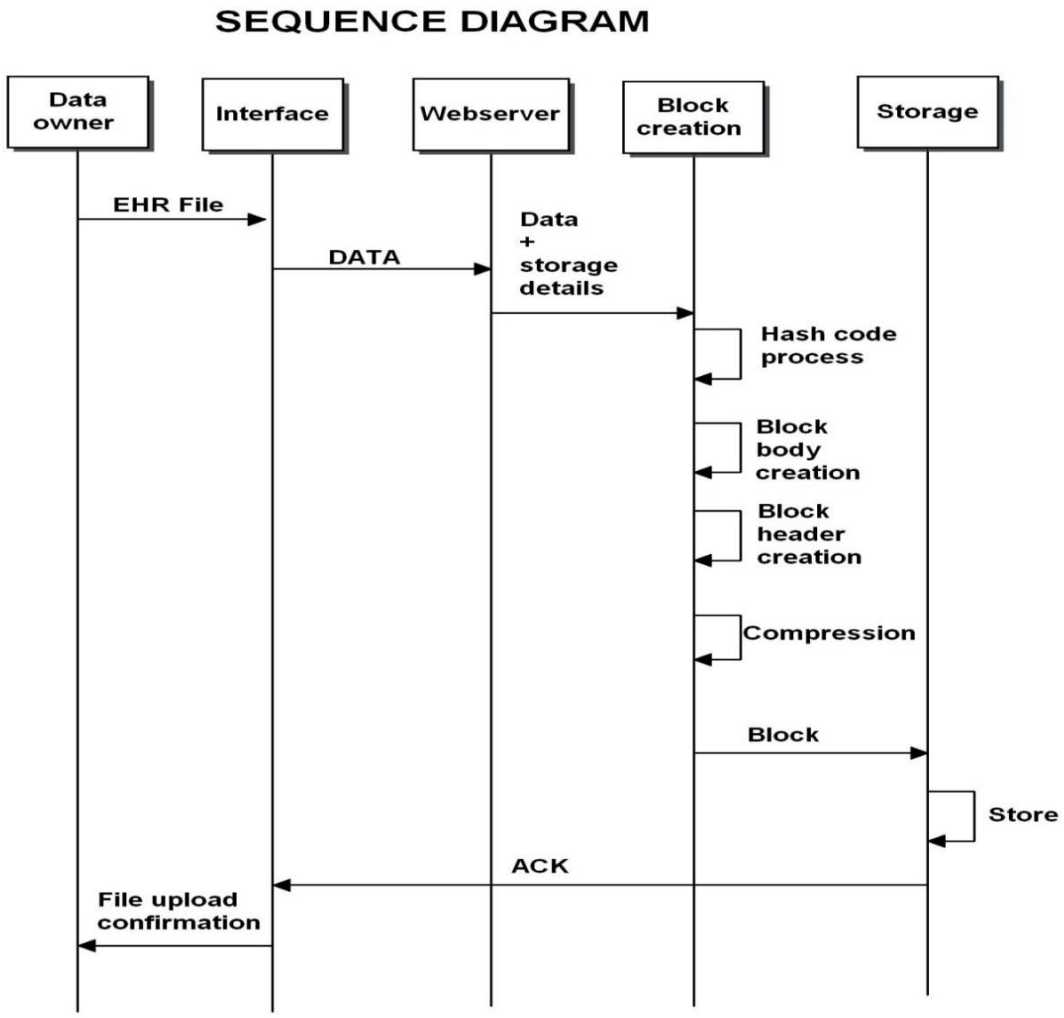

Fig. 5.9 Sequence diagram

5.4 Modules Description

Admin

The admin is the super user and he is also called trusted authority (TA) who will create cryptography keys. Admin has to create user and Police officers. The working of the admin module is given below:

- Login
- Profile
  - View Profile
  - Edit Profile
  - Change Password
- User
  - View User
  - Add User
  - Update User
- Police

  - View Police
  - Add Police
  - Update Police
- Key Generation
- Department (Attribute I)
- Designation (Attribute II)
- Logout

Users

Users who own the complaint, user must have separate login page and he can able to login into home page using authorized password. The working of the User module is given below:

- Login
- Profile
  - View Profile
  - Edit Profile
  - Change Password
- File Upload
  - Upload File
  - View Details
- File Access Control

  Control File Access Control

  View File Access Control
- Transaction
- Logout

Police Officer

Police officer can able to login into their home page with authorized password. Police can able to download files from Block Chain Server.

- Login
- Profile
  - View Profile
  - Edit Profile
  - Change Password
- File Download
- Transaction

- Logout

The implementation phase involves more than just writing code. Code also needs to be tested and debugged as well as compiled and built into a complete executable product. We usually need to utilize configuration management in order to keep track of different version of code. This is the stage of the project where the theoretical design is turned into a working system. If the implementation is not carefully planned and controlled, it can cause chaos and confusions. It is always a good idea to keep in mind that some characteristics that should be found in a good implementation like Readability- our code is written in MVC Architecture, JAVA to achieve the objective of the project that is to introduce a novel scheme of mechanism design for balancing the resource consumption.

Our implementation stage requires the following tasks:

- Careful planning
- Investigation of system and constraints
- Design of methods to achieve the changeover
- Evaluation of the changeover method
- Correct decisions regarding selection of the platform
- Appropriate selection of the language for application development
- Java Technology is both a programming language and a platform.

5.5 JAVA

Java is a computer-programming language that is concurrent ,class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most programming languages particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

5.6 J2EE

The Java EE stands for Java Enterprise Edition, which was earlier known as J2EE and is currently known as Jakarta EE. It is a set of specifications wrapping around Java SE (Standard Edition). The Java EE provides a platform for developers with enterprise features such as distributed computing and web services. Java EE

applications are usually run on reference run times such as micro servers or application servers. Examples of some contexts where Java EE is used are e-commerce, accounting, banking information systems. The J2EE technologies consist of Servlets, Java Server Pages (JSP), Java Database Connectivity (JDBC), Enterprise Java Bean (EJB), Java Message Service (JMS) etc. In our project, we use the MVC architecture containing Servlets, JSP and JDBC technologies.

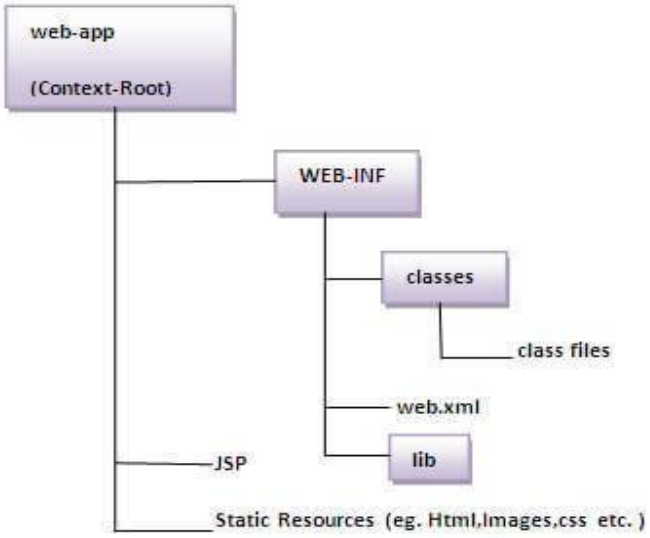

Fig. 5.10 Directory structure of the web application

5.6.1 MVC Architecture

The Model-View-Controller (MVC) is a well-known design pattern in the web development field. It is way to organize our code. It specifies that a program or application shall consist of data model, presentation information and control information. The MVC pattern needs all these components to be separated as different objects. This project we developed as web-based application which is having model view controller.

The model designs based on the MVC architecture follow MVC design pattern. The application logic is separated from the user interface while designing the software using model designs. The MVC pattern architecture consists of three layers:

- **Model**: It represents the business layer of application. It is an object to carry the data that can also contain the logic to update controller if data is changed. In our project, Model is the .java files.
- **View**: It represents the presentation layer of application. It is used to visualize the data that the model contains. In our project, View is the .jsp files which giving the user interface of our application.
- **Controller**: It works on both the model and view. It is used to manage the flow of application, i.e. data flow in the model object and to update the view whenever data is changed. In our project, Controller is the web.xml file which is controlling our web application

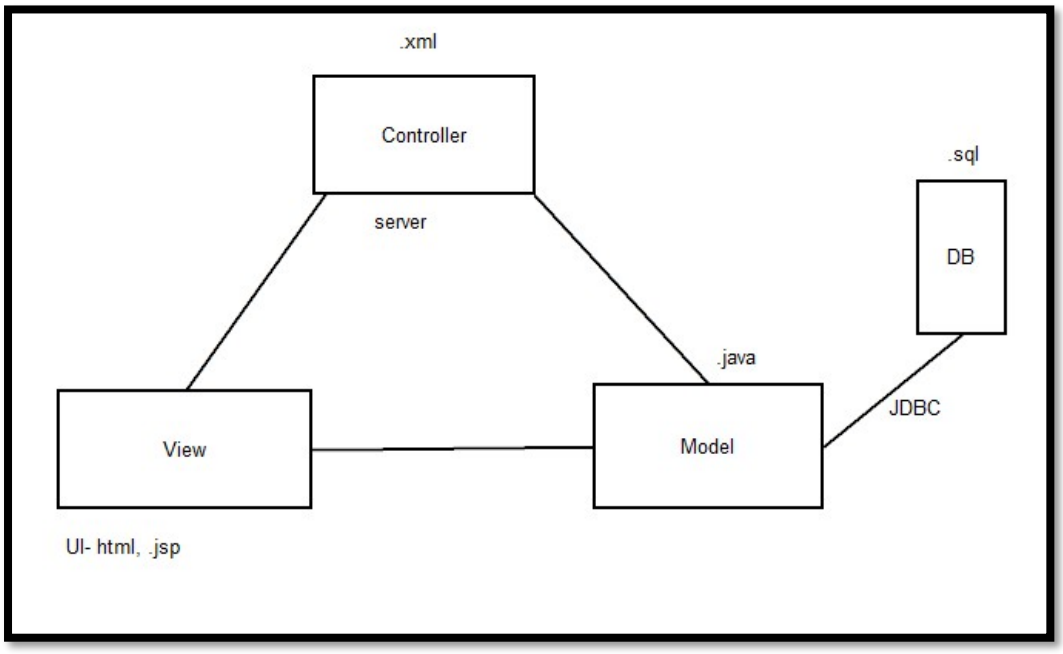

Fig. 5.11 MVC Architecture

5.6.2 Servlets

Servlets technology is used to create a web application (resides at server side and generates a dynamic web page).

Servlet technology is robust and scalable because of java language. There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

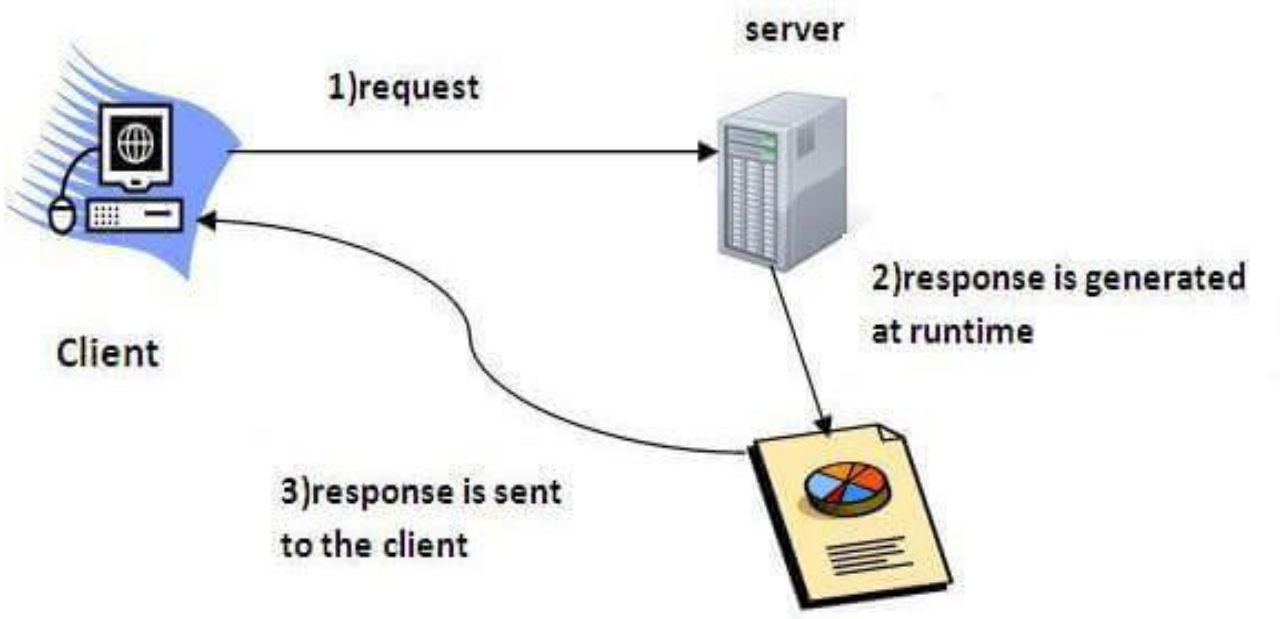

Fig. 5.12 Servlets Technology working

What is a Servlet?

- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.

HTTP

The Hypertext Transfer Protocol (Http) is application-level protocol for collaborative, distributed, hypermedia information systems. It is the data communication protocol used to establish communication between client and server.

There are given 6 steps to create a **servlet example**.

- Create a directory structure
- Create a Servlet
- Compile the Servlet
- Create a deployment descriptor
- Start the server and deploy the project
- Access the servlet

These steps are required for all the servers. The servlet example can be created by three ways:

- By implementing Servlet interface,
- By inheriting GenericServlet class, (or)
- By inheriting HttpServlet class
- The mostly used approach is by extending HttpServlet because it provides http request specific method such as doGet(), doPost(), doHead() etc.

The **deployment descriptor** is an xml file, from which Web Container gets the information about the servlet to be invoked. Web.xml file is the deployment descriptor in our project.

Sample web.xml file structure is given above. The explanation for each line is given below:

- <web-app> represents the whole application.
- <servlet> is sub element of <web-app> and represents the servlet.
- <servlet-name> is sub element of <servlet> represents the name of the servlet.
- <servlet-class> is sub element of <servlet> represents the class of the servlet.
- <servlet-mapping> is sub element of <web-app>. It is used to map the servlet.

- <url-pattern> is sub element of <servlet-mapping>. This pattern is used at client side to invoke the servlet.

```
<web-app>
<servlet>
<servlet-name>Login</servlet-name>
<servlet-class>com.admin.Login</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Login</servlet-name>
<url-pattern>/Login</url-pattern>
</servlet-mapping>
</web-app>
```

The **Request Dispatcher** interface provides the facility of dispatching the request to another resource it may be html, Servlet or JSP. The Forward method: Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

5.6.3 JSP

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than Servlet.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development.

The scripting elements provide the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag
- expression tag
- declaration tag

5.6.4 JDBC:

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a

consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

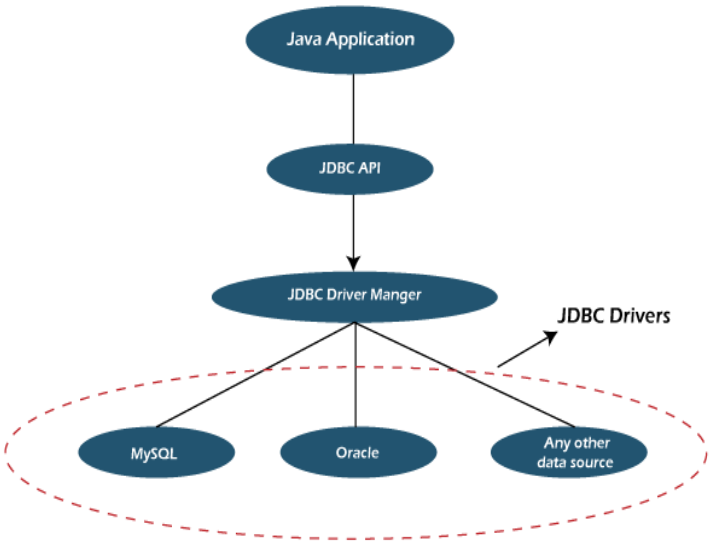

Fig. 5.13 JDBC Architecture

Here the DriverManager plays an important role. It uses some database specific drivers to communicate with the J2EE application to database. The DriverManager class is the component of JDBC API and also a member of the java.sql package. The DriverManager class acts as an interface between users and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. In our project, we are using the com.mysql.jar file as the JDBC Driver.

5 Steps involved in connecting a Java Application with Database using a JDBC:

- **Register the Driver.**

To begin with, you first need load the driver or register it before using it in the program. Registration is to be done once in your program. This step is mandatory.

Class.forName("com.mysql.jdbc.Driver");

- **Create a Connection.**

After loading the driver, establish connections using:

Connection con = DriverManager.getConnection(url,user,password)

- **Create SQL Statement.**

Once a connection is established you can interact with the database. Use of JDBC Statement is as follows:

Statement st = con.createStatement();

- **Execute SQL Statement.**

Now comes the most important part i.e executing the query. Query here is an SQL Query. Now we know we can have multiple types of queries. Some of them are as follows:

Query for updating / inserting table in a database.

Query for retrieving data.

- **Closing the connection.**

The close() method of Connection interface is used to close the connection.

This step is optional. Example : con.close();

The sample code for the JDBC establishment is given below.

```
import java.sql.*;
 class AdminDAO
    {
       public static void main(String args[])
       {
        try
          {   Class.forName("com.mysql.jdbc.Driver");    //Loading the driver

             //Creating the comnnection
             Connection con=DriverManager.getConnection(  "jdbc:mysql://localhost:3306/db_name" , "root", "admin");

             //Creating the statement
             Statement stmt=con.createStatement();
                        ResultSet rs=stmt.executeQuery("select * from  m_admin");  //Executing the query
                        while(rs.next()) {
                                   System.out.println(rs.getString(1)+"  "+rs.getString(2));
                                    }
               con.close();    //Closing the connection
        }
         catch(Exception e)
              { System.out.println(e);
              }
     }
}
```

5.7 Eclipse

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug, including Ada, ABAP, C, C++, C#, Clojure, COBOL, D, Erlang, Fortran, Groovy, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, and Scheme. It can also be used to develop documents with LaTeX (via a TeXlipse plug-in) and packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

5.8 Tomcat

Apache Tomcat, often referred to as Tomcat Server, is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several JavaEE specifications including Java Servlet, Java Server Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run.

5.9 MySQL

MySQL ("My Sequel") is (as of 2008) the world's most widely used open source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python. "Free-software open source projects that require a full-featured database management system often use MySQL.

MySQL is a relational database management system (RDBMS), and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools, or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. MySQL, like most other transactional relational databases, is strongly limited by hard disk performance.

5.10 Algorithm

RSA Algorithm :

RSA (**Rivest–Shamir–Adleman**) is a public key cryptosystem, one of the oldest widely used for secure data transmission. The initialism "RSA" comes from the surnames of Ron Rivest, Aid Shamir and Leonard Adleman, who publicly described the algorithm in 1977. An equivalent system was developed secretly in 1973 at Government Communications Headquarters (GCHQ), the British signals intelligence agency, by the English mathematician Clifford Cocks. That system was declassified in 1997.

In a public-key cryptosystem , the encryption key is public and distinct from the decryption key, which is kept secret (private). An RSA user creates and publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers are kept secret. Messages can be encrypted by anyone, via the public key, but can only be decrypted by someone who knows the private key.

The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem". Breaking RSA encryption is known as the RSA problem. Whether it is as difficult as the factoring problem is an open question. There are no published methods to defeat the system if a large enough key is used.

RSA is a relatively slow algorithm. Because of this, it is not commonly used to directly encrypt user data. More often, RSA is used to transmit shared keys for symmetric-key cryptography, which are then used for bulk encryption–decryption.

The RSA algorithm involves four steps:

- Key generation
- Key distribution
- Encryption
- Decryption.

A basic principle behind RSA is the observation that it is practical to find three very large positive integers e, d, and n, such that for all integers m ($0 \leq m < n$), both

$(m^e)^d = m \pmod n$

However, when given only e and n, it is extremely difficult to find d.

The integers n and e comprise the public key, d represents the private key, and m represents the message. The modular exponentiation to e and d corresponds to encryption and decryption, respectively.

In addition, because the two exponents can be swapped, the private and public key can also be swapped, allowing for message signing and verification using the same algorithm.

A cryptographically strong random number generator, which has been properly seeded with adequate entropy, must be used to generate the primes p and q. An analysis comparing millions of public keys gathered from the Internet was carried out in early 2012 by Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung and Christophe Wachter. They were able to factor 0.2% of the keys using only Euclid's algorithm.

Key generation

The keys for the RSA algorithm are generated in the following way:

- **Choose two large prime numbers p and q.**
- To make factoring harder, p and q should be chosen at random, be both large and have a large difference.

For choosing them the standard method is to choose random integers and use a primality test until two primes are found.

- p and q are kept secret.
- **Compute n = pq.**
- n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
- n is released as part of the public key.
- **Compute λ(n), where λ is Carmichael's totient function.**
  Since n = pq, λ(n) = lcm(λ(p), λ(q)), and since p and q are prime, λ(p) = φ(p) = p − 1, and likewise λ(q) = q − 1. Hence λ(n) = lcm(p − 1, q − 1).
- The lcm may be calculated through the Euclidean algorithm, since

$$\operatorname{lcm}(a, b) = \frac{|ab|}{\gcd(a, b)}.$$

- λ(n) is kept secret.
- **Choose an integer e such that 1 < e < λ(n) and gcd(e, λ(n)) = 1; that is, e and λ(n) are coprime.**
- e having a short bit-length and small Hamming weight results in more efficient encryption – the most commonly chosen value for e is 216 + 1 = 65537. The smallest (and fastest) possible value for e is 3, but such a small value for e has been shown to be less secure in some settings.
- e is released as part of the public key.
- **Determine d as d ≡ e−1 (mod λ(n)); that is, d is the modular multiplicative inverse of e modulo λ(n).**
- This means: solve for d the equation de ≡ 1 (mod λ(n)); d can be computed efficiently by using the extended Euclidean algorithm, since, thanks to e and λ(n) being coprime, said equation is a form of Bézout's identity, where d is one of the coefficients.
- d is kept secret as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the private (or decryption) exponent d, which must be kept secret. p, q, and λ(n) must also be kept secret because they can be used to calculate d. In fact, they can all be discarded after d has been computed.

In the original RSA paper, the Euler totient function φ(n) = (p − 1)(q − 1) is used instead of λ(n) for calculating the private exponent d. Since φ(n) is always divisible by λ(n), the algorithm works as well. The possibility of using Euler totient function results also from Lagrange's theorem applied to the multiplicative group of integers modulo pq. Thus any d satisfying d·e ≡ 1 (mod φ(n)) also satisfies d·e ≡ 1 (mod λ(n)). However, computing d modulo φ(n) will sometimes yield a result that is larger than necessary (i.e. d > λ(n)). Most of the implementations of RSA will accept exponents generated using either method (if they use the private exponent d at all, rather than using the optimized decryption method based on the Chinese remainder

theorem described below), but some standards such as FIPS 186-4 (Section B.3.1) may require that $d < \lambda(n)$. Any "oversized" private exponents not meeting this criterion may always be reduced modulo $\lambda(n)$ to obtain a smaller equivalent exponent.

Since any common factors of $(p - 1)$ and $(q - 1)$ are present in the factorisation of $n - 1 = pq - 1 = (p - 1)(q - 1) + (p - 1) + (q - 1)$,[17][self-published source?] it is recommended that $(p - 1)$ and $(q - 1)$ have only very small common factors, if any, besides the necessary.

Key distribution

Suppose that Bob wants to send information to Alice. If they decide to use RSA, Bob must know Alice's public key to encrypt the message, and Alice must use her private key to decrypt the message.

To enable Bob to send his encrypted messages, Alice transmits her public key (n, e) to Bob via a reliable, but not necessarily secret, route. Alice's private key (d) is never distributed.

Encryption

After Bob obtains Alice's public key, he can send a message M to Alice.

To do it, he first turns M (strictly speaking, the un-padded plaintext) into an integer m (strictly speaking, the padded plaintext), such that $0 \leq m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c, using Alice's public key e, corresponding to

$C \equiv m^e \pmod{n}$.

This can be done reasonably quickly, even for very large numbers, using modular exponentiation. Bob then transmits c to Alice. Note that at least nine values of m will yield a ciphertext c equal to m, but this is very unlikely to occur in practice.

Decryption

Alice can recover m from c by using her private key exponent d by computing

$c^d \equiv (m^e)^d \equiv m \pmod{n}$

Given m, she can recover the original message M by reversing the padding scheme.

Advantages

- **Security:** RSA algorithm is considered to be very secure and is widely used for secure data transmission.
- **Public-key cryptography:** RSA algorithm is a public-key cryptography algorithm, which means that it

uses two different keys for encryption and decryption. The public key is used to encrypt the data, while the private key is used to decrypt the data.

- **Key exchange:** RSA algorithm can be used for secure key exchange, which means that two parties can exchange a secret key without actually sending the key over the network.
- **Digital signatures:** RSA algorithm can be used for digital signatures, which means that a sender can sign a message using their private key, and the receiver can verify the signature using the sender's public key.
- **Widely used:** Online banking, e-commerce, and secure communications are just a few fields and applications where the RSA algorithm is extensively developed.

5.11 Modules

Module 1: FTPUpload.java

- The **FTPUpload** module in the provided Java servlet is designed to handle secure file processing and uploading to cloud servers.
- It begins by validating the user session and retrieving relevant session attributes such as username and file details. The module encrypts the uploaded file using RSA encryption for confidentiality, ensuring secure storage and transmission.
- A priority level (High, Medium, or Low) is dynamically assigned based on the text analysis of the file content. For integrity, the module generates an MD5 hash of the file, along with a unique nonce for transaction identification.
- These details are bundled into a ZIP archive containing the encrypted file and a confidential key file with metadata.
- A random selection algorithm determines the cloud server (from multiple options) where the file is uploaded via FTP. The module logs transaction details in the database and notifies the user via email upon successful upload. This structured approach ensures the process is secure, transparent, and efficient.

Module 2: Download.java

- The **Download.java** servlet handles secure file downloads by retrieving and decrypting files from cloud storage.
- It begins by validating the user session and retrieving necessary details, such as the file ID, cloud ID, and block ID, from the database. Based on the cloud ID, it determines the appropriate cloud directory and uses FTP credentials to download the file. If the file is part of a ZIP archive, it is extracted to retrieve the encrypted file.

- The servlet then decrypts the file using RSA encryption with a private key fetched from the database, ensuring data confidentiality. Once the file is successfully decrypted and saved to a local directory, the servlet logs the transaction details, including user and file identifiers, into the database.
- Finally, it redirects the user to a success or error page depending on the outcome of the download process. This design ensures secure, efficient, and traceable file retrieval for users.

Module 3: HashingTechnique.java

- The code implements a hashing technique using the MD5 algorithm. The MD5 method takes a string input (data), processes it using Java's MessageDigest class to compute the MD5 hash, and returns the result as a hexadecimal string.
- It converts the resulting hash bytes into a BigInteger to ensure proper formatting and generates a human-readable hexadecimal output. The code includes error handling to catch exceptions, ensuring robustness if the MD5 algorithm encounters issues.
- This implementation is suitable for generating unique fingerprints of data but is not recommended for security-critical applications, as MD5 is considered cryptographically weak.

Module 4: EncryptionDecryption.java

- The code implements file encryption and decryption using the RSA algorithm. It begins by generating a 1024-bit RSA key pair consisting of a public and private key using the KeyPairGenerator class.
- These keys are stored securely in both a database (via serialization) and as files on disk, enabling future reuse. The public key is used to encrypt files, ensuring confidentiality, while the private key decrypts the encrypted files back to their original form.
- The core logic for encryption and decryption is handled by the encryptDecryptFile method, which processes files in chunks using the Cipher class initialized with the RSA algorithm and the "RSA/ECB/PKCS1Padding" scheme.
- Utilities such as SaveKeyToFile and readPublicKey or readPrivateKey facilitate storing and retrieving the keys from files, while helper methods like encrypt and decrypt manage the conversion of data during encryption and decryption.
- The main method demonstrates the complete process, from key generation to encrypting and decrypting files.
- This implementation ensures secure handling of sensitive data, leveraging RSA's asymmetric encryption for robust file security.

Module 5: UploadFile.java

- The **UploadFile.java** servlet handles file uploads by parsing multipart form-data requests. It validates the content type, extracts file metadata such as name and content type, and writes the uploaded file to a specified server directory.
- The servlet dynamically generates the file path using the deployment directory and ensures compatibility with operating systems like Windows and Linux. After storing the file, it saves key details like the file path and name in the session for further processing.
- Once the upload is complete, the request is forwarded to the FTPUpload servlet for subsequent operations like encryption and cloud storage. Robust error handling ensures the servlet manages failures gracefully, maintaining a seamless user experience.

Module 6: GetFiles.java

- The **GetFiles.java** servlet is designed to retrieve, verify, and validate files stored on cloud servers. It starts by handling user input, including the file name and requested action (e.g., "Ok" or "Verify").
- **Ok** : it retrieves the file details and redirects the user to a page displaying file information.
- **Verify** : the servlet validates the selected file against its hash value stored in the blockchain to ensure data integrity.

Module 7: GenerateKey.java

- The **GenerateKey** class is a servlet designed to generate RSA encryption keys. When a POST request is made to this servlet, it determines the file paths for storing the public and private keys using the server's real path (req.getRealPath()).
- It then calls a method from the RSA_File_EncryptionDecryption utility class to generate the RSA key pair and save them in the specified file paths.
- Based on the success of the operation, the servlet forwards the user to a JSP page (generate_key.jsp) with a status code (no=1 for success, no=2 for failure) appended as a query parameter. It also handles exceptions by logging the error details to the console. **protected void**.
- Choose two large prime numbers p and q, Compute $n = pq$ and $\lambda(n)$.
- Choose an integer e such that $1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1$; that is, e and $\lambda(n)$ are coprime.
- Determine d as $d \equiv e^{-1} \pmod{\lambda(n)}$; that is, d is the modular multiplicative inverse of e modulo $\lambda(n)$.

Chapter 6

TESTING AND RESULTS

6.1 Software Testing Introduction

Software testing is a process used to help identify the correctness, completeness and quality of developed computer software. Software testing is the process used to measure the quality of developed software .Testing is the process of executing a program with the intent of finding errors. Software testing is often referred to as verification & validation

6.2 Explanation for SDLC and STLC

SDLC: The software development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application.

6.3 Phases of Software Development

- Requirement Analysis
- Software design
- Development or Coding
- Testing
- Maintenance

6.3.1 Requirement Analysis

The requirements of a desired software product are extracted. Based the business scenario the SRS (Software Requirement Specification) document is prepared in this phase.

6.3.2 Design

Plans are laid out concerning the physical construction, hardware, operating systems, programming, communications, and security issues for the software. Design phase is concerned with making sure the software system will meet the requirements of the product.

There are 2 stages in design,

HLD – High Level Design

LLD – Low Level Design

HLD – A High-Level Design (HLD) document provides an overview of a system's architecture, components, and interactions. It serves as a blueprint for developers, outlining the major modules, their functions, and how they communicate. HLD typically includes system architecture diagrams, data flow, database design, and technology stack details.

It bridges the gap between requirement specifications and low-level design (LLD), ensuring clarity in system structure. Key elements include module descriptions, interfaces, dependencies, and security considerations. It helps stakeholders understand the overall design without delving into code-level details. HLD ensures alignment with business objectives, scalability, and maintainability. It also facilitates team collaboration and serves as a reference throughout the development lifecycle. By defining high-level interactions and dependencies, it minimizes risks in implementation. A well-prepared HLD enhances efficiency, reducing rework and improving software quality.

LLD – A Low-Level Design (LLD) document provides a detailed, module-specific blueprint for system implementation. It breaks down the High-Level Design (HLD) into smaller components, defining classes, methods, data structures, and algorithms. LLD includes detailed flowcharts, sequence diagrams, and pseudocode to guide developers in coding. It specifies database schema details, API definitions, and error handling mechanisms. Every module's logic, dependencies, and interactions are precisely documented to ensure consistency and maintainability.

LLD focuses on implementation-level details, ensuring the system meets functional and non-functional requirements. It helps developers write efficient, scalable, and secure code while maintaining coding standards. This document aids in debugging, testing, and future enhancements. LLD is crucial for ensuring smooth development, reducing ambiguity, and minimizing integration issues. A well-structured LLD improves software quality and accelerates the development process.

6.3.3 Testing

Testing is evaluating the software to check for the user requirements. Here the software is evaluated with intent of finding defects.

6.3.4 Maintenance

Once the new system is up and running for a while, it should be exhaustively evaluated. Maintenance must be kept up rigorously at all times. Users of the system should be kept up-to-date concerning the latest modifications and procedures

6.4 SDLC Models

6.4.1 Water Fall Model

The **Waterfall Model** is a traditional software development methodology that follows a linear and sequential approach. It consists of several distinct phases, including:

- Requirement Gathering – Understanding and documenting all customer needs.
- System Design – Creating architectural and detailed design plans.

- Implementation (Coding) – Writing code based on design specifications.
- Testing – Verifying that the developed system meets the requirements.
- Deployment – Releasing the final product to users.
- Maintenance – Handling bug fixes and updates after deployment.

6.4.2 Proto Type Model

The Prototype Model is an iterative development approach where an initial version (prototype) of the software is created based on initial requirements. This prototype is shared with users, who provide feedback, and refinements are made until the final system is developed.

Key Steps in the Prototype Model:

- Requirement Analysis – Understanding initial needs.
- Quick Design – Creating a rough system blueprint.
- Prototype Development – Building a functional but incomplete version.
- User Evaluation & Feedback – Collecting feedback to improve the prototype.
- Refinement & Final Development – Repeating the cycle until a satisfactory system is built.
- 6. Deployment & Maintenance – Delivering the final product and making necessary updates.

6.4.3 Rapid Application Development Model (RAD):

The RAD Model focuses on speed and flexibility, using component-based development and continuous user involvement. It breaks the project into small modules, which are developed independently and then integrated into the final system.

Phases of RAD Model:

- Requirement Planning – Gathering initial requirements quickly.
- User Design (Prototyping & Feedback) – Rapidly developing user interfaces and functional modules.
- Construction – Rapid coding, testing, and integration using reusable components.
- Cutover (Deployment & Maintenance) – Delivering the product with final optimizations.

6.4.4 Spiral Model

The Spiral Model is a risk-driven approach that combines iterative and Waterfall-like development processes. It is suitable for large, high-risk, and complex projects. The process is divided into several spiral cycles, each consisting of four stages:

- Planning – Defining objectives, constraints, and alternatives.
- Risk Analysis – Identifying and addressing potential risks.

- Development & Testing – Building and validating the system in increments.
- Evaluation – Reviewing progress and deciding whether to continue or refine.

6.4.5 V-Model

The V-Model is an enhanced version of the Waterfall Model, where each development phase has a corresponding testing phase. It ensures early defect detection, reducing the chances of issues in later stages**.**

Phases of the V-Model**:**

- Requirement Analysis → Acceptance Testing – Ensuring final software meets customer needs.
- High-Level Design → Integration Testing – Testing interactions between system modules.
- Low-Level Design → Unit Testing – Testing individual components for correctness.
- Coding – Writing and implementing the actual code.

6.5 STLC (Software Testing Life Cycle)

The Software Testing Life Cycle (STLC) is a structured process that defines the stages involved in testing a software product. It includes phases such as requirement analysis, test planning, test case development, test environment setup, test execution, and test closure.

Each phase has specific deliverables to ensure systematic and efficient testing. STLC helps in detecting defects early, improving software quality, and ensuring that the final product meets user expectations. By following STLC, organizations can achieve a well-defined and organized testing process.

Testing itself has many phases i.e. called as STLC. STLC is part of SDLC

- Test Plan
- Test Development
- Test Execution
- Analyze Results
- Defect Tracking
- Summaries Report

6.5.1 Test Plan:

It is a document which describes the testing environment, purpose, scope, objectives, test strategy, schedules, mile stones, testing tool, roles and responsibilities, risks, training, staffing and who is going to test the application, what type of tests should be performed and how it will track the defects.

6.5.2 Test Development:

Preparing test cases, test data, Preparing test procedure, Preparing test scenario, Writing test script

6.5.3 Test Execution:

In this phase we execute the documents those are prepared in test development phase

6.5.4 Analyze Result:

Once executed documents will get results either pass or fail. we need to analyze the results during this phase.

6.5.5 Defect Tracking:

Whenever we get defect on the application we need to prepare the bug report file and forwards to Test Team Lead and Dev Team. The Dev Team will fix the bug. Again we have to test the application. This cycle repeats till we get the software without defects.

6.6 Types of Testing:

- White Box Testing
- Black Box Testing
- Grey box testing

6.6.1 White Box Testing:

White Box Testing is a software testing technique that examines the internal structure, design, and code of an application. It requires knowledge of programming and focuses on verifying the flow of inputs and outputs through the system. Testers analyze control flow, data flow, loops, and statements to ensure accuracy and efficiency. Testers have full access to source code, allowing them to examine how inputs work .This method helps identify logical errors, security vulnerabilities, and hidden defects in the code. White Box Testing enhances software quality by improving code coverage and ensuring proper functionality.

6.6.2 Black Box Testing:

Black Box Testing is a software testing method that evaluates the functionality of an application without examining its internal code structure or logic. Testers focus on input-output behavior, ensuring the system meets user requirements and specifications. This approach is useful for detecting usability issues, incorrect outputs, and missing functionalities. Common techniques include equivalence partitioning, boundary value analysis, and exploratory testing. Black Box Testing enhances software reliability by validating its external behavior against expected outcomes.

6.6.3 Grey Box Testing:

Grey Box Testing is a software testing approach that combines elements of both White Box and Black Box testing. Testers have partial knowledge of the internal structure while primarily focusing on functional testing. This method helps identify security flaws, integration issues, and data flow problems by analyzing both internal processes and external outputs. It is commonly used in web applications, penetration testing, and database testing. Grey Box Testing improves software quality by balancing code-level analysis with user-perspective validation.

Testing Used for Web Based Application:

This is done for 3 tier applications (developed for Internet / intranet /Extranet).Here we will be having Browser, web server and DB server. The applications accessible in browser would be developed in HTML, DHTML, XML, JavaScript etc. (We can monitor through these applications)

6.7 Level of Testing Used In Project:

6.7.1 Unit Testing Cases:

Initialization testing is the first level of dynamic testing and is first the responsibility of developers and then that of the test engineers. Unit testing is performed after the expected test results are met or differences are explainable/acceptable**.**

Sl # Test Case : -	UNTC-1
Name of Test: -	Login as Admin
Items being tested: -	Admin model
Sample Input: -	Correct Username & Password is given as inputs
Expected output: -	Depending on the correct inputs, it must login as Admin
Actual output: -	Login successful
Remarks: -	Pass.

Table 6.1 Unit Test Case 1

Sl # Test Case : -	UNTC-2
Name of Test: -	Login as User
Items being tested: -	User model
Sample Input: -	Incorrect Username & Password is given as inputs
Expected output: -	Depending on the incorrect inputs, it shouldn't login as User
Actual output: -	Login Failed.
Results:-	Pass

Table 6.2 Unit Test Case 2

Do the changes of wrong values of username & password in data base with correct username & correct password and save the database.

6.7.2 System Testing

To test the complete system in terms of functionality and non functionality. It is black box testing, performed by the Test Team, and at the start of the system testing the complete system is configured in a controlled environment.

Sl # Test Case : -	STC-1
Name of Test: -	System testing in various versions of OS
Sample Input: -	Execute the program in windows server 2003/XP/ Windows-7
Expected output: -	Performance is better in windows-8
Actual output: -	Same as expected output, performance is better in windows-8
Remarks: -	Pass

Table 6.3 System Test Case 1

6.7.3 Integration Testing:

The outgoing links from all the pages from specific domain under test. Test all internal links. Test links jumping on the same pages. Check for the default values of fields. Wrong inputs to the fields in the forms.

Sl # Test Case : -	ITC-1
Name of Test: -	Change Password for Admin
Sample Input: -	Incorrect Old password, give new password.
Expected output: -	Failed to update new password.
Actual output: -	Password Update failed
Remarks: -	Pass.

Table 6.4 Integration Test Case 1

Sl # Test Case : -	ITC-2
Name of Test: -	RSA Encryption and Decryption
Sample Input: -	Text file with Complaint
Expected output: -	The original file is retrieved after decryption without any corruption or loss.
Actual output: -	Encryption & Decryption Passed.
Remarks: -	Pass.

Table 6.5 Integration Test Case 2

Sl # Test Case : -	ITC-3
Name of Test: -	Blockchain Immutability
Sample Input: -	Attempt to modify a record stored on the blockchain.

Expected output: -	Modification attempt fails, and the original data remains unchanged.
Actual output: -	Modification attempt fails.
Remarks: -	Pass.

Table 6.6 Integration Test Case 3

Sl # Test Case : -	ITC-4
Name of Test: -	Verify Document Integrity
Sample Input: -	Retrieve a document using its IPFS hash.
Expected output: -	Hashes match, confirming data integrity.
Actual output: -	Hashes matched correctly.
Remarks: -	Pass.

Table 6.7 Integration Test Case 4

Sl # Test Case : -	ITC-5
Name of Test: -	Role-Based Access Control
Sample Input: -	Login as different Police and check access to Complaint.
Expected output: -	Each Police can access only the Complaints assigned to it.
Actual output: -	Access Control integrated successfully.
Remarks: -	Pass.

Table 6.8 Integration Test Case 5

6.7.4 Alpha Testing:

The primary goal is to identify bugs, usability issues, and any gaps in functionality while ensuring the software meets its design and requirements. Alpha testing involves rigorous testing of the application using white-box and black-box techniques and may include simulated user behaviour.

6.7.5 Beta Testing:

This phase helps uncover issues that may not have been identified during alpha testing, such as compatibility problems, real-user workflows, or unforeseen edge cases. Beta testers provide valuable feedback that developers use to refine and stabilize the software before its official release.

6.7.6 Functional Testing:

It is a quality assurance (QA) process and a type of black-box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (unlike white-box testing).

Functional testing for the blockchain-based police complaint management system focuses on ensuring that the system meets its specified requirements and performs all intended operations seamlessly.

Sl # Test Case : -	FTC-1
Name of Test: -	Submit Complaint
Sample Input: -	Fill the file name and navigate the text document from the system and upload.
Expected output: -	Complaint is uploaded and Conformation mail is sent.
Actual output: -	Complaint is uploaded and Conformation mail is sent successfully.
Remarks: -	Pass.

Table 6.9 Functional Test Case 1

6.8 Results:

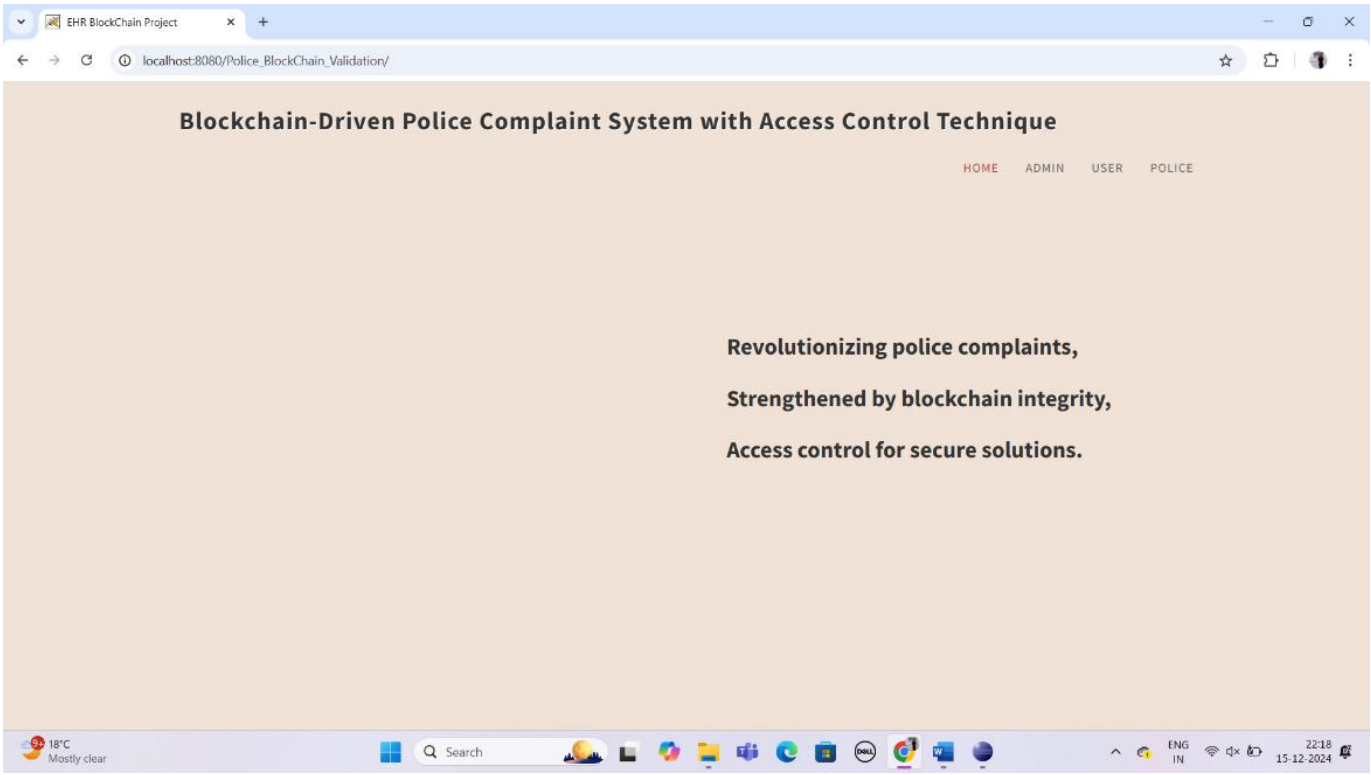

Fig 6.1 Home Screen

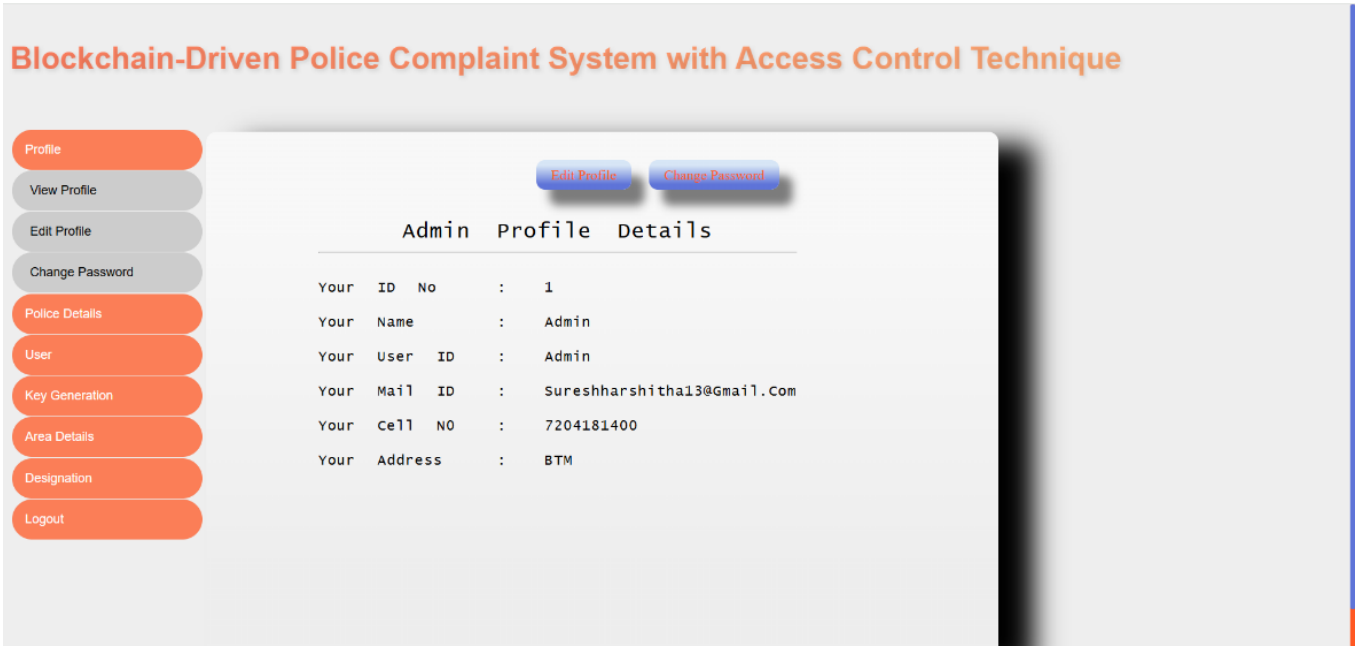

Fig 6.2 Admin Screen

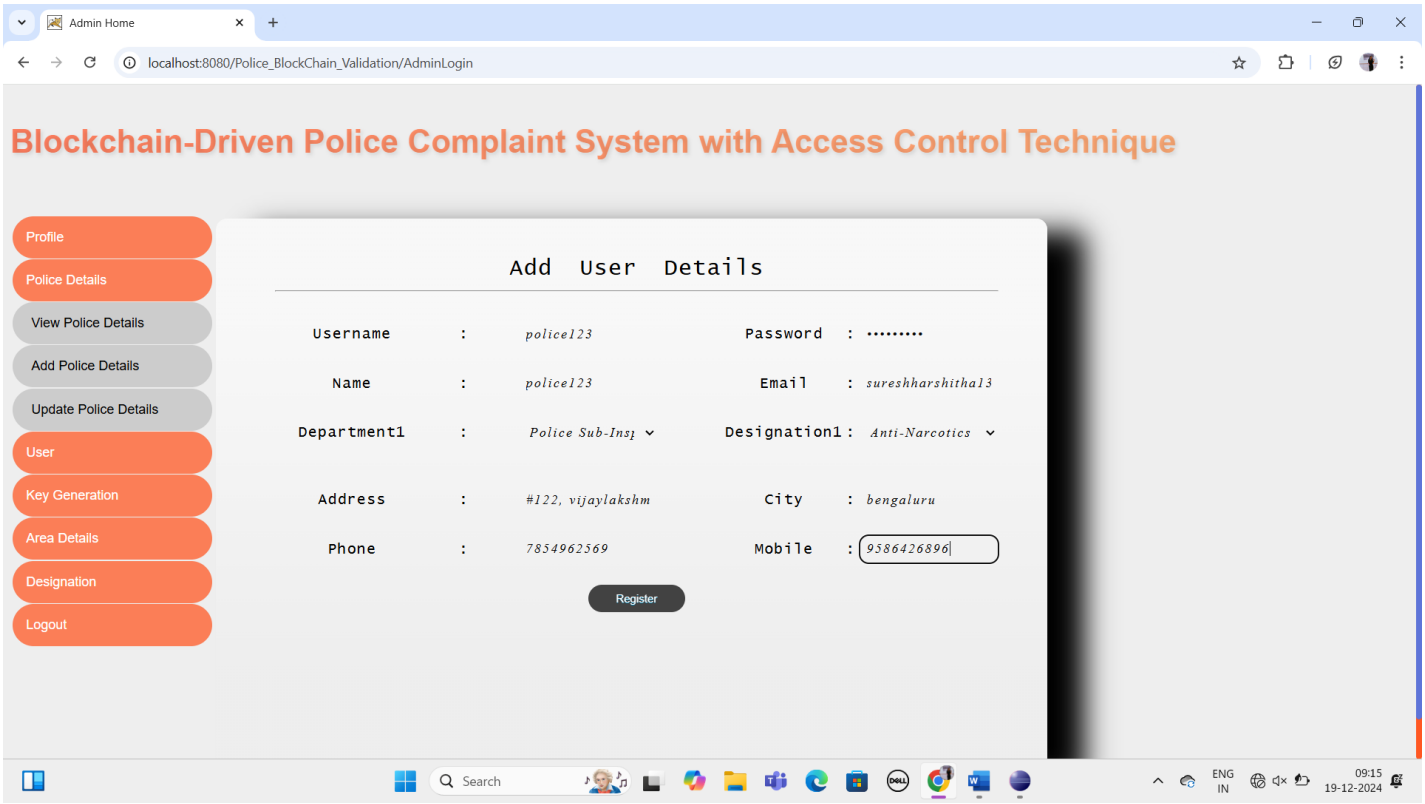

Fig 6.3 Police Data Creation

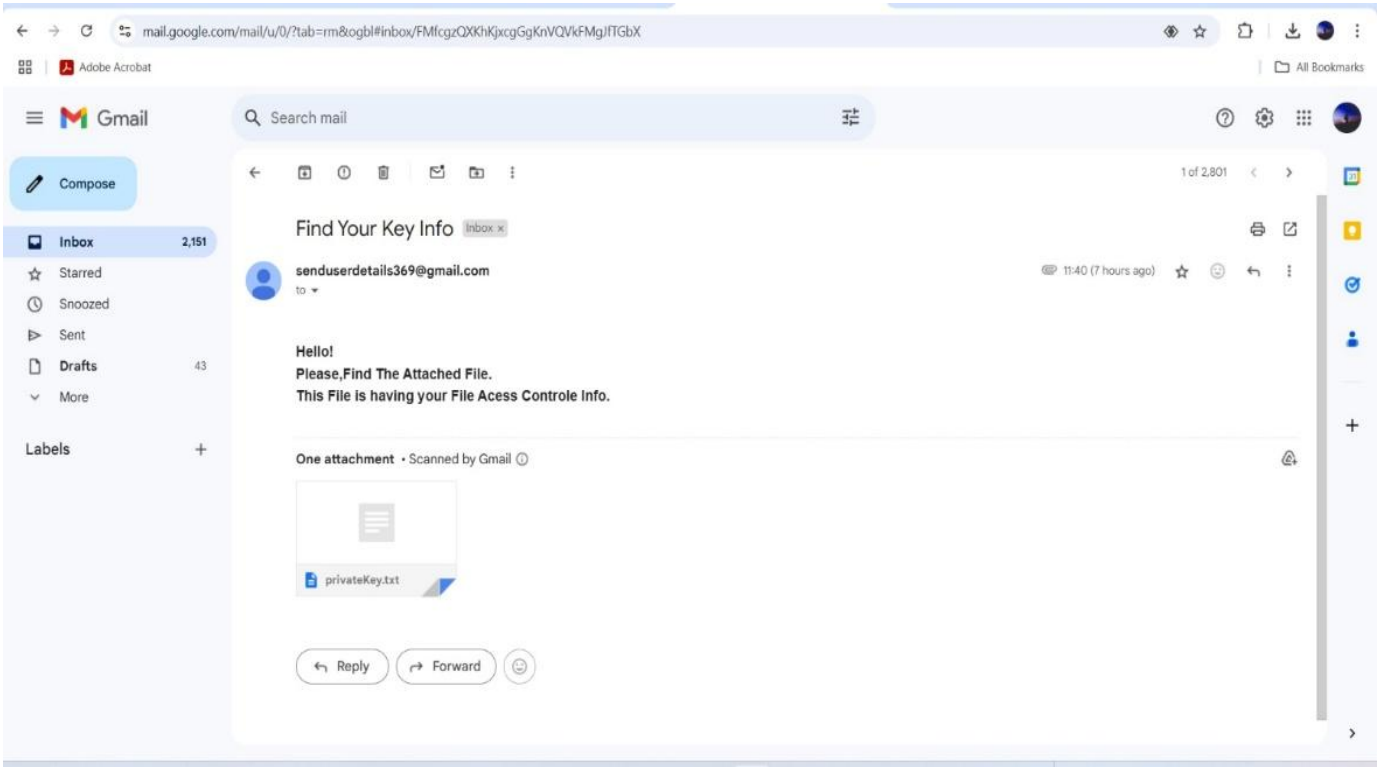

Fig 6.4 Private key Mail

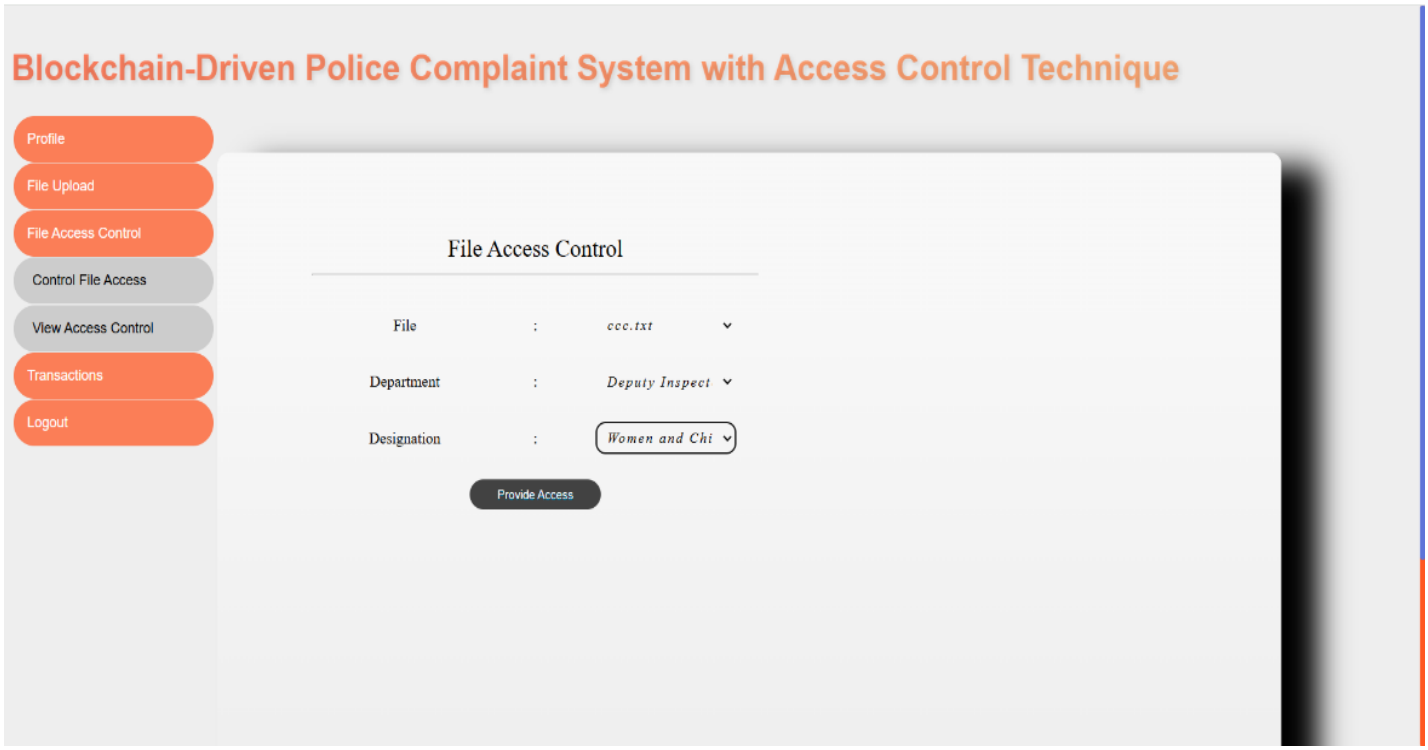

Fig 6.5 User Access Control

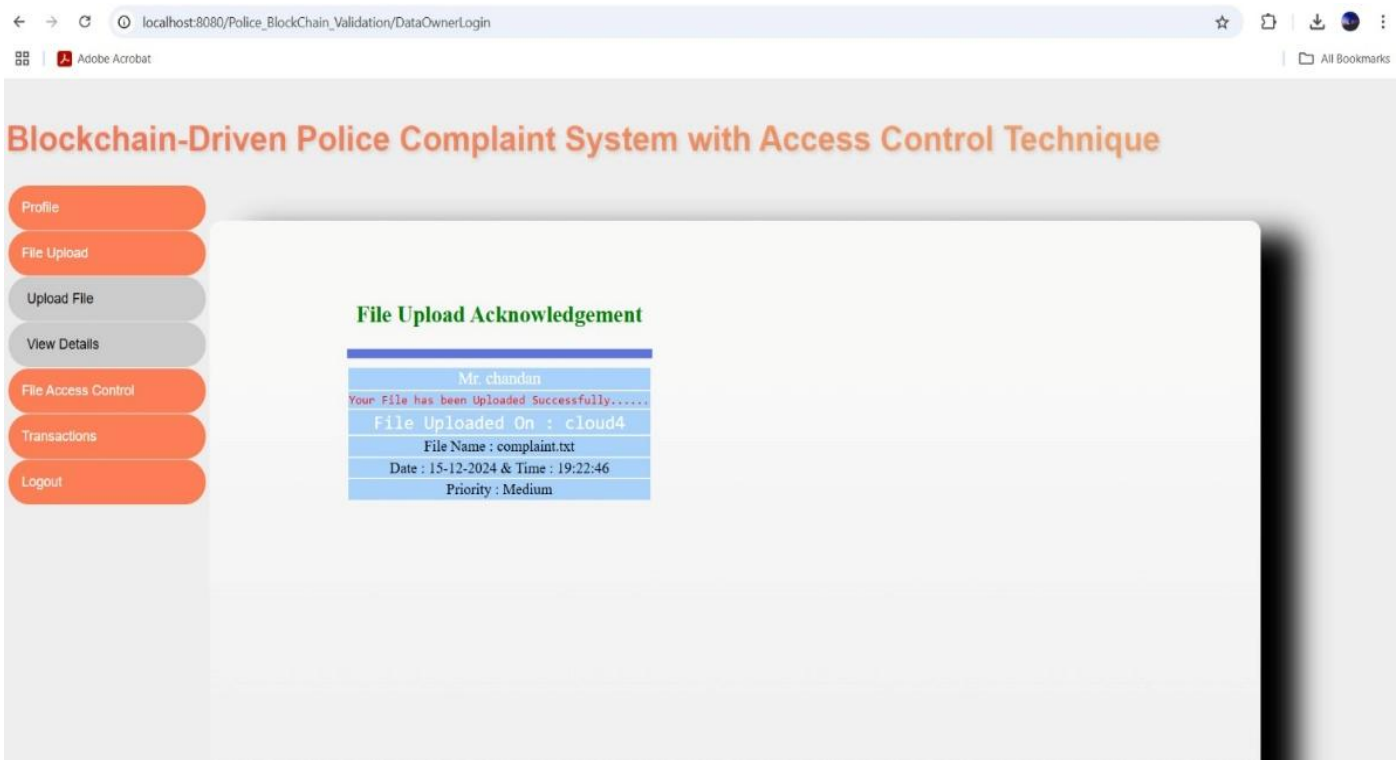

Fig 6.6 Complaint file upload

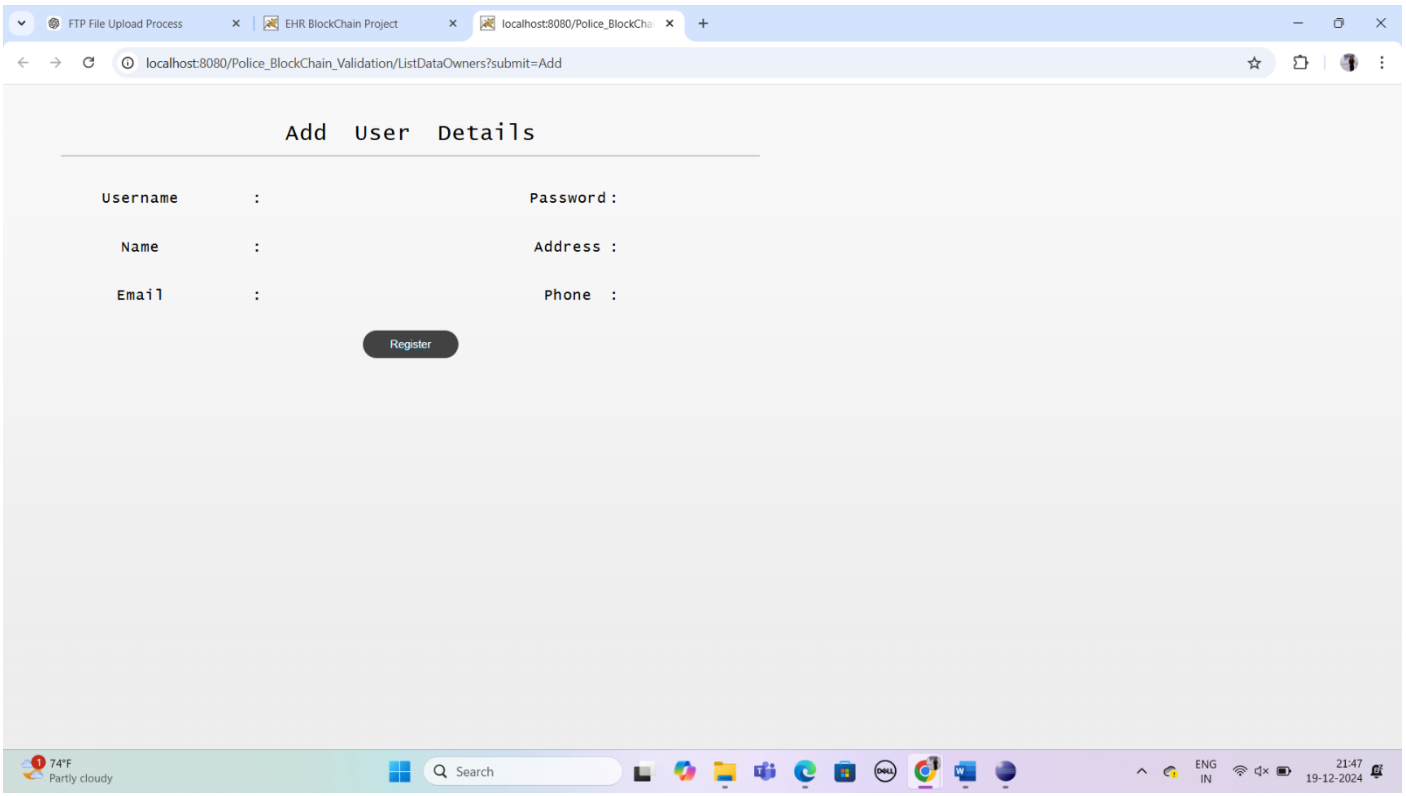

Fig 6.7 User Registration

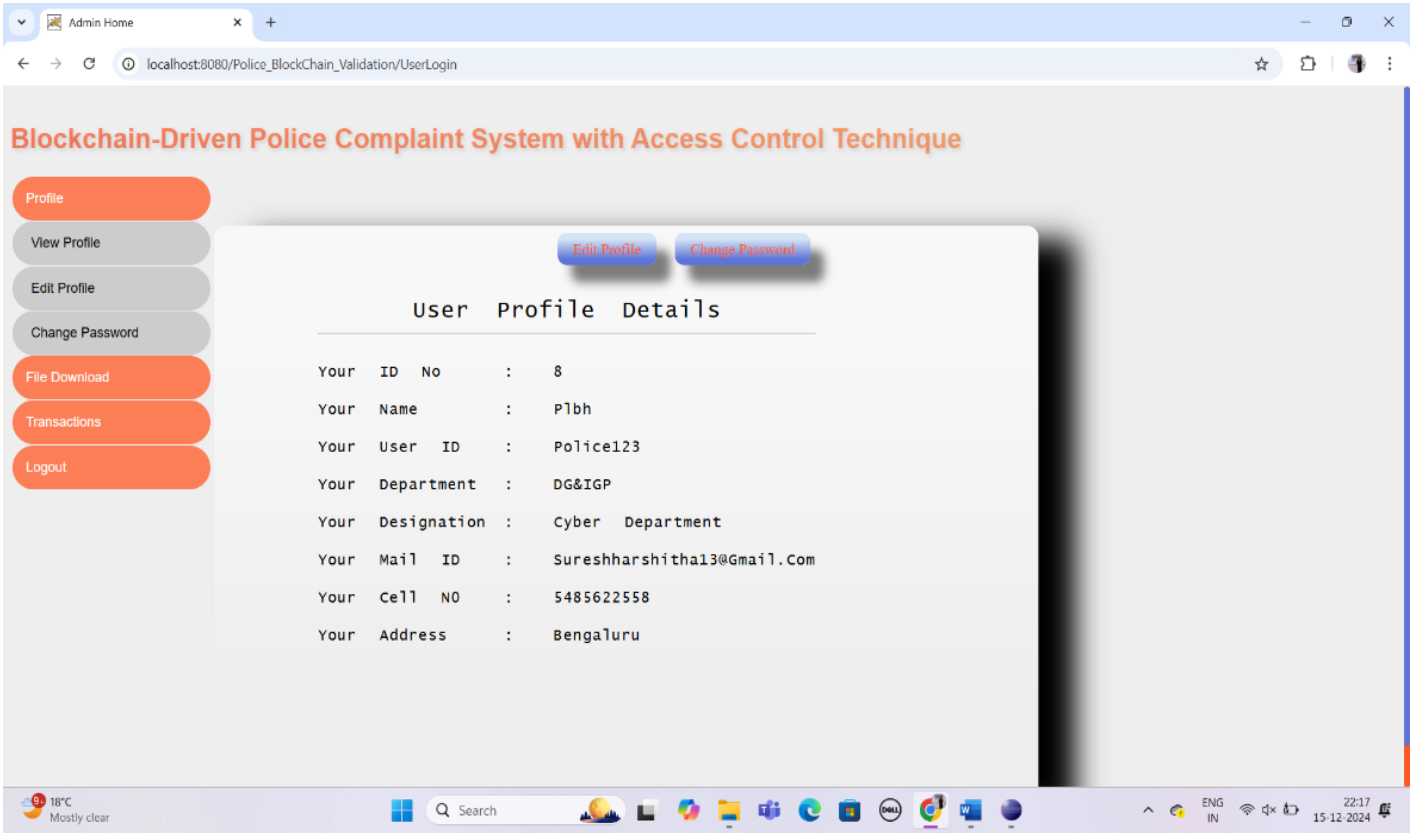

Fig 6.8 Police profile

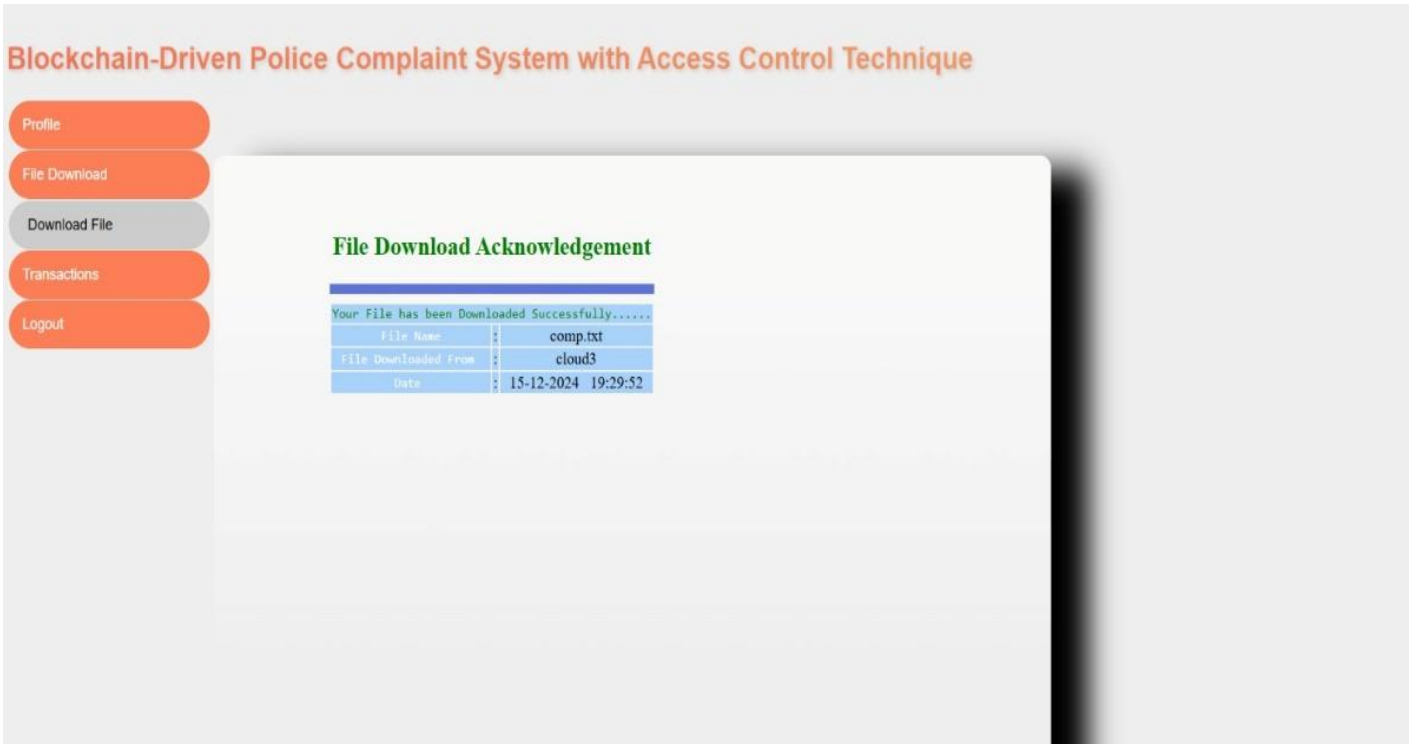

Fig 6.9 Complaint Download

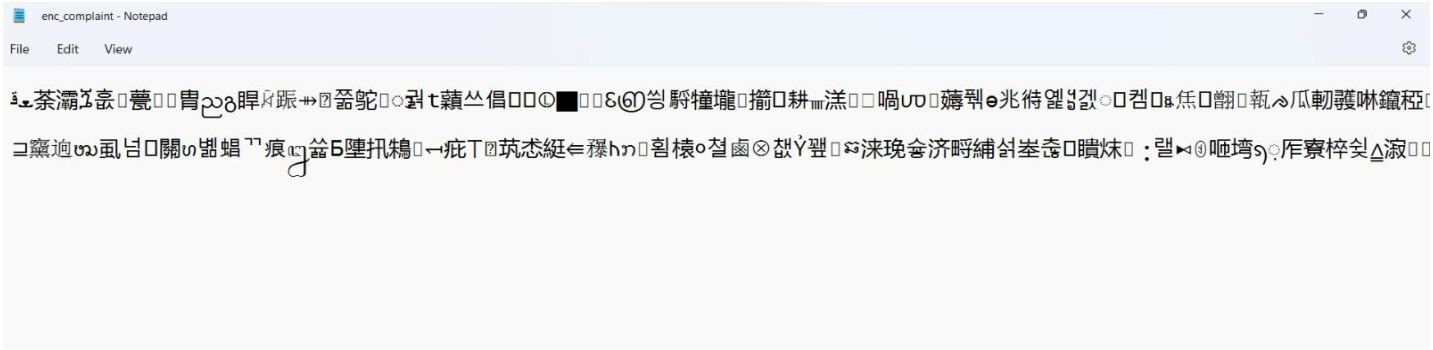

Fig 6.10 Encrypted File

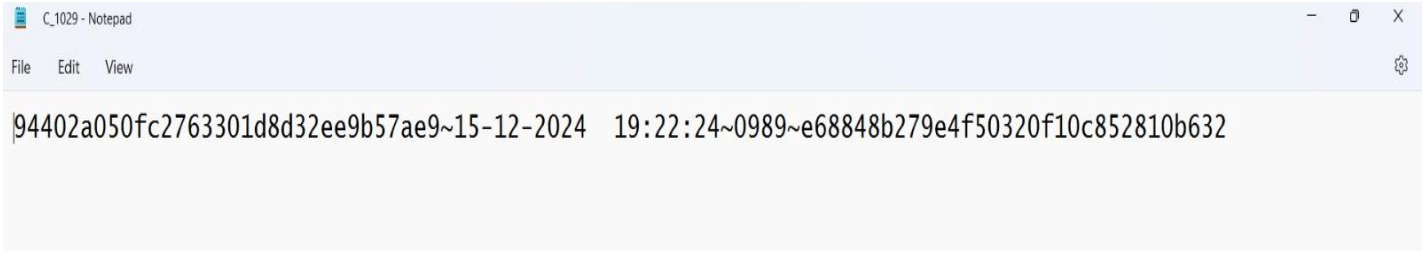

Fig 6.11 Blockchain values

Chapter 7

CONCLUSION

- The blockchain-based police complaint management system addresses the limitations of traditional FIR lodging methods and the centralized nature of existing systems like CCTNS.
- By leveraging blockchain technology, the system ensures secure, transparent, and efficient handling of both cognizable and non-cognizable crimes, thereby enhancing accountability and trust in law enforcement.
- The integration of encryption and public cloud networks provides a balance between data privacy and transparency, enabling secure access for users and authorized officers.
- This innovative approach promotes consistency in complaint management and fosters a more reliable, responsive, and credible police department, contributing to a safer and more trustworthy environment for citizens.
- Overall, this system represents a significant step toward creating a safer and more trustworthy environment for citizens. It sets the foundation for a modernized, technology-driven police infrastructure that prioritizes the needs of the public while adhering to principles of integrity, accountability, and transparency.

FUTURE WORK

- **Integration of Advanced AI:** Implement artificial intelligence for automated complaint analysis and prioritization based on severity, urgency, and crime patterns. AI-powered chatbots can assist in initial complaint filing, reducing manual workload and improving response times.

- **Scalability Enhancements:** Adopt robust blockchain frameworks such as Hyperledger Fabric or Ethereum Layer 2 solutions to enhance scalability. This will ensure the system can handle a growing volume of complaints efficiently without compromising speed or security.

- **Cross-Jurisdictional Collaboration:** Enable seamless crime handling across jurisdictions by allowing different law enforcement agencies to share, access, and update complaint records securely. Implementing a federated blockchain model can facilitate coordination between multiple police departments while maintaining data privacy.

- **Anonymous Reporting**: Introduce features for anonymous complaint submissions to encourage whistleblowers and victims of sensitive crimes (such as harassment or corruption) to report incidents without fear of retaliation. Blockchain ensures immutable and tamper-proof records, maintaining credibility.

- **Multilingual Support:** Expand the system's accessibility by adding support for multiple languages, allowing complainants from diverse linguistic backgrounds to file complaints in their preferred language. AI-driven translation tools can help in automatic conversion of complaints for easy processing.

- **Offline Functionality:** Enable complaint submission and access even in offline mode by incorporating decentralized storage solutions and progressive web applications (PWAs). This will ensure users in remote areas with limited internet connectivity can still report crimes, and data synchronization will occur once connectivity is restored.

REFERENCES

[1]. Shivaganesh Pillai, “Online Fir Registration and Sos System”, int. jour. eng. com. sci, vol. 5, no. 4, Dec. 2017. Omoregbe, Nicholas Misra, Sanjay Maskeliunas, Rytis Damasevicius, Robertas Adesola, Falade Adewumi, Adewole. (2019).

[2]. P. Kormpho, P. Liawsomboon, N. Phongoen and S. Pongpaichet, ”Smart Complaint Management System,” 2018 Seventh ICT International Student Project Conference (ICT-ISPC), Nakhonpathom, 2018, pp. 1-6, doi: 10.1109/ICT-ISPC.2018.85239

[3]. A. T. Dini, E. Gabriel Abete, M. Colombo, J. Guevara, B. S. Mench´on Hoffmann and M. Claudia Abeledo, ”Analysis of implementing blockchain technology to the argentinian criminal records information system,” 2018 Congreso Argentino de Ciencias de la Inform´atica yDesarrollos de Investigaci´on (CACIDI), Buenos Aires, 2018, pp. 1-3, doi: 10.1109/CACIDI.2018.8584365.

[4]. S.P. GodlinJesil, Rajat Basant, Pratishvir, “CRIME REPORTING SYSTEM USING ANDROID APPLICATION,” International Journal of Pure and Applied Mathematics, vol. 119,no. 7, 2018, https://acadpubl.eu/jsi/2018-119-7/articles/7a/56.pdf.

[5]. Archana M, Durga S, Saveetha K, “Online Crime Reporting System,” Int. Jnl. Of Advanced Networking Applications (IJANA),https://www.ijana.in/papers/82.pdf.

[6]. K. Tabassum, H. Shaiba, S. Shamrani and S. Otaibi, ”e-Cops: An Online Crime Reporting and Management System for Riyadh City,” 2018 1st International Conference on Computer Applications Information Security (ICCAIS), Riyadh, 2018, pp. 1-8, doi: 10.1109/CAIS.2018.8441987.

[7]. Imran Bashir, Mastering Blockchain: Unlocking the Power of Cryptocurrencies, Smart Contracts, and Decentralized Applications, Packt Publishing, 3rd Edition, 2020.

[8]. William Stallings, Cryptography and Network Security: Principles and Practice, Pearson, 8th Edition,2020.

NAME: ANUSHA B S
USN: 1EE21CS004
SEM: 7
BRANCH: CSE
PHONE NO: 9535308837
EMAIL ID: anushasiddu29@gmail.com

NAME: HARSHITHA S
USN: 1EE21CS019
SEM: 7
BRANCH: CSE
PHONE NO: 7204181400
EMAIL ID: sureshharshitha13@gmail.com

NAME: PRIYANKA H
USN: 1EE21CS039
SEM: 7
BRANCH: CSE
PHONE NO: 7019338259
EMAIL ID: priyankah669@gmail.com

NAME: RAMYA B
USN: 1EE21CS040
SEM: 7
BRANCH: CSE
PHONE NO: 9193251191
EMAIL ID: ramyabasavaraju21@gmail.com