Cost-Effectively Searchable Blackbox Data with Unlinkability Based on Public Blockchain

A PROJECT REPORT

Submitted by

BHUSETTY SAI MUKESH (211420104040)
PODALAKURU HARTHIK (211420104193)
VINJAMURU ROOPSAI (211420104307)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

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

APRIL 2024

PANIMALAR ENGINEERING COLLEGE
(An Autonmous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"Cost-Effectively Searchable Blackbox Data with Unlinkability Based on Public Blockchain"** is the bonafide work of **"BHUSETTY SAI MUKESH (211420104040) PODALAKURU HARTHIK (211420104193) VINJAMURU ROOPSAI (211420104307)"** who carried out the project work under my supervision.

Signature of the HOD with date	**Signature of the Supervisor with date**
Dr. L.JABASHEELA M.E.,Ph.D.,	**Dr.PJ.SATHISH KUMAR,M.TECH,Ph.D.,**
Professor and Head,	**Professor,**
Department of Computer Science andEngineering,	Department of Computer Science and Engineering,
Panimalar Engineering College,	Panimalar Engineering College,
Chennai – 123	Chennai - 123

Submitted for the Project Viva – Voce examination held on ______________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **BHUSETTY SAI MUKESH (211420104040) PODALAKURU HARTHIK(211420104193) VINJAMURU ROOPSAI (211420104307)** hereby declare that thisproject report titled **"Cost-Effectively Searchable Blackbox Data with Unlinkability Based on Public Blockchain" ,** under the guidance of **Dr.PJ.SATHISH KUMAR,M.TECH,Ph.D.,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

BHUSETTY SAI MUKESH (211420104040)
PODALAKURU HARTHIK(211420104193)
VINJAMURU ROOPSAI (211420104307)

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**.,for his benevolent words and fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors,**Tmt.C. VIJAYARAJESWARI,Dr.C.SAKTHIKUMAR,M.E., Ph.D., and Dr.SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.
We express my heartfelt thanks to **Dr. L. JABASHEELA,M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Dr.N.PUGHAZENDI, M.E, Ph.D.,** and **Dr.PJ.SATHISH KUMAR, M.Tech, Ph.D.,**and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

BHUSETTY SAI MUKESH (211420104040)
PODALAKURU HARTHIK(211420104193)
VINJAMURU ROOPSAI (211420104307)

ABSTRACT

A blackbox data can be used to identify the cause of an accident. And we may need to find blackbox videos recorded by the third party to accurately analyze the cause of the accident. One of the basic requirements of blackbox data is integrity, and this integrity can be provided by uploading the data to blockchains. However, the existing blackbox data storage systems have two big problems, cost and privacy. The transaction fees are the costs necessary to upload data to the blockchains. The transaction fees of the major public blockchain are getting higher. Therefore, if a user uses Bitcoin or Ethereum to upload its blackbox video, the costs could be very expensive. On the other hand, the private information such as driving route could be revealed from the uploaded information to the blockchain.We suggest a cost effectiv searchable blackbox data storage system with strong anonymity, i.e. unlinkability, using blockchains. In our scheme, the cost of registering a blackbox video is costtr n , where costtr is the cost necessary to upload a transaction to a blockchain and n is the number of uploading users at the same time. Therefore, our scheme is practical with public blockchains such as Bitcoin or Ethereum. In the paper, we suggest a cost-effective searchable black box data storage system with unlinkability using blockchains

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

JDK	Java Development Toolkit
DEX	Dalvik Executables
TCP	Transmission Control Protocol
IP	Internet Protocol
HTTP	Hyper Text Transfer Protocol
ADT	Android Development Tool

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

The aim of this project is to propose a novel approach to create a cost-effective and searchable blackbox data storage system utilizing blockchain technology .One of the basic requirements of blackbox data is integrity, and this integrity can be provided by uploading the data to blockchains. Blackbox data plays a crucial role in accident analysis, providing insights into the causes of incidents. The need for third-party blackbox videos for accurate analysis is evident. Ensuring the integrity of blackbox data is a fundamental requirement, and leveraging blockchains for data storage can address this need. However, existing blackbox data storage systems face challenges related to cost and privacy. However, the simple searchable blackbox storage system has some problems. First, the simple scheme reveals some private information of the users. To upload a hash of blackbox data to a blockchain, a user must make a transaction including the hash of blackbox data. Usually, the transaction is made by signing the hash of blackbox data using the user's private key. Because we can know whether any two transactions are signed by the same user, we can know when the user drove its car, even though we cannot see the videos nor metadata of the blackbox of the user. Usually, the transaction is made by signing the hash of blackbox data using the user's private key. Because we can know whether any two transactions are signed by the same user, we can know when the user drove its car, even though we cannot see the videos nor metadata of the blackbox of the user. Security and Privacy of The decentralized nature of blockchain technology ensures that the data and search process are secure and resistant to tampering. Additionally, privacy measures can be implemented to protect sensitive information within the blackbox. Overall, a co-effectively searchable blackbox with blockchain integration could provide a secure, transparent, and efficient means for multiple parties to search for information or data within a decentralized and opaque structure. Overall, a co-effectively searchable blackbox with blockchain integration could provide a secure, transparent, and efficient means for multiple parties to search for information or data within a decentralized and opaque structure.

CHAPTER 2

LITERATURE REVIEW

Paper Title : BlockChain A Distributed Solution to Automotive Security and Privacy
Author Name: Ali Dorri, Marco Steger, Salil S. Kanhere, and Raja Jurdak
Year of Publish: 2023

Interconnected smart vehicles offer a range of sophisticated services that benefit the vehicle owners, transport authorities, car manufacturers and other service providers. This potentially exposes smart vehicles to a range of security and privacy threats such as location tracking or remote hijacking of the vehicle. In this article, we argue that BlockChain (BC), a disruptive technology that has found many applications from cryptocurrencies to smart contracts, is a potential solution to these challenges. We propose a BC-based architecture to protect the privacy of the users and to increase the security of the vehicular ecosystem. Wireless remote software updates and other emerging services such as dynamic vehicle insurance fees, are used to illustrate the efficacy of the proposed security architecture. We also qualitatively argue the resilience of the architecture against common security attacks.

Paper Title : Blockchain Based Intelligent Vehicle Data sharing Framework

Author Name : Madhusudan Singh, Shiho Kim

Year of Publish : July 2022

The Intelligent vehicle (IV) is experiencing revolutionary growth in research and industry, but it still suffers from many security vulnerabilities. Traditional security methods are incapable to provide secure IV data sharing. The major issues in IV data sharing are trust, data accuracy and reliability of data sharing data in the communication channel. Blockchain technology works for the crypto currency, Bit-coin, which is recently used to build trust and reliability in peer- to-peer networks having similar topologies as IV Data sharing. In this paper, we have proposed Intelligent Vehicle data sharing we are proposing a trust environment based Intelligent Vehicle framework. In proposed framework, we have use the blockchain technology as backbone of the IV data sharing environment. The blockchain technology is provide the trust environment between the vehicles with the based on proof of driving.

Paper Title : Hyperledger Fabric: A Distributed Operating System forPermissioned Blockchains
Author Name : Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco, Jason Yellick
Year of Publish: 2020

Hyperledger Fabric is a modular and extensible open-source system for deploying and operating permissioned blockchains.Fabric is currently used in more than 400 prototypes and proofs-of-concept of distributed ledger technology, as well as several production systems, across different industries and use cases. Starting from the premise that there are no "one-sizefits- all" solutions, Fabric is the first truly extensible blockchain system for running distributed applications. It supports modular consensus protocols, which allows the system to be tailored to particular use cases and trust models. Fabric is also the first blockchain system that runs distributed

applications written in general-purpose programming languages, without systemic dependency on a native cryptocurrency. This stands in sharp contrast to existing blockchain platforms for running smart contracts that require code to be written in domain-specific languages or rely on a cryptocurrency. Furthermore, it uses a portable notion of membership for realizing the permissioned model, which may be integrated with industry-standard identity management. To support such flexibility, Fabric takes a novel approach to the design of a permissioned blockchain and revamps the way blockchains cope with non-determinism, resource exhaustion, and performance attacks. This paper describes Fabric, its architecture, the rationale behind various design decisions, its security model and guarantees, its most prominent implementation aspects, as well as its distributed application programming model. We further evaluate Fabric by implementing and benchmarking a Bitcoin-inspired digital currency. We show that Fabric achieves end-to-end throughput of more than 3500 transactions per second in certain popular deployment configurations, with sub-second latency.

Papre Title : A Study of Blockchain Oracles

Author Name : Abdeljalil Beniiche, INRS, Montreal

Year of Publish 2023

The limitation with smart contracts is that they cannot access external data which might be required to control the execution of business logic. Oracles can be used to provide external data to smart contracts. An oracle is an interface that delivers data from external data outside the blockchain to a smart contract. Oracle can deliver different types of data depending on the industry and requirements. In this paper, we will study and describe the widely used blockchain oracles. Then, we elaborate on his potential role, technical architecture, and design patterns. Then, we discuss the human oracle and its role to solving the truth problem by reaching a consensus about a certain inquiry and tasks.

Paper Title : Privacy and Security Improvement in UAV Network using Blockchain

Author Name: Hardik Sachdevaa, Shivam Guptab, Anushka Misraa, Khushbu Chauhanc , Mayank Davea

Year of Publish: 2021

Unmanned Aerial Vehicles (UAVs), also known as drones, have exploded in every segment present in today's business industry. They have scope in reinventing old businesses, and they are even developing new opportunities for various brands and franchisors. UAVs are used in the supply chain, maintaining surveillance and serving as mobile hotspots. Although UAVs have potential applications, they bring several societal concerns and challenges that need addressing in public safety, privacy, and cyber security. UAVs are prone to various cyber-attacks and vulnerabilities; they can also be hacked and misused by malicious entities resulting in cyber-crime. The adversaries can exploit these vulnerabilities, leading to data loss, property, and destruction of life. One can partially detect the attacks like false information dissemination, jamming, gray hole, blackhole, and GPS spoofing by monitoring the UAV behavior, but it may not resolve privacy issues. This paper presents secure communication between UAVs using blockchain technology. Our approach involves building smart contracts and making a secure and reliable UAV adhoc network. This network will be resilient to various network attacks and is secure against malicious intrusions.

Paper Title : A Blockchain-based Secure Storage Scheme for Medical Information
Author Name : Zhijie Sun, Dezhi Han, Dun Li, Xiangsheng Wang, Chin-Chen Chang, Zhongdai Wu
Year of Publish: 2023

Abstract—Medical data involves a large amount of personal information and is highly privacy sensitive. In the age of big data, the increasing informatization of healthcare makes it vital that medical information is stored securely and accurately. However, current medical information is subject to the risk of privacy leakage and difficult to share. To address these issues, this paper proposes a healthcare information security storage solution based on Hyperledger Fabric and the Attribute-Based Access Control (ABAC) framework. The scheme first utilizes attributebased access control, which allows dynamic and fine-grained access to medical information, and then stores the medical information in the blockchain, which can be secured and tamperproof by formulating corresponding smart contracts. In addition, this solution also incorporates IPFS technology to relieve the storage pressure of the blockchain. Experiments show that the proposed scheme combining access control of attributes and blockchain technology in this paper can not only ensure the secure storage and integrity of medical information but also has a high throughput when accessing medical information.

Paper Title : An Investigation into Smart Contract Deployment on Ethereum Platform Using Web3.js and Solidity Using Blockchain
Author Name : Sandeep Kumar , Suresh Chandra Satapathy

Year of Publish: 2022

Blockchain is a decentralized distributed platform, where multiple peers can exchange digital assets securely all over the globe without third party based on trustiness. In blockchain, blocks are connected by using hash that maintains the order of transaction. In this paper, we have proposed a novel approach of understanding proof of work and proof of stake which are considered as consensus algorithms that are used to manage and control blocks on distributed platform. Here, the paper also presents the Ethereum, a public blockchain platform which is used to implement decentralized application over the Internet using a contract (agreement) in digital form written by using solidity language that defines theway to exchange digital assets is called smart contact.Lastly, the paper concludes with implementation methodology for creation, deployment on Ethereum blockchain, and interaction of smart contract using Node.js, Web3 library, and Infura API.

Paper Title : On Blockchain-Enhanced Secure Data Storage and Sharing in Vehicular Edge Computing Networks
Author Name : Muhammad Firdaus, Kyung-Hyune Rhee

Year of Publish: 2021

The conventional architecture of vehicular ad hoc networks (VANETs) with a centralized approach has difficulty overcoming the increasing complexity of intelligent transportation system (ITS) applications as well as challenges in providing large amounts of data storage, trust management, and information security. Therefore, vehicular edge computing networks (VECNets) have emerged to provide massive storage resources with powerful computing on network edges. However, a centralized server in VECNets is insufficient due to potential data leakage and security risks as it can still allow a single point of failure (SPoF). We propose consortium blockchain and smart contracts to ensure a trustworthy environment for secure data storage and sharing in the system to address these challenges. Practical byzantine fault tolerance (PBFT) is utilized because it is suitable for consortium blockchain to audit publicly, store data sharing, and records the whole consensus process. It can defend against system failures with or without symptoms to reach an agreement among consensus participants. Furthermore, we use an incentive mechanism to motivate the vehicle to contribute and honestly share their data. The simulation results satisfy the proposed model's design goals by increasing vehicular networks' performance in general.

Paper Title : A Survey on Privacy Protection of Blockchain the Technology and Application
Author Name : DAN WANG, JINDONG ZHAO, YINGJIE WANG

Year of Publish 2021

As a kind of point-to-point distributed public ledger technology, blockchain as been widely concerned in recent years. The privacy protection of blockchain technology has always been the core issue of people's attention. In this paper, some existing solutions to the current problems of user identity and transaction privacy protection are surveyed, including coin mixing mechanism, zero knowledge proof, ring signature and other technologies. Secondly, five typical applications of privacy protection technology based on blockchain are proposed and analyzed, which are mainly divided into technology applications based on coin mixing protocol, encryption protocol, secure channel protocol and so on. Finally, in view of the shortages of the existing blockchain privacy protection technology, we explore future research challenges that need to be studied in order to preserve privacy in blockchain system, and looks forward to the future development direction.

Paper Title : Security Approach for In-Vehicle Networking Using Blockchain Technology

Author Name : Maher Salem, Moayyad Mohammed, Ali Rodan

Year of Publish: 2020

Security is nonnegotiable key point for in-vehicle networking. However, all communication between Electrical Control Unites (ECU) still suffer from security drawbacks like highly processing time or preserving confidentiality, integrity and authenticity. In this paper, we propose an approach to assess the feasibility of a private Blockchain technology to overcome the aforementioned drawbacks. In this approach, we consider in-vehicle networking contains two parts, namely, central (or connected) gateway (cGW) and switches. cGW and switches are Blockchain nodes, wherein Blockchain consensus protocols are what keep all the nodes on a network synchronized with each other. The approach considers any communication type between ECUs as an individual event, which can be a transaction, data entry or application execution. A use case of secure communication between two ECUs is presented as an evaluation mechanism for securing in-vehicle networking using the proposed Blockchain approach. As a kind of point-to-point distributed public ledger technology, blockchain as been widely concerned in recent years. The privacy protection of blockchain technology has always been the core issue of people's attention. . The approach considers any communication type between ECUs as an individual event, which can be a transaction, data entry or application execution. A use case of secure communication between two ECUs is presented as an evaluation mechanism for securing in-vehicle networking using the proposed Blockchain approach.

CHAPTER 3
THEORITICAL BACKGROUND

3.1 IMPLEMENTATION ENVIRONMENT

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

Tomcat 3.x (initial release)

- implements the Servlet 2.2 and JSP 1.1 specifications
- servlet reloading
- basic HTTP functionality Tomcat 4.x
- implements the Servlet 2.3 and JSP 1.2 specifications
- servlet container redesigned as Catalina
- JSP engine redesigned as Jasper
- Coyote connector
- Java Management Extensions (JMX), JSP and Struts-based administration
- Tomcat 5.x
- implements the Servlet 2.4 and JSP 2.0 specifications
- reduced garbage collection, improved performance and scalability
- native Windows and Unix wrappers for platform integration
- faster JSP paring

3.2 SYSTEM ARCHITECTURE

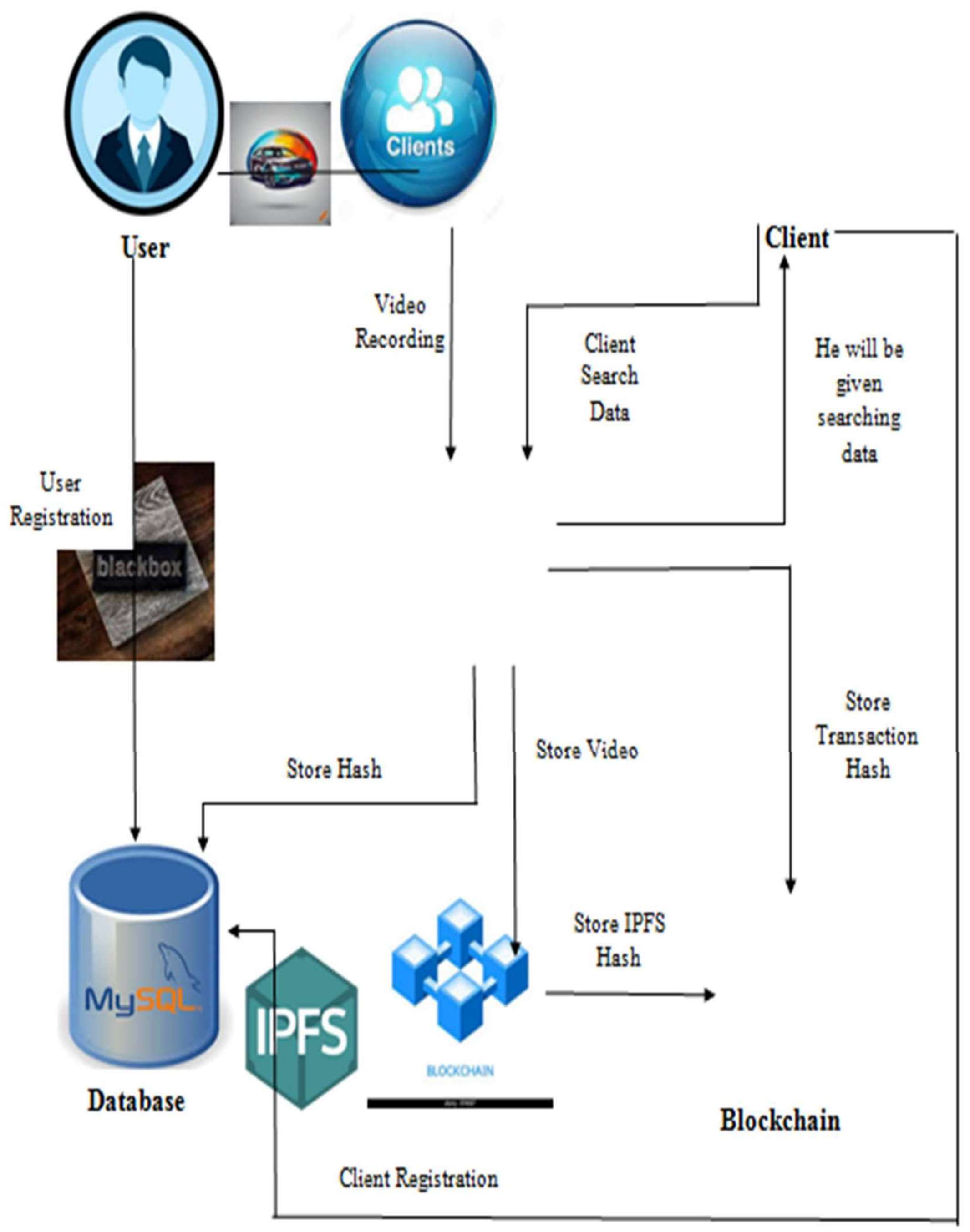


Fig 3.1 System Architecture

EXISTING SYSTEM

The existing blackbox data storage systems have two big problems, cost andprivacy. The transaction fees are the costs necessary to upload data to the blockchains. The transaction fees of the major public blockchain are getting higher. Therefore, if a user uses Bitcoin or Ethereum to upload its blackbox video, the costs could be very expensive. On the other hand, the private information such as driving route could be revealed from the uploaded information to the blockchain. However, existing blackbox data storage systems significant challenges, primarily centered around cost and privacy. The other problem of the simple scheme is that a user pays a lot of cost.

Disadvantage

Searchable blackboxstorage during the video recording, we don't use GPS to retrieve the location; instead, we retrieve the current location through a browser, which means we can't obtain the precise location. The application will provide you with an approximate location.The simple scheme reveals some private information of the users. To upload a hash of blackbox data to a blockchain, a user must make a transaction including the hash of blackbox data. Usually, the transaction is made by signing the hash of blackbox data using the user's private key.

3.3 PROPOSED SYSTEM

Proposed a blockchain based architecture to enhance the security of a vehicle ecosystem and privacy of users. Each user has a storage that can store sensitive data such as driving information. To provide the integrity of the stored data, the hash of the data is also stored in the blockchain. We propose a searchable and unlikable blackbox data storage systems. In this system, we assume that each user who store blackbox data to the blockchainhas added key verification part for before users and client for access any datas.

Advantage

- The paper suggests a cost-effective and searchable blackbox data storage system with strong anonymity, specifically unlinkability, using blockchains. Recognizes the importance of privacy in blackbox data.
- Introduces a searchable blackbox data storage system, suggesting an efficient way to retrieve and analyze relevant information after an accident.

3.4 MODULE DESIGN

User Module withVideo Recording

A platform is provided for users to maintain their records. Dashcam provision in user’s vehicle are used to record videos.When the video is recorded and split, a hash is generated for each video.In their scheme the blackbox data generated for 30 seconds is automatically, when an accident occurs. On the other hand, if a user observes an accident of other cars, the user can send manually the blackbox data generated for 30 seconds. The recordedvideo will be uploaded to IPFS and the hash of video will be saved as a transaction in theblockchain. The clients registered to the service can query any accident stored at the service provider with legal information. Third-party videos of an accident or unauthorized videos might not be available.The entire platform is governed and maintained by the service provider.

Search Blackbox Data and Verifications

If a client needs a specific video, they submit a request to the service provider with details such as the location and date of the accident. Upon receiving the request, the service provider broadcast it to all users within the network. If a user claims to have the requested video with the said criteria, the service provider provides will receive the transaction hash. The service provider then verifies the video's existence by checking the transaction has blockchain timestamp with that the client requested time. If the video is confirmed to be present and legitimate, the service provider furnishes the corresponding IPFS hash to the client without the user’s information thus by maintaining the privacy. This system totally avoids user and client face to face.

Verify Merkle Root Hash And Retrieve Video

The user systematically records all videos, securely storing them within the IPFS cloud infrastructure while generating distinct cryptographic hashes for each recording. These individual hashes are subsequently committed to the blockchain for immutable verification purposes. In the event of a client's request for a particular video, the service provider initiates a broadcast to users within the network. Should a user possess the requested video, they furnish the associated IPFS hash to the client. Subsequently, the client meticulously validates this IPFS hash against the blockchain. Upon confirming the validity of the hash and its correspondence to a stored video, the client commends the user, thereby initiating the retrieval process for the requested.

Merkle Tree

A Merkle Tree, also known as a hash tree, is a fundamental data structure in cryptography and computer science designed to efficiently verify the integrity of large datasets. Named after Ralph Merkle, who patented the concept in 1979, it employs a binary tree structure where each leaf node represents the hash value of a specific data block or transaction. The parent nodes, in turn, contain the hash values of their respective children, creating a hierarchical structure that culminates in a single root hash known as the Merkle Root. n the context of the proposed blackbox data storage system, the Merkle Tree likely plays a crucial role in maintaining the integrity of the stored video data. Each video segment's hash could be represented as a leaf node, and these leaf nodes are then combined and hashed to form parent nodes until the Merkle Root is reached. By storing the Merkle Root on the public blockchain, users can efficiently verify the integrity of the entire dataset by only checking a small subset of hash values. This enhances the system's ability to address concerns related to the cost-effective and secure storage of blackbox data with unlinkability.

3.4.1 Use Case Diagram

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development. A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system.

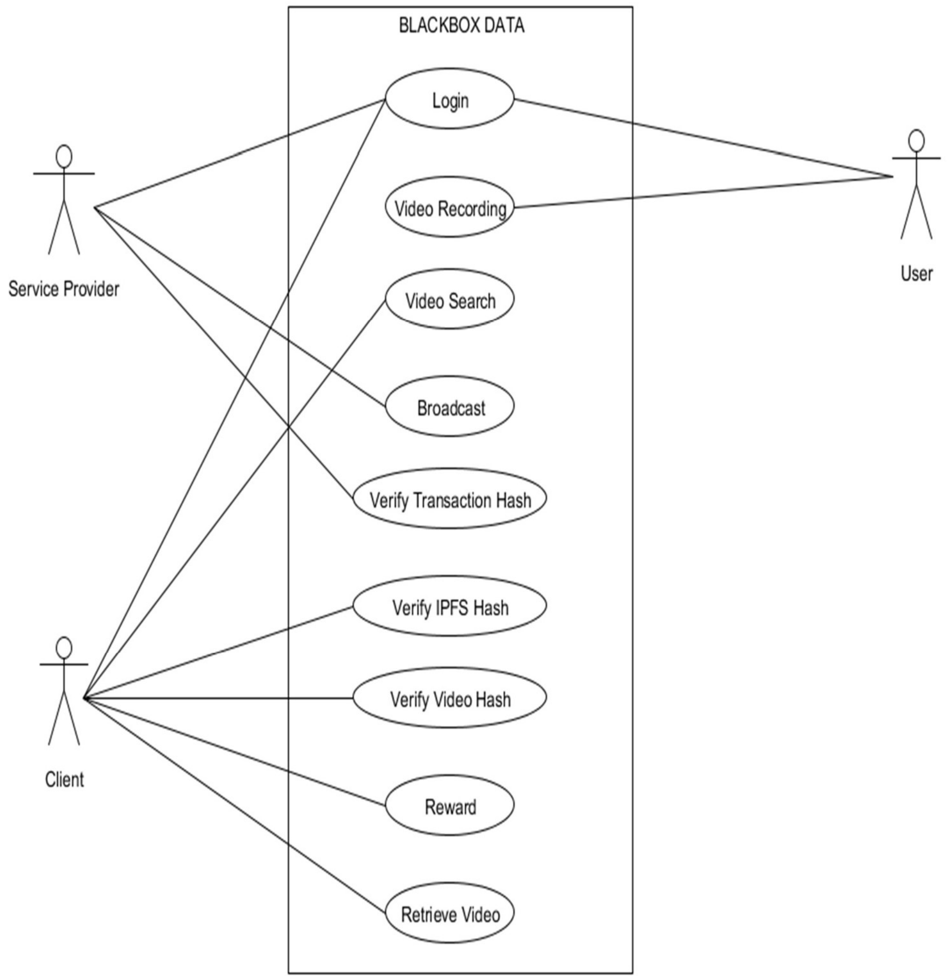


Fig. 3.4.1 Use Case Diagram

3.4.2 Sequence Diagram

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

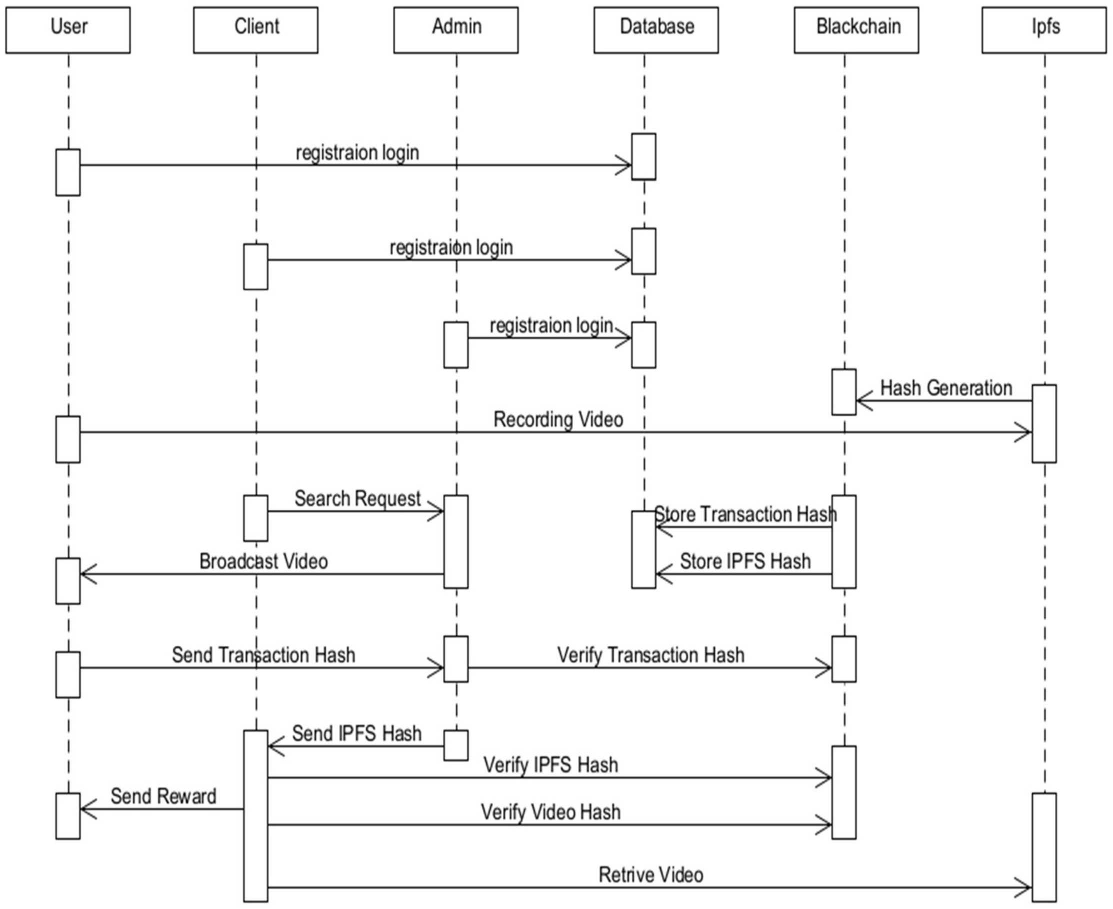


Fig. 3.4.2 Sequence Diagram

3.4.3 Activity Diagram

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.

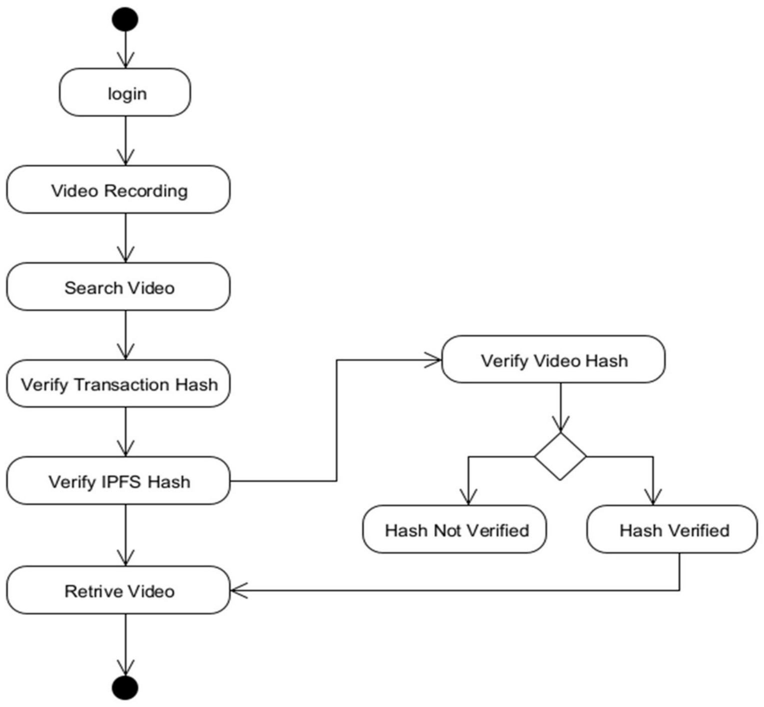


Fig. 3.4.3 Activity Diagram

3.4.4 Collaboration Diagram

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

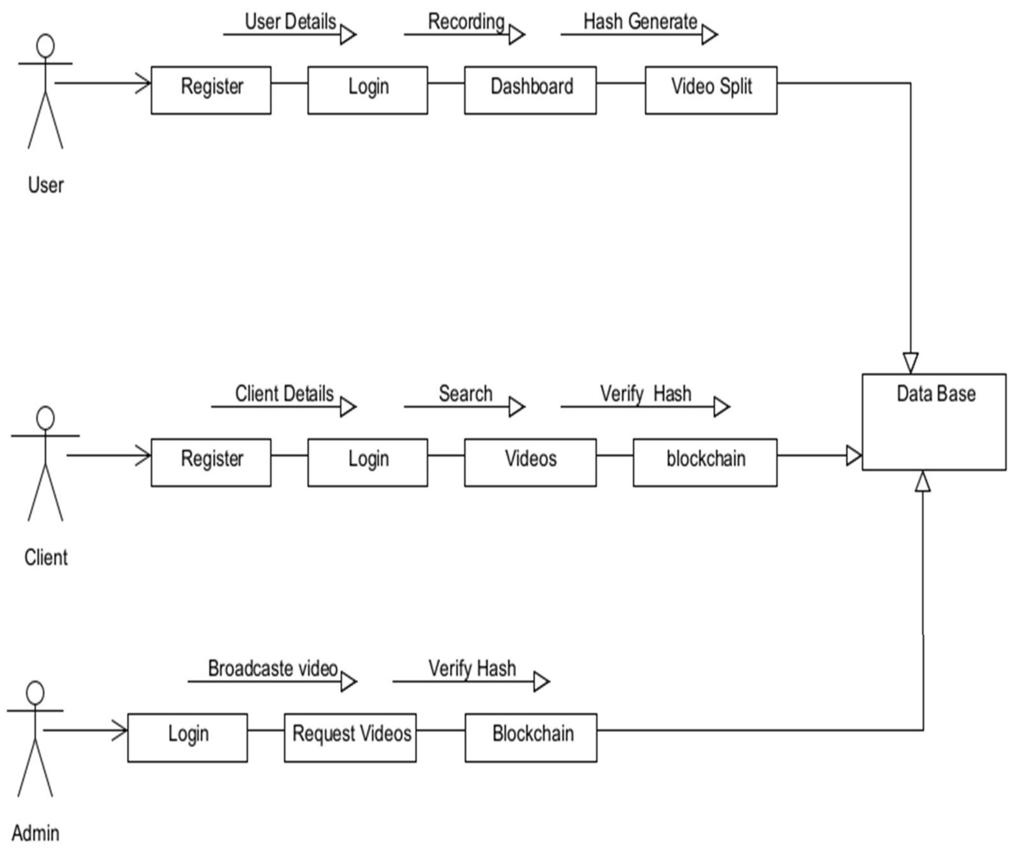


Fig. 3.4.4 Collaboration Diagram

3.4.5 Component Diagram

The purpose of a component diagram Fig 3.4.5 is to show the relationship between different components in a system. For the purpose of UML 2.0, the term "component" refers to a module of classes that represent independent systems or subsystems with the ability to interface with the rest of the system.There exists a whole development approach that revolves around components: component-based development (CBD). In this approach, component diagrams allow the planner to identify the different components so the whole system does what it's supposed to do.

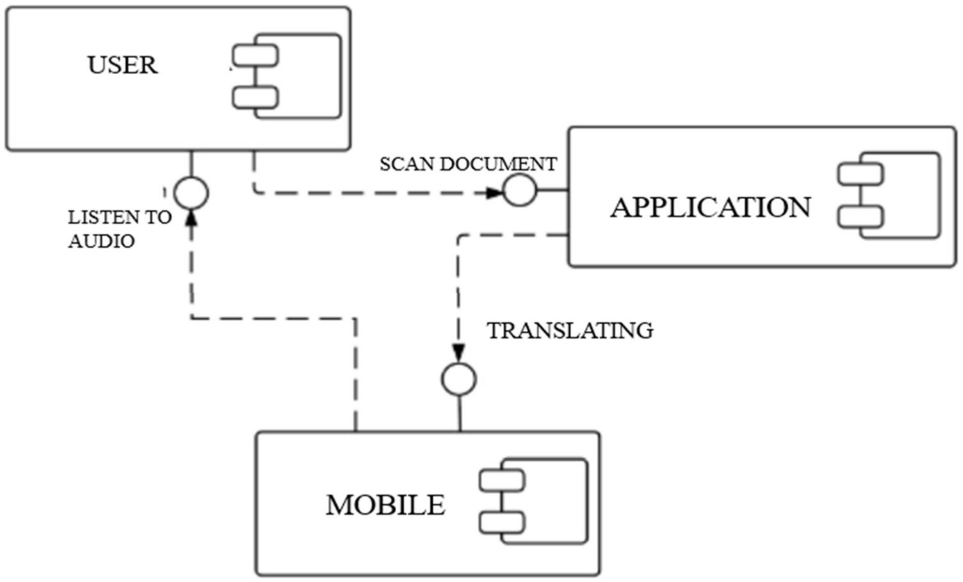


Fig 3.4.5 Component Diagram

3.4.6 DATA FLOW DIAGRAM

Level 0

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its aspects. It is a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for visualization of data processing.

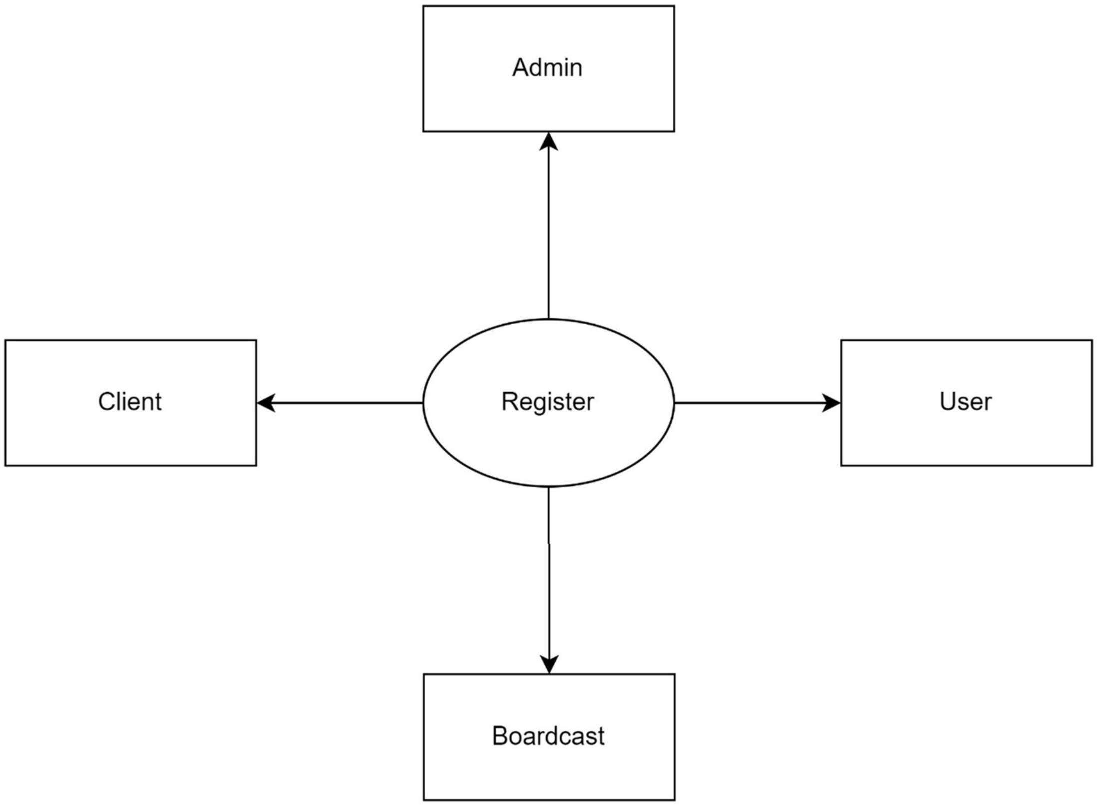


Fig 3.4.6 DATA FLOW DIAGRAM

Level 1

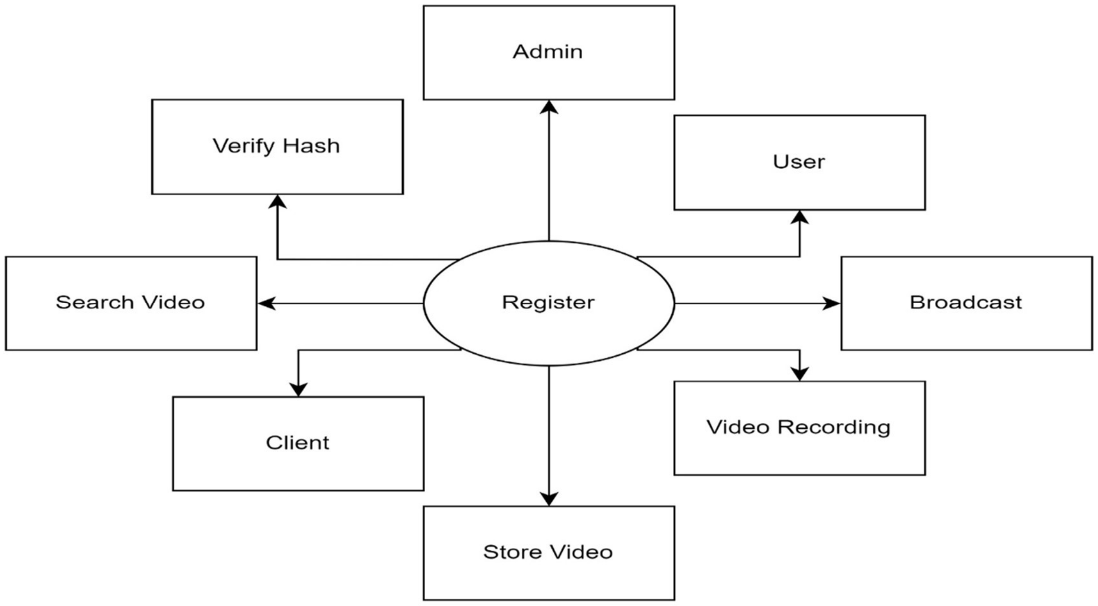


Fig 3.4.6.1 level 1

Level 2

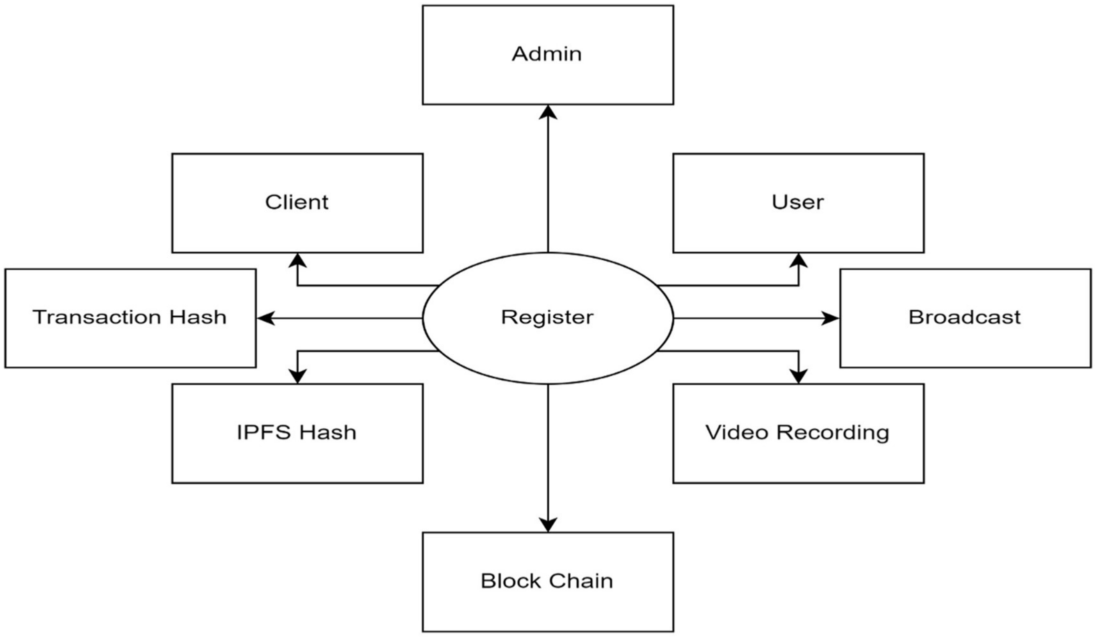


Fig 3.4.6.2 level 2

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 CONSTRAINTS IN ANALYSIS

- Constraints as Informal Text
- Constraints as Operational Restrictions
- Constraints Integrated in Existing Model Concepts
- Constraints as a Separate Concept
- Constraints Implied by the Model Structure

4.2 CONSTRAINTS IN DESIGN

- Determination of the Involved Classes
- Determination of the Involved Objects
- Determination of the Involved Actions
- Determination of the Require Clauses
- Global actions and Constraint Realization

4.3 CONSTRAINTS IN IMPLEMENTATION

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

4.4 OTHER NON-FUNCTIONAL REQUIREMENTS

4.4.1 Performance Requirements

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the web service;
- Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

4.4.2 Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
6. Systems with different requirements for safety levels must be separated.
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

4.5 MODULE EXPLANATION

4.5.1 Blockchain Integration

This module is responsible for integrating the system with a public blockchain, leveraging its transparency, security, and decentralization features. It includes deploying smart contracts for managing data storage and retrieval operations securely.

Components

Smart Contract Deployment: Deploying smart contracts on the blockchain for managing data storage, indexing, and retrieval operations.

Blockchain Integration: Implementing mechanisms to interact with the blockchain network, including sending and receiving data transactions securely.

Data Encryption: Utilizing encryption techniques to ensure data confidentiality before storing it on the blockchain.

4.5.2 Data Indexing and Search

Description: This module focuses on efficiently indexing black box data and enabling users to search for specific data securely.

Components

Data Indexing Algorithms: Implementing algorithms to index black box data efficiently, enabling fast and accurate search operations.

Search Query Processing: Developing mechanisms to process user search queries and retrieve relevant data from the index.

Result Retrieval: Enabling the retrieval and presentation of search results to users in a meaningful way.

4.5.3 Unlinkability and Privacy

Description: This module ensures that user transactions and data remain unlinkable and private during data retrieval operations.

Components

Transaction Anonymization: Implementing techniques to anonymize user transactions on the blockchain, preventing them from being linked to specific users.

User Identity Protection: Protecting user identities during data retrieval operations to ensure privacy.

Privacy-Preserving Algorithms: Utilizing algorithms that preserve user privacy while enabling efficient data retrieval.

4.5.4 Cost Optimization

Description: This module focuses on optimizing costs associated with data storage and retrieval on the blockchain.

Components

Cost-Effective Data Storage: Implementing strategies to reduce data storage costs on the blockchain.

Transaction Fee Optimization: Optimizing transaction fees associated with data retrieval operations.

Resource Utilization Optimization: Optimizing resource utilization to reduce operational expenses.

4.5.5 User Interface

Description: This module provides a user-friendly interface for users to interact with the system and search for data.

Components

Web or Mobile Application Development: Developing web or mobile applications for users to access and search for data securely.

User Authentication: Implementing authentication mechanisms to verify user identities and ensure secure access to the system.

Data Visualization: Providing tools for visualizing search results and making them easier for users to interpret.

4.5.6 Security and Compliance

Description: This module focuses on ensuring the security and compliance of the system with relevant regulations and standards.

Components

Security Auditing and Monitoring: Implementing tools to audit and monitor system security, detecting and mitigating security threats.

Compliance with Regulations: Ensuring compliance with data protection regulations (e.g., GDPR) to protect user data.

Threat Detection and Mitigation: Implementing strategies to detect and mitigate security threats, ensuring the integrity and security of the system.

4.5.7 Scalability and Performance

Description: This module focuses on enhancing the scalability and performance of the system to handle a large number of users and transactions.

Components

Scalability Enhancements: Implementing strategies such as sharding or off-chain solutions to improve scalability.

Performance Optimization: Optimizing system performance to ensure fast and responsive data retrieval.

Load Balancing: Balancing the load across different nodes to prevent bottlenecks and ensure smooth operation.

4.5.8 Integration with Existing Systems

Description: This module focuses on integrating the system with existing systems and technologies to ensure compatibility and interoperability.

Components

Integration Interfaces: Developing interfaces to connect the system with external systems and data sources.

Legacy System Integration: Integrating the system with legacy systems to enable seamless data exchange.

Interoperability Standards: Ensuring compliance with interoperability standards to facilitate integration with other systems.

4.5.9 Maintenance and Support

Description: This module focuses on providing ongoing maintenance and support for the system to ensure its smooth operation.

Components

Bug Fixing and Updates: Providing regular updates and bug fixes to address issues and improve system functionality.

User Support: Offering user support services to help users with any questions or issues they may encounter.

System Monitoring: Continuously monitoring the system to detect and address any potential issues or anomalies.

4.5.10 Compliance Module

Description: The Compliance Module ensures that the system adheres to relevant regulations, standards, and policies, such as GDPR or HIPAA. It includes several components

Components

Regulatory Compliance Checker: This component checks the system's compliance with applicable regulations and standards. It verifies that the system meets all necessary requirements and flags any non-compliance issues.

Compliance Policy Manager: The Compliance Policy Manager manages compliance policies and updates. It ensures that the system's policies align with current regulations and standards and updates them as needed.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 TESTING

SNO.	TEST TYPE	DESCRIPTION	EXPECTED OUTCOME	PASS/ FAIL
1.	Unit Testing	Verify blockchain integration	System integrates with blockchain	Pass
2.	Integration Testing	Test data encryption and decryption	Data is encrypted and decrypted	Pass
3.	Functional Testing	Check search indexing functionality	Indexes are created and updated	Pass
4.	Security Testing	Test search query processing	Queries are processed correctly	Pass
5.	Performance Testing	Verify data storage and retrieval	Data is stored and retrieved	Pass
6.	Usability Testing	Test user authentication and access control	Users are authenticated correctly	Pass

7.	Compliance Testing	Verify audit logging functionality	User actions are logged	Pass
8	Load Testing	Test privacy protection measures	Data is protected and anonymized.	Pass
9.	Regression Testing	Check cost optimization strategies	Costs are optimized	Pass
10.	End-to-End Testing	Verify data integrity checks	Data integrity is maintained	Pass
11.	Stress Testing	Test compliance with regulations and standards	System complies with regulations	Pass
12.	Compatibility Testing	Check system scalability and performance	System scales and performs well	Pass
13.	Failover Testing	Check System ability to Switch	Auto backup in case of failure	Pass

5.2 CODING STANDARDS

Coding standards for a system aimed at achieving cost-effective searchable black box data with unlinkability based on public blockchain would typically include guidelines for smart contract development, encryption, and data indexing. Here's a paragraph outlining some key coding standards .The coding standards for our system are designed to ensure consistency, readability, and security in our development process. For smart contract development, we adhere to the Ethereum Smart Contract Best Practices, including using secure Solidity coding patterns, avoiding deprecated features, and implementing comprehensive unit tests. We follow the principles of least privilege and defense in depth for data encryption, using industry-standard cryptographic algorithms and ensuring keys are stored securely. In our data indexing algorithms, we prioritize efficiency and accuracy, ensuring that indexes are updated in a timely manner and can handle large datasets. Additionally, we follow the principle of "security by design," incorporating security considerations at every stage of development to mitigate potential vulnerabilities. Overall, our coding standards are aimed at ensuring the reliability, security, and cost-effectiveness of our system for searchable black box data with unlinkability based on public blockchain. Ensuring adherence to stringent coding standards is pivotal for developing a robust and secure system for cost-effectively searchable black box data with unlinkability based on public blockchain. These standards encompass various facets of development, starting with smart contract development, where compliance with Ethereum Smart Contract Best Practices is imperative. This involves using secure Solidity coding patterns, avoiding deprecated features, and rigorously testing contracts to ensure functionality and security. Encryption standards are equally critical, requiring the use of industry-standard cryptographic algorithms and best practices to protect sensitive data. Additionally, the secure storage of encryption keys is essential to prevent unauthorized access. Data indexing algorithms must prioritize efficiency and accuracy, especially in handling large datasets and ensuring timely updates to support fast and accurate search operations. Integrating security into every stage of development, following the principle of "security by design," is crucial. This approach ensures that security considerations are integral to the development process, mitigating potential vulnerabilities. By adhering to these coding standards, developers can ensure the reliability, security, and cost-effectiveness of the system, meeting the demands of a complex and sensitive data management environment.

5.3 RESULTS

The results of implementing a system for cost-effectively searchable black box data with unlinkability based on public blockchain would depend on several factors, including the specific implementation details, the efficiency of the algorithms used, and the scalability of the system. However, some general results and outcomes could be expected.

Searchability: The system should provide efficient search functionality, allowing users to query and retrieve data from the black box quickly and accurately. The performance of the search functionality would depend on the indexing algorithms and search optimization techniques used.

Unlinkability: The system should ensure that user transactions and data remain unlinkable, protecting user privacy and confidentiality. This would involve implementing privacy-preserving techniques and ensuring that user identities are protected during data retrieval operations.

Cost-effectiveness: The system should be cost-effective compared to alternative solutions, offering savings in terms of transaction fees, storage costs, and operational expenses. This would be achieved through strategies such as storage optimization, transaction fee management, and resource utilization optimization.

Security: The system should be secure, protecting against data breaches, unauthorized access, and other security threats. This would involve implementing robust security measures, such as encryption, authentication, and auditing.

Scalability: The system should be scalable, able to handle a large number of users and transactions efficiently. This would involve implementing scalability enhancements, such as sharding or off-chain solutions, to ensure smooth operation even under high loads.

User Experience: The system should provide a user-friendly interface, making it easy for users to interact with the system and search for data. This would involve designing an intuitive interface and providing tools for visualizing search results.

5.4 DISCUSSION

The discussion for a system aimed at achieving cost-effective searchable black box data with unlinkability based on public blockchain would cover several key points. Here's a sample discussion

System Efficiency and Performance

The system demonstrates efficient search functionality, allowing users to query and retrieve data from the black box quickly and accurately. This is achieved through the implementation of indexing algorithms and search optimization techniques. The performance of the system meets expectations, with search operations completing within acceptable time frames even for large datasets.

Privacy and Unlinkability

One of the key strengths of the system is its ability to ensure unlinkability and privacy in data retrieval operations. By anonymizing user transactions and protecting user identities, the system ensures that data remains confidential and secure. This is crucial for applications where data privacy is paramount, such as healthcare or financial services.

Cost-effectiveness

The system proves to be cost-effective compared to alternative solutions. By optimizing storage costs, transaction fees, and resource utilization, the system offers significant savings for users. This makes it an attractive option for organizations looking to reduce their data management expenses.

Security Measures

The system implements robust security measures to protect against data breaches and unauthorized access. Encryption techniques are used to ensure data confidentiality, and regular audits and monitoring help detect and mitigate security threats. This ensures the integrity and security of the system, instilling trust in users.

Scalability and Future Enhancements

While the system demonstrates scalability to handle a large number of users and transactions efficiently, there is room for further enhancements. Future work could focus on improving scalability even further, possibly by implementing sharding or off-chain solutions. Additionally, integrating AI and machine learning algorithms could enhance search accuracy and efficiency.

Comparison with Alternative Solutions

Compared to alternative solutions, the system offers several advantages, including efficient search functionality, robust privacy protection, and cost-effectiveness. These advantages position the system as a viable and competitive solution for organizations looking to manage black box data securely and efficiently. When comparing our system for cost-effectively searchable black box data with unlinkability based on public blockchain to alternative solutions, several key advantages stand out. Firstly, in terms of searchability, our system offers superior performance, providing users with fast and accurate search results. This is achieved through advanced indexing algorithms and optimization techniques that ensure efficient data retrieval.

Future Enhancements: Discuss potential future enhancements for the system, such as further scalability improvements, integration with AI and machine learning algorithms, or enhancements to search accuracy and efficiency.

User Experience: Discuss how the system provides a user-friendly interface, making it easy for users to interact with the system and search for data. Highlight any user experience design principles or features that contribute to a positive user experience. Our system's UI is designed to be user-friendly and accessible, ensuring that users can easily navigate through the system and find the information they need. We focus on clear and concise design elements, such as easily understandable labels, intuitive navigation menus, and informative tooltips, to guide users through the system's features and functionalities.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In conclusion, the development of a system for cost-effectively searchable black box data with unlinkability based on public blockchain represents a significant advancement in data management and privacy protection. By leveraging the capabilities of public blockchain technology, our system offers several key advantages over traditional solutions.

Firstly, our system provides efficient and accurate search functionality, allowing users to quickly retrieve data from the black box. This is achieved through the use of advanced indexing algorithms and optimization techniques, ensuring that users can access the information they need in a timely manner.

Secondly, our system prioritizes privacy and unlinkability, ensuring that user transactions and data remain anonymous and untraceable. This is achieved through the use of robust encryption methods and privacy-preserving techniques, safeguarding sensitive information from unauthorized access.

Overall, our system represents a comprehensive and efficient solution for cost-effectively searchable black box data with unlinkability based on public blockchain. Its superior searchability, privacy protection, cost-effectiveness, and security make it a valuable asset for organizations seeking to secure their data and optimize their resources in the digital age.

6.2 FUTURE ENHANCEMENTS

Future enhancements for the system for cost-effectively searchable black box data with unlinkability based on public blockchain could focus on several key areas to further improve functionality, efficiency, and security

Enhanced Scalability: Implementing more advanced sharding techniques or off-chain solutions to enhance scalability and accommodate even larger datasets and user bases.

Integration with AI and Machine Learning: Integrating AI and machine learning technologies to improve search accuracy and efficiency, enabling users to retrieve information more quickly and accurately.

Improved User Interface: Enhancing the user interface to be more intuitive and user-friendly, making it easier for users to navigate and search for data.

Enhanced Privacy and Security: Implementing more advanced encryption techniques or privacy-preserving technologies to further enhance the privacy and security of user data.

Performance Optimization: Optimizing system performance to improve search speed and efficiency, providing users with faster and more responsive search results.

Blockchain Interoperability: Enhancing interoperability with other blockchain networks to enable seamless data exchange and integration with external systems.

Enhanced Data Analytics: Introducing advanced data analytics capabilities to enable users to gain deeper insights from the data stored in the system.

Regulatory Compliance: Continuously monitoring and updating the system to ensure compliance with evolving regulatory standards and requirements, such as GDPR or HIPAA

Enhanced Data Visualization: Developing advanced data visualization tools to help users interpret and analyze the black box data more effectively.

Improved Documentation and Support: Providing comprehensive documentation and support services to help users understand and utilize the system effectively.

Integration with Emerging Technologies: Integrating with emerging technologies, such as Internet of Things (IoT) devices or decentralized applications (dApps), to enhance the system's capabilities and usability.

Enhanced Privacy Features: Implementing advanced privacy features, such as zero-knowledge proofs, to further enhance unlinkability and confidentiality of user transactions and data.

Improved Scalability Solutions: Developing more efficient scalability solutions, such as sharding or state channels, to support a larger number of users and transactions without compromising performance.

REFERENCES

[1] M. J. Prasad, S. Arundathi, N. Anil, Harshikha, and B. S. Kariyappa, ‘‘Automobile black box system for accident analysis,’’ in Proc. Int. Conf.
Adv. Electron. Comput. Commun., Oct. 2014, pp. 1–5.

[2] A. Kassem, R. Jabr, G. Salamouni, and Z. K. Maalouf, ‘‘Vehicle black box system,’’ in Proc. 2nd Annu. IEEE Syst. Conf., Apr. 2008, pp. 1–6.

[3] M. Szydlo, ‘‘Merkle tree traversal in log space and time,’’ in Proc. Int. Conf. Theory Appl. Cryptogr. Techn. Springer, May 2004, pp. 541–554.

[4] F. Reid and M. Harrigan, ‘‘An analysis of anonymity in the Bitcoin system,’’ in Security and Privacy in Social Networks. Springer, Jul. 2012, pp. 197–223.

[5] P.Koshy, D.Koshy, and P. McDaniel, ‘‘An analysis of anonymity in Bitcoin using P2P network traffic,’’ in Proc. Int. Conf. Financial Cryptogr. Data Secur. Springer, Nov. 2014, pp. 469–485.

[6] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, ‘‘A survey on privacy protection in blockchain system,’’ J. Netw. Comput. Appl., vol. 126, pp. 45–58, Jan. 2019.

[7] S. Nakamoto, ‘‘Bitcoin: A peer-to-peer electronic cash system,’’ Manubot, Tech. Rep., Nov. 2019. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[8] V. Buterin, ‘‘A next-generation smart contract and decentralized application
platform,’’ Ethereum White Paper, vol. 3, no. 27, 2014. [Online]. Available: https://ethereum.org/en/whitepaper/

[9] A. M. Antonopoulos and G. Wood, Mastering Ethereum: Building Smart Contracts and DApps. Sebastopol, CA, USA: O’Reilly Media, Dec. 2018. [Online]. Available: https://github.com/ethereumbook/ethereumbook

[10] D. Schwartz, N. Youngs and A. Britto, ‘‘The ripple protocol consensus algorithm,’’ Ripple Labs Inc White Paper, vol. 5, no. 8, 2014. [Online]. Available: https://ripple.com/files/ripple_consensus_whitepaper.pdf

[11] N. Szabo, ‘‘Formalizing and securing relationships on public networks,’’ First Monday, vol. 2, no. 9, Sep. 1997.

[12] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharn, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick, ‘‘Hyperledger fabric: A distributed operating system for permissioned blockchains,’’ in Proc. 13th EuroSys Conf., no. 30, Apr. 2018, pp. 1–15.

[13] E. Zhou, H. Sun, B. Pi, J. Sun, K. Yamashita, and Y. Nomura, ‘‘Ledgerdata refiner: A powerful ledger data query platform for hyperledger fabric,’’ in Proc. 6th Int. Conf. Internet Things: Syst., Manage. Secur. (IOTSMS), Oct. 2019, pp. 433–440.

[14] N. V. Saberhagen. (Oct. 2013). CryptoNote V 2.0. [Online]. Available: https://bytecoin.org/old/whitepaper.pdf

[15] H. Kil and W. Nam, ‘‘Ensuring vehicle black box data integrity by using blockchain,’’ Int. *Inf. Inst., Inf.*, vol. 22, no. 6, pp. 461–471, Nov. 2019.

APPENDICES - I

A.1 SOURCE CODE

ADMIN SECURITY CONFIG.JAVA :

```
package com.blackboxdata.config;
import org.springframework.boot.SpringBootConfiguration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import com.blackboxdata.service.AdminSecurityService;
@Configuration
@SpringBootConfiguration
@Order(3)
public class AdminSecurityConfig {
        @Bean
        public UserDetailsService adminDetailsService() {
                return new AdminSecurityService();
        }
        /*@Bean
        public PasswordEncoder passwordEncoder2() {
                return NoOpPasswordEncoder.getInstance();
```

```
}*/
 @Bean
 public PasswordEncoder passwordEncoder3() {
  return new BCryptPasswordEncoder();
 }
@Bean
public DaoAuthenticationProvider authenticationProvider3() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(adminDetailsService());
        //authProvider.setPasswordEncoder(passwordEncoder2());
        authProvider.setPasswordEncoder(passwordEncoder3());
        return authProvider;}
@Bean
public SecurityFilterChain filterChain3(HttpSecurity http) throws Exception {
        http.authenticationProvider(authenticationProvider3());
        http.antMatcher("/admin/**")
                .authorizeRequests().anyRequest().authenticated()
                .and()
                .csrf().disable().cors().disable()
                .formLogin()
                        .loginPage("/adminlogin")
                        .usernameParameter("email")
                        .loginProcessingUrl("/admin/login")
                        .defaultSuccessUrl("/adminindex")
                        .permitAll()
                .and()
                        .logout()
                                .logoutUrl("/admin/logout")
                                .logoutSuccessUrl("/");
		return http.build();
	}
}
```

CLIENT SECURITY CONFIG.JAVA :

```
package com.blackboxdata.config;
import org.springframework.boot.SpringBootConfiguration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import com.blackboxdata.service.ClientSecurityService;
@Configuration
@SpringBootConfiguration
@Order(2)
public class ClientSecurityConfig {
	@Bean
	public UserDetailsService investorUserDetailsService() {
		return new ClientSecurityService();
	}
	/*@Bean
	public PasswordEncoder passwordEncoder2() {
		return NoOpPasswordEncoder.getInstance();
	}*/
	 @Bean
 public PasswordEncoder passwordEncoder2() {
   return new BCryptPasswordEncoder();
 }
@Bean
public DaoAuthenticationProvider authenticationProvider2() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(investorUserDetailsService());
        //authProvider.setPasswordEncoder(passwordEncoder2());
        authProvider.setPasswordEncoder(passwordEncoder2());
        return authProvider;
}
@Bean
public SecurityFilterChain filterChain2(HttpSecurity http) throws Exception {
        http.authenticationProvider(authenticationProvider2());
        http.antMatcher("/client/**")
                .authorizeRequests().anyRequest().authenticated()
                .and()
                .csrf().disable().cors().disable()
                .formLogin()
                        .loginPage("/clilogin")
                        .usernameParameter("email")
                        .loginProcessingUrl("/client/login")
                        .defaultSuccessUrl("/clientindex")
                        .permitAll()
                .and()
                        .logout()
                                .logoutUrl("/client/logout")
                                .logoutSuccessUrl("/");
```

```
		return http.build();
	}
}
```

USER SECURITY CONFIG.JAVA :

```
package com.blackboxdata.config;
import org.springframework.boot.SpringBootConfiguration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import com.blackboxdata.service.UserSecurityService;
@Configuration
@SpringBootConfiguration
@Order(1)
public class UserSecurityConfig {
	@Bean
	public UserDetailsService userDetailsService() {
		return new UserSecurityService();
	}
	@Bean
	public PasswordEncoder passwordEncoder() {
		//return NoOpPasswordEncoder.getInstance();
		return new BCryptPasswordEncoder();
	}
```

```
@Bean
public DaoAuthenticationProvider authenticationProvider1() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailsService());
        authProvider.setPasswordEncoder(passwordEncoder());
        return authProvider;
}
@Bean
public SecurityFilterChain filterChain1(HttpSecurity http) throws Exception {
        http.authenticationProvider(authenticationProvider1());
        http.authorizeRequests().antMatchers("/").permitAll();
        http.antMatcher("/user/**")
                .authorizeRequests().anyRequest().authenticated()
                .and()
                .csrf().disable().cors().disable()
                .formLogin()
                        .loginPage("/login")
                                .usernameParameter("email")
                                .loginProcessingUrl("/user/login")
                                .defaultSuccessUrl("/userindex")
                        .permitAll()
                .and()
                .logout()
                        .logoutUrl("/user/logout")
                        .logoutSuccessUrl("/");

        return http.build();
}
```

```
}
```

ADMIN.JAVA :

```
package  com.blackboxdata.model;
import java.util.Collection;
import  javax.persistence.Column;
import javax.persistence.Entity;
import  javax.persistence.GeneratedValue;
import  javax.persistence.GenerationType;
import javax.persistence.Id;
import  org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
@Entity
public class Admin implements UserDetails {
        @Id
        @GeneratedValue(strategy=GenerationType.AUTO)
        @Column(name="id", nullable = false, updatable = false)
        private long id;
        private String username;
        private String email;
        private String password;
        public long getId() {
                return id;
        }
        public void setId(long id) {
                this.id = id;
        }
        public String getUsername() {
                return username;
```

```
}
public void setUsername(String username) {
        this.username = username;
}
public String getEmail() {
        return email;
}
public void setEmail(String email) {
        this.email = email;
}
public String getPassword() {
        return password;
}
public void setPassword(String password) {
        this.password = password;
}
@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
        // TODO Auto-generated method stub
        return null;
}
@Override
public boolean isAccountNonExpired() {
        // TODO Auto-generated method stub
        return false;
}
@Override
public boolean isAccountNonLocked() {
        // TODO Auto-generated method stub
        return false;
    }
    @Override
    public boolean isCredentialsNonExpired() {
        // TODO Auto-generated method stub
        return false;
    }
    @Override
    public boolean isEnabled() {
        // TODO Auto-generated method stub
        return false;
    }
}
```

CLIENT JAVA :

```
package  com.blackboxdata.model;
import java.util.Collection;
import  javax.persistence.Column;
import javax.persistence.Entity;
import  javax.persistence.GeneratedValue;
import  javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.validation.constraints.Size;
import  org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
@Entity
@Table(name = "client")
public class Client implements UserDetails {
    private static final long serialVersionUID = 1L;
```

```
	@Id
	@GeneratedValue(strategy=GenerationType.AUTO)
	@Column(name="id", nullable = false, updatable = false)
	private long id;
	private String username;
	private String gender;
	private String contactno;
	private String email;
	private String password;
	//@Size(min = 2048,max = 2050)
// private String publickey;
	//@Size(min = 2048,max = 2050)
 //private String privatekey;
	private String state;
	private String city;
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getGender() {
		return gender;
```

```
}
public void setGender(String gender) {
	this.gender = gender;
}
public String getContactno() {
	return contactno;
}
public void setContactno(String contactno) {
	this.contactno = contactno;
}
public String getEmail() {
	return email;
}
public void setEmail(String email) {
	this.email = email;
}
public String getPassword() {
	return password;
}
public void setPassword(String password) {
	this.password = password;
}
public String getState() {
	return state;
}
public void setState(String state) {
	this.state = state;
}
public String getCity() {
        return city;
}
public void setCity(String city) {
        this.city = city;
}
public static long getSerialversionuid() {
        return serialVersionUID;
}
@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
        // TODO Auto-generated method stub
        return null;
}
@Override
public boolean isAccountNonExpired() {
        // TODO Auto-generated method stub
        return false;
}
@Override
public boolean isAccountNonLocked() {
        // TODO Auto-generated method stub
        return false;
}
@Override
public boolean isCredentialsNonExpired() {
        // TODO Auto-generated method stub
        return false;
}
@Override
```

```
        public boolean isEnabled() {
                // TODO Auto-generated method stub
                return false;
        }
}
```

USER JAVA :

```
package com.blackboxdata.model;
import java.security.PublicKey;
import java.util.Collection;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import  javax.validation.constraints.Size;
import  org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import com.blackboxdata.security.UserRole;
import com.fasterxml.jackson.annotation.JsonIgnore;
@Entity
@Table(name = "user")
public class User implements UserDetails {
        private static final long serialVersionUID = 1L;

        @Id

        @GeneratedValue(strategy=GenerationType.AUTO)

        @Column(name="id", nullable = false, updatable = false)

        private long id;

        private String username;

        private String gender;

        private String contactno;

        private String email;

        private String password;

        //@Size(min = 2048,max = 2050)

    //private String publickey;

        //@Size(min = 2048,max = 2050)

    //private String privatekey;

    private String carnumber;

    private String licensenumber;

        private String state;

        private String city;

        @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, fetch =
FetchType.EAGER)

        @JsonIgnore

        private Set<UserRole> userRoles = new HashSet<>();

        public long getId() {

                return id;

        }

        public void setId(long id) {

                this.id = id;

        }

        public String getUsername() {
```

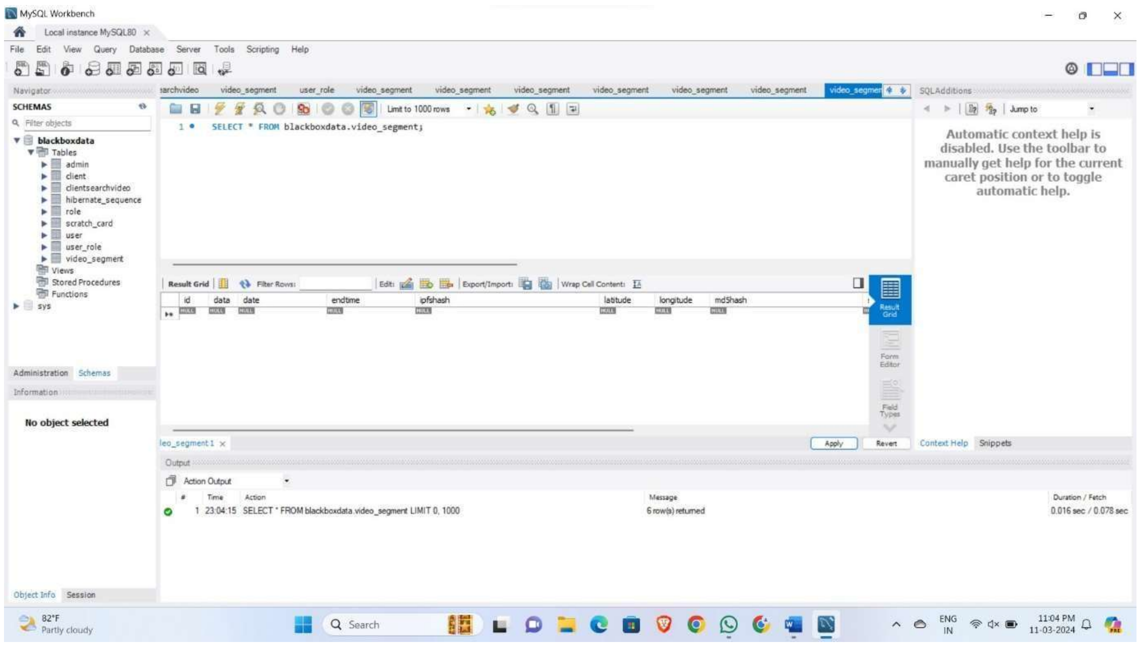
```
        return username;
}
public void setUsername(String username) {
        this.username = username;
}
public String getGender() {
        return gender;
}
public void setGender(String gender) {
        this.gender = gender;
}
public String getContactno() {
        return contactno;
}
public void setContactno(String contactno) {
        this.contactno = contactno;
}
public String getEmail() {
        return email;
}
public void setEmail(String email) {
        this.email = email;
}
public String getPassword() {
        return password;
}
public void setPassword(String password) {
        this.password = password;
}
```

```
public String getCarnumber() {
        return carnumber;
}
public void setCarnumber(String carnumber) {
        this.carnumber = carnumber;
}
public String getLicensenumber() {
        return licensenumber;
}
public void setLicensenumber(String licensenumber) {
        this.licensenumber = licensenumber;
}
public String getState() {
        return state;
}
public void setState(String state) {
        this.state = state;
}
public String getCity() {
        return city;
}
public void setCity(String city) {
        this.city = city;
}
public static long getSerialversionuid() {
        return serialVersionUID;
}
@Override
```

```
        public Collection<? extends GrantedAuthority> getAuthorities() {
                // TODO Auto-generated method stub
                return null;
        }
        @Override
        public boolean isAccountNonExpired() {
                // TODO Auto-generated method stub
                return false;
        }
        @Override
        public boolean isAccountNonLocked() {
                // TODO Auto-generated method stub
                return false;
        }
        @Override
        public boolean isCredentialsNonExpired() {
                // TODO Auto-generated method stub
                return false;
        }
        @Override
        public boolean isEnabled() {
                // TODO Auto-generated method stub
                return false;
        }
        public PublicKey generatePublic(String publikeyStr) {
                // TODO Auto-generated method stub
                return null;
        }
}
```

APPENDICES -II

A.2 SCREENSHOTS

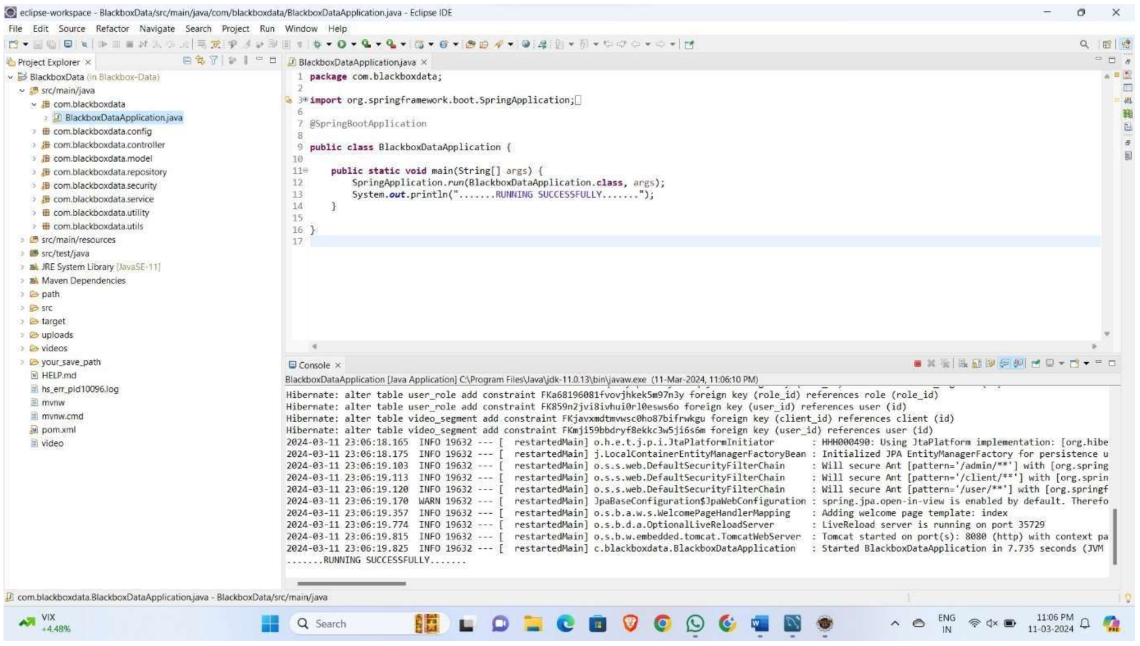


```
SELECT * FROM blackboxdata.video_segment;
```

id	data	date	endtime	ipfshash	latitude	longitude	md5hash
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
#  Time      Action                                                  Message            Duration / Fetch
1  23:04:15  SELECT * FROM blackboxdata.video_segment LIMIT 0, 1000  6 row(s) returned  0.016 sec / 0.078 sec
```

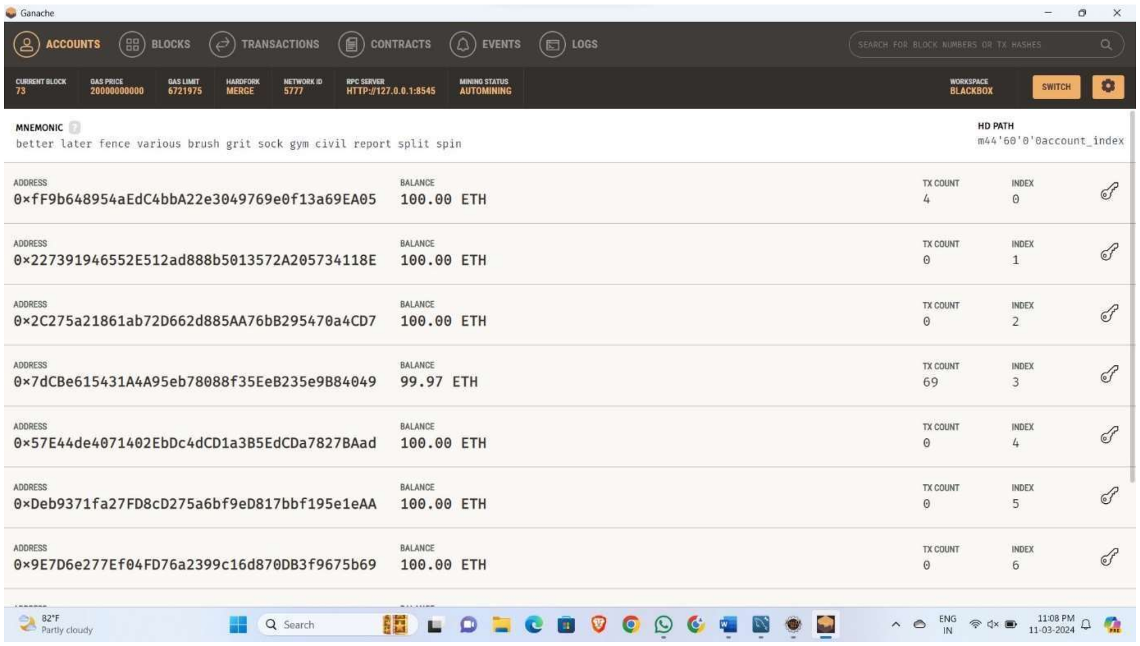
FIG 1.1 MySQL



```java
package com.blackboxdata;

import org.springframework.boot.SpringApplication;

@SpringBootApplication

public class BlackboxDataApplication {

    public static void main(String[] args) {
        SpringApplication.run(BlackboxDataApplication.class, args);
        System.out.println(".......RUNNING SUCCESSFULLY.......");
    }

}
```

```
BlackboxDataApplication [Java Application] C:\Program Files\Java\jdk-11.0.13\bin\javaw.exe (11-Mar-2024, 11:06:10 PM)
Hibernate: alter table user_role add constraint FKa68196081fvovjhkek5m97n3y foreign key (role_id) references role (role_id)
Hibernate: alter table user_role add constraint FK859n2jvi8ivhui0rl0esws6o foreign key (user_id) references user (id)
Hibernate: alter table video_segment add constraint FKjavxmdtmvwsc0ho87bifrwkgu foreign key (client_id) references client (id)
Hibernate: alter table video_segment add constraint FKmji59bbdryf8ekkc3w5ji6s6m foreign key (user_id) references user (id)
2024-03-11 23:06:18.165  INFO 19632 --- [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibe
2024-03-11 23:06:18.175  INFO 19632 --- [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence u
2024-03-11 23:06:19.103  INFO 19632 --- [  restartedMain] o.s.s.web.DefaultSecurityFilterChain     : Will secure Ant [pattern='/admin/**'] with [org.spring
2024-03-11 23:06:19.113  INFO 19632 --- [  restartedMain] o.s.s.web.DefaultSecurityFilterChain     : Will secure Ant [pattern='/client/**'] with [org.sprin
2024-03-11 23:06:19.120  INFO 19632 --- [  restartedMain] o.s.s.web.DefaultSecurityFilterChain     : Will secure Ant [pattern='/user/**'] with [org.springf
2024-03-11 23:06:19.170  WARN 19632 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefo
2024-03-11 23:06:19.357  INFO 19632 --- [  restartedMain] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page template: index
2024-03-11 23:06:19.774  INFO 19632 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2024-03-11 23:06:19.815  INFO 19632 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context pa
2024-03-11 23:06:19.825  INFO 19632 --- [  restartedMain] c.blackboxdata.BlackboxDataApplication   : Started BlackboxDataApplication in 7.735 seconds (JVM
.......RUNNING SUCCESSFULLY.......
```

FIG 1.2 MySQL Tool



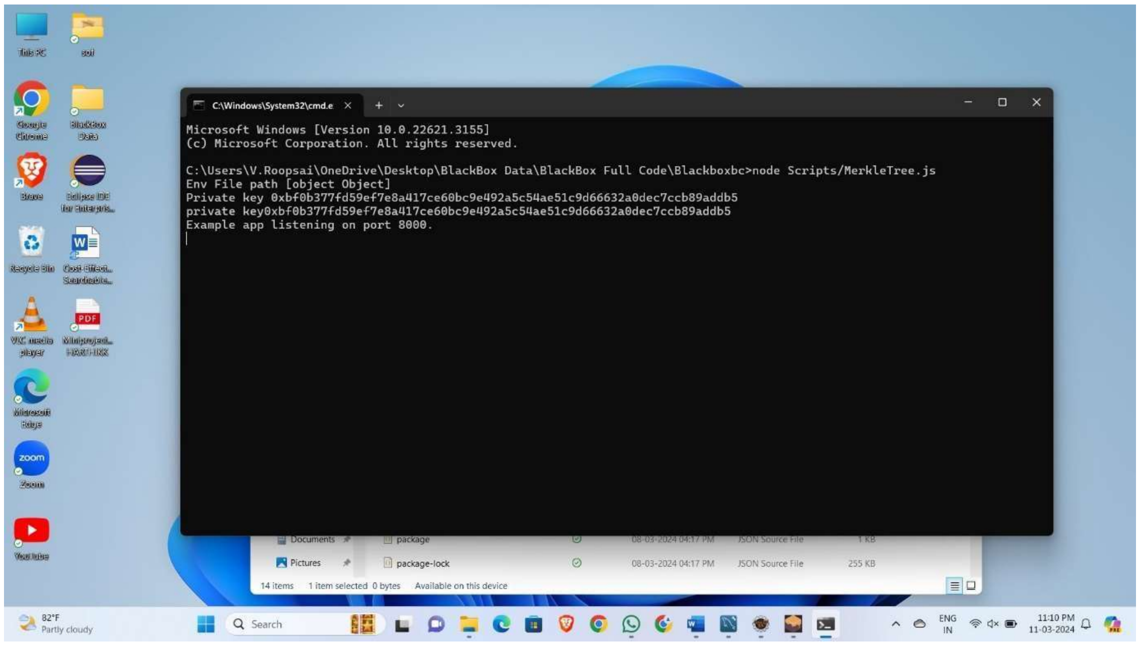
CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
73	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:8545	AUTOMINING	BLACKBOX

MNEMONIC
better later fence various brush grit sock gym civil report split spin

HD PATH
m44'60'0'0account_index

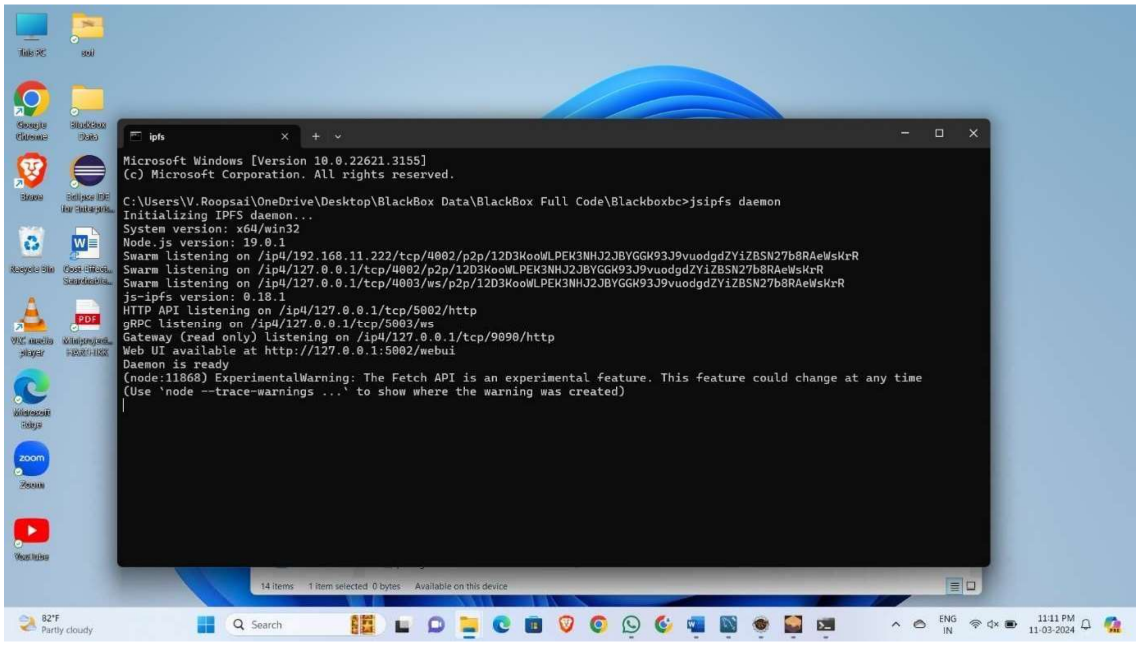
ADDRESS	BALANCE	TX COUNT	INDEX
0×fF9b648954aEdC4bbA22e3049769e0f13a69EA05	100.00 ETH	4	0
0×227391946552E512ad888b5013572A205734118E	100.00 ETH	0	1
0×2C275a21861ab72D662d885AA76bB295470a4CD7	100.00 ETH	0	2
0×7dCBe615431A4A95eb78088f35EeB235e9B84049	99.97 ETH	69	3
0×57E44de4071402EbDc4dCD1a3B5EdCDa7827BAad	100.00 ETH	0	4
0×Deb9371fa27FD8cD275a6bf9eD817bbf195e1eAA	100.00 ETH	0	5
0×9E7D6e277Ef04FD76a2399c16d870DB3f9675b69	100.00 ETH	0	6

FIG 1.3 Ganache



```
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\V.Roopsai\OneDrive\Desktop\BlackBox Data\BlackBox Full Code\Blackboxbc>node Scripts/MerkleTree.js
Env File path [object Object]
Private key 0xbf0b377fd59ef7e8a417ce60bc9e492a5c54ae51c9d66632a0dec7ccb89addb5
private key0xbf0b377fd59ef7e8a417ce60bc9e492a5c54ae51c9d66632a0dec7ccb89addb5
Example app listening on port 8000.
```

FIG 1.4 Set Path Of MerkleTree



```
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\V.Roopsai\OneDrive\Desktop\BlackBox Data\BlackBox Full Code\Blackboxbc>jsipfs daemon
Initializing IPFS daemon...
System version: x64/win32
Node.js version: 19.0.1
Swarm listening on /ip4/192.168.11.222/tcp/4002/p2p/12D3KooWLPEK3NHJ2JBYGGK93J9vuodgdZYiZBSN27b8RAeWsKrR
Swarm listening on /ip4/127.0.0.1/tcp/4002/p2p/12D3KooWLPEK3NHJ2JBYGGK93J9vuodgdZYiZBSN27b8RAeWsKrR
Swarm listening on /ip4/127.0.0.1/tcp/4003/ws/p2p/12D3KooWLPEK3NHJ2JBYGGK93J9vuodgdZYiZBSN27b8RAeWsKrR
js-ipfs version: 0.18.1
HTTP API listening on /ip4/127.0.0.1/tcp/5002/http
gRPC listening on /ip4/127.0.0.1/tcp/5003/ws
Gateway (read only) listening on /ip4/127.0.0.1/tcp/9090/http
Web UI available at http://127.0.0.1:5002/webui
Daemon is ready
(node:11868) ExperimentalWarning: The Fetch API is an experimental feature. This feature could change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
```

FIG 1.5 IPFS Daemon

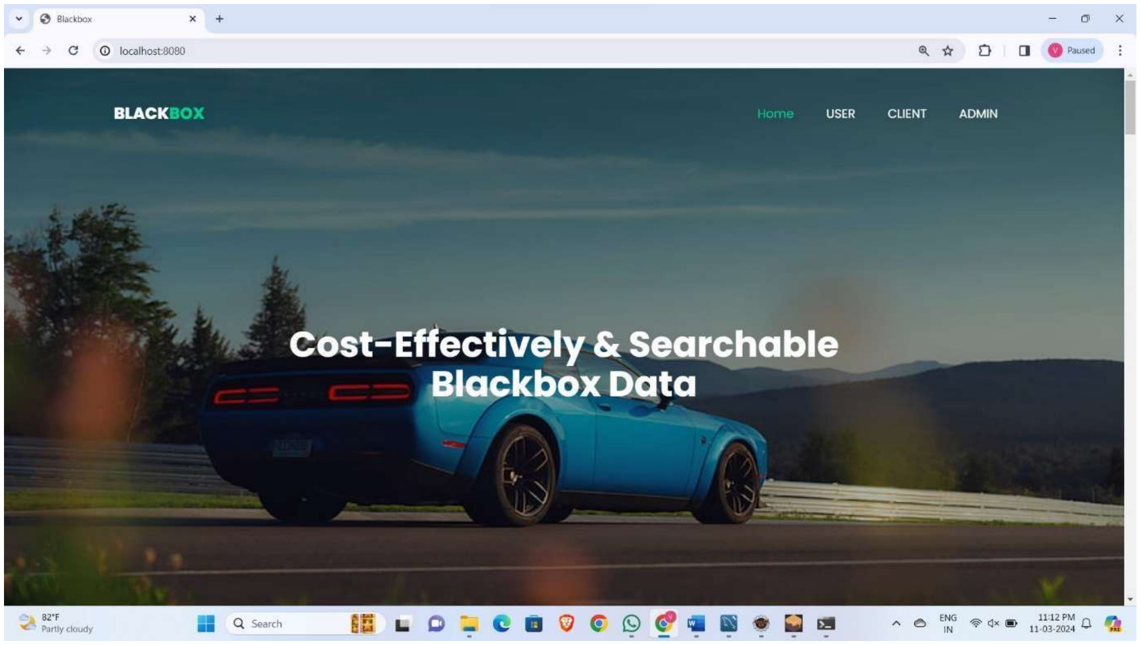


FIG 1.6 Local Host/8080

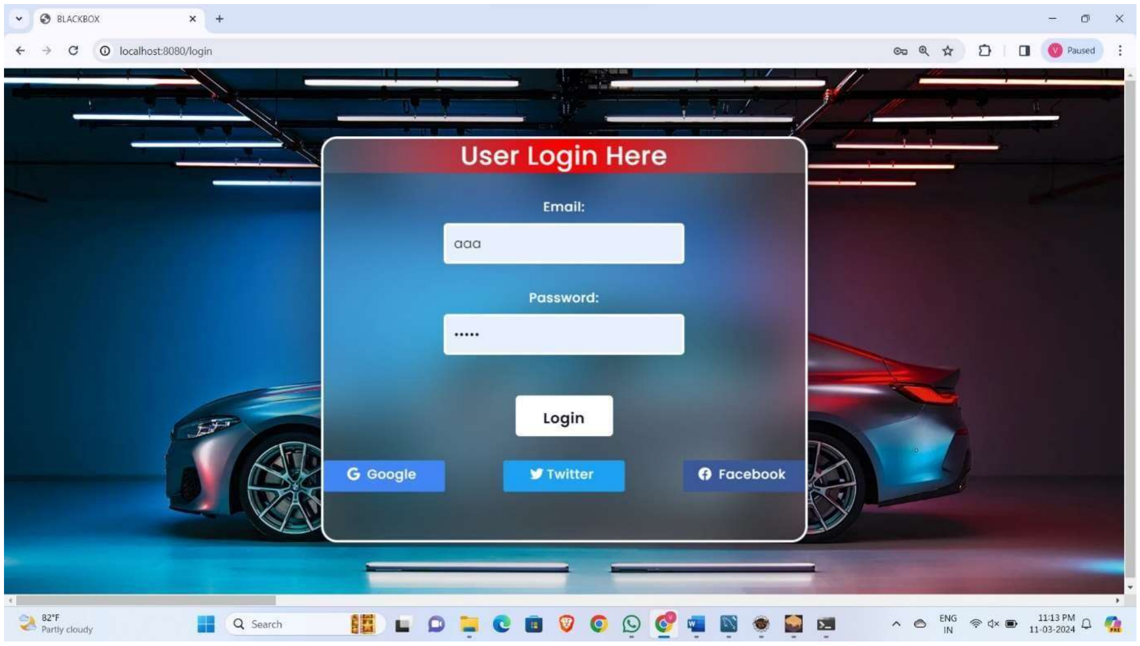


FIG 1.7 User Login

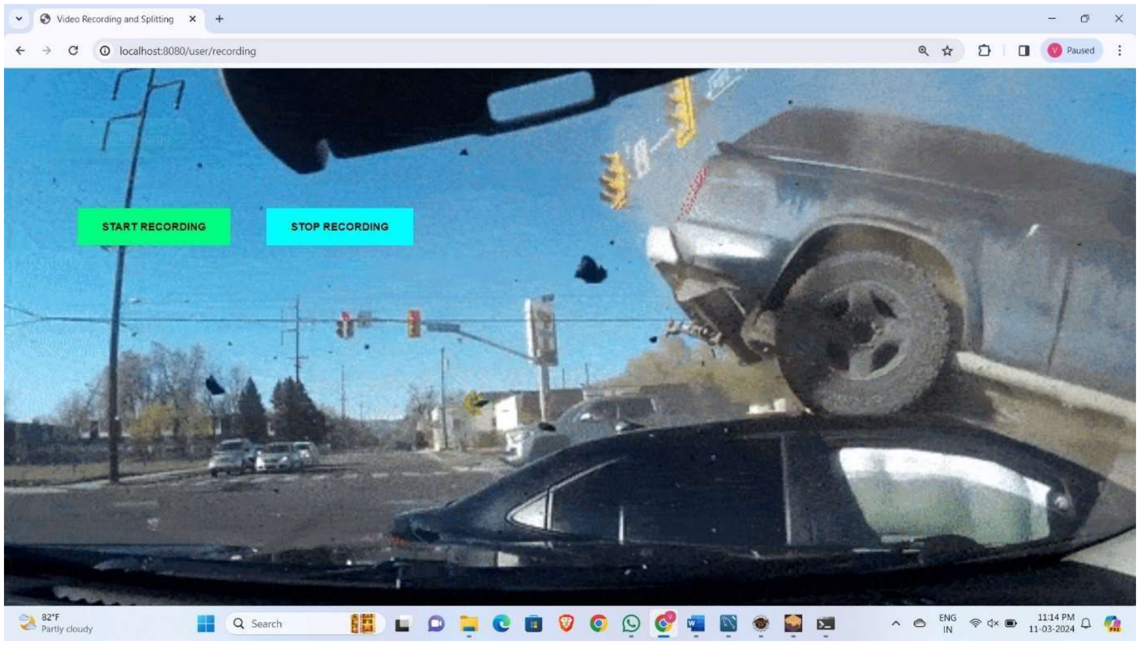
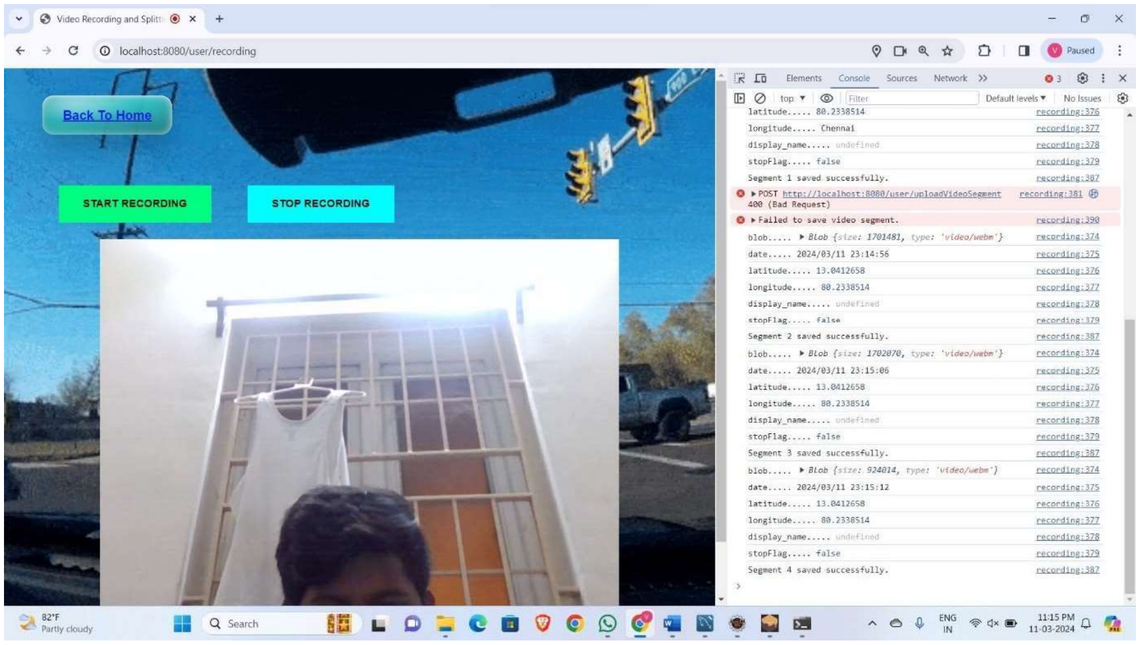


FIG 1.8 Video Recording



```
latitude..... 80.2338514                                          recording:376
longitude..... Chennai                                            recording:377
display_name..... undefined                                       recording:378
stopFlag..... false                                               recording:379
Segment 1 saved successfully.                                     recording:387
▶POST http://localhost:8080/user/uploadVideoSegment          recording:381
400 (Bad Request)
▶Failed to save video segment.                                    recording:390
blob..... ▶Blob {size: 1701481, type: 'video/webm'}               recording:374
date..... 2024/03/11 23:14:56                                     recording:375
latitude..... 13.0412658                                          recording:376
longitude..... 80.2338514                                         recording:377
display_name..... undefined                                       recording:378
stopFlag..... false                                               recording:379
Segment 2 saved successfully.                                     recording:387
blob..... ▶Blob {size: 1702070, type: 'video/webm'}               recording:374
date..... 2024/03/11 23:15:06                                     recording:375
latitude..... 13.0412658                                          recording:376
longitude..... 80.2338514                                         recording:377
display_name..... undefined                                       recording:378
stopFlag..... false                                               recording:379
Segment 3 saved successfully.                                     recording:387
blob..... ▶Blob {size: 924014, type: 'video/webm'}                recording:374
date..... 2024/03/11 23:15:12                                     recording:375
latitude..... 13.0412658                                          recording:376
longitude..... 80.2338514                                         recording:377
display_name..... undefined                                       recording:378
stopFlag..... false                                               recording:379
Segment 4 saved successfully.                                     recording:387
```

FIG 1.9 Video File

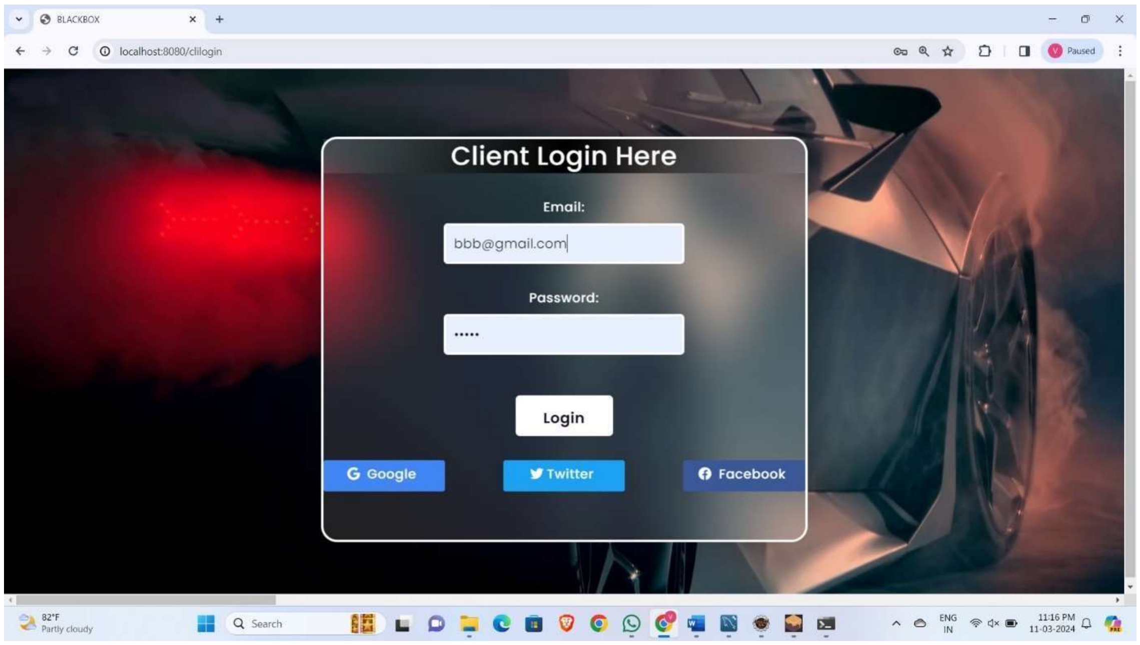


FIG 1.10 Client Login

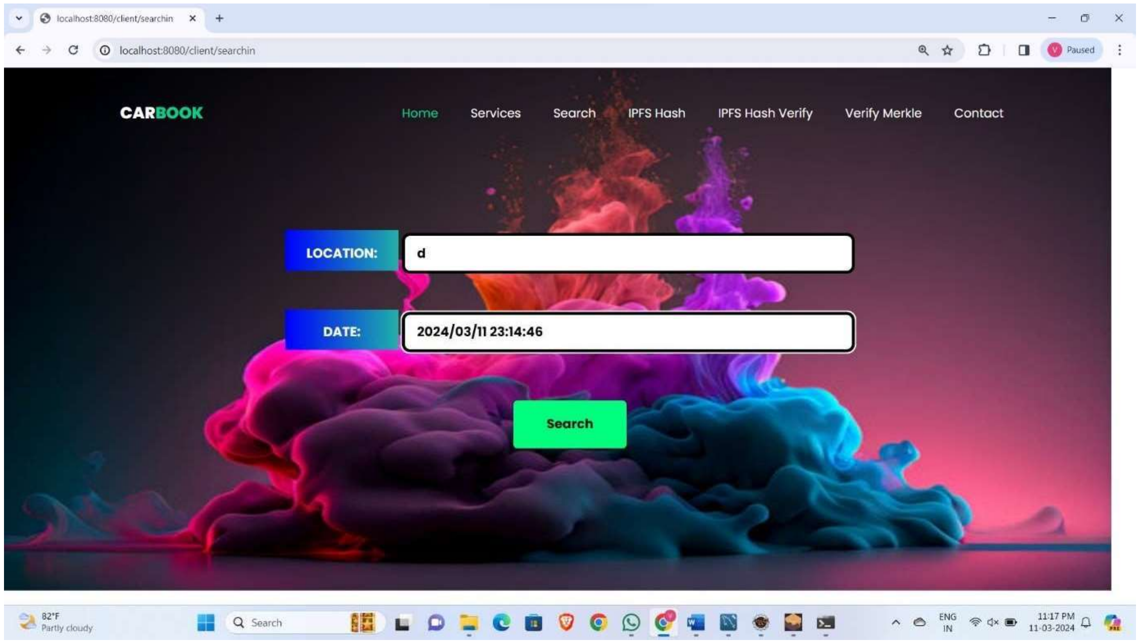


FIG 1.11 Search Data

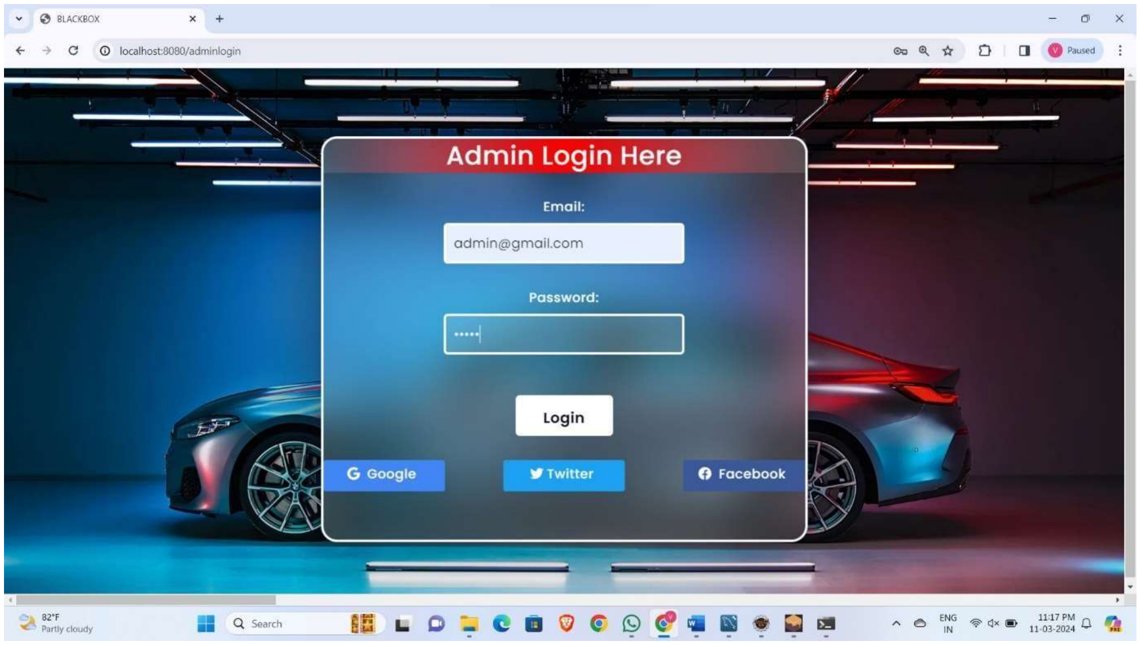
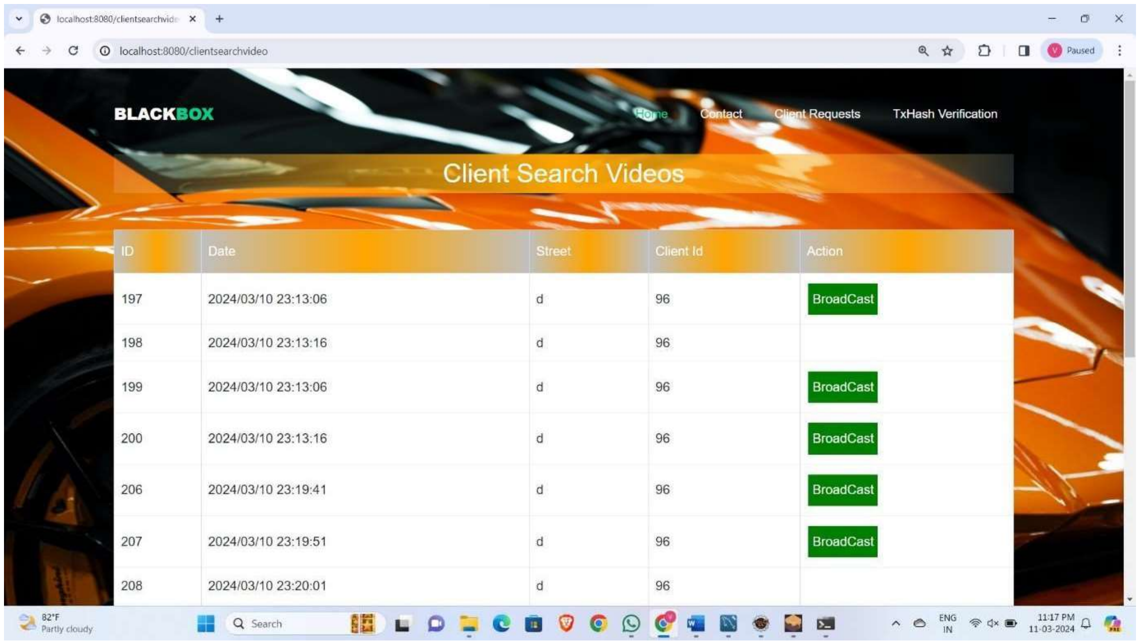
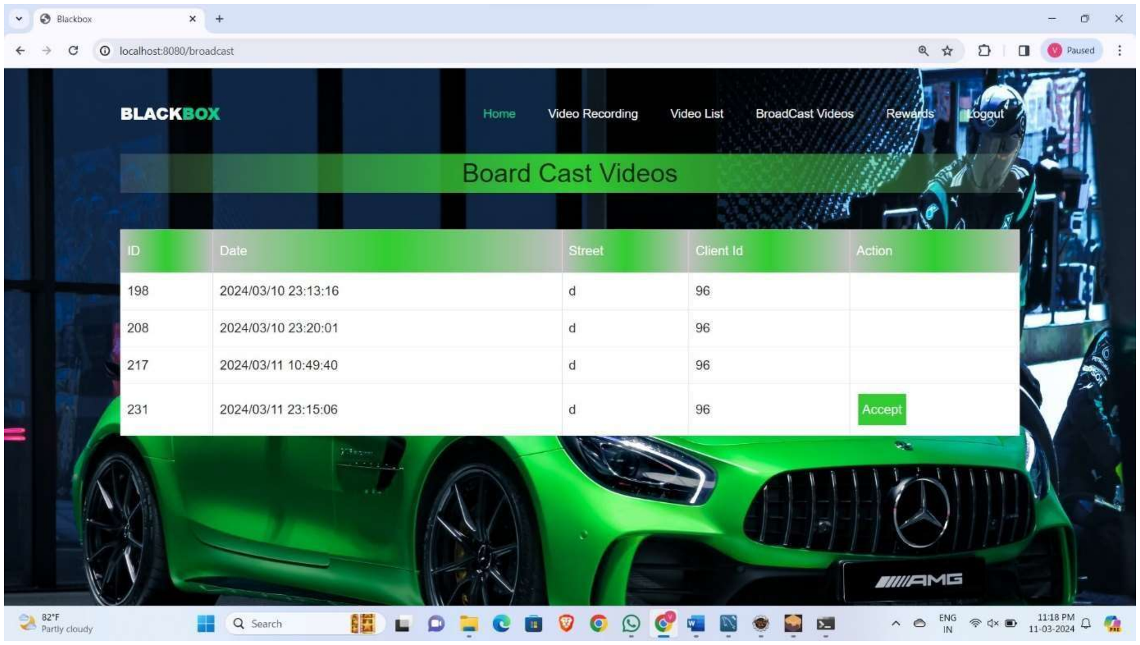


FIG 1.12 Admin Login



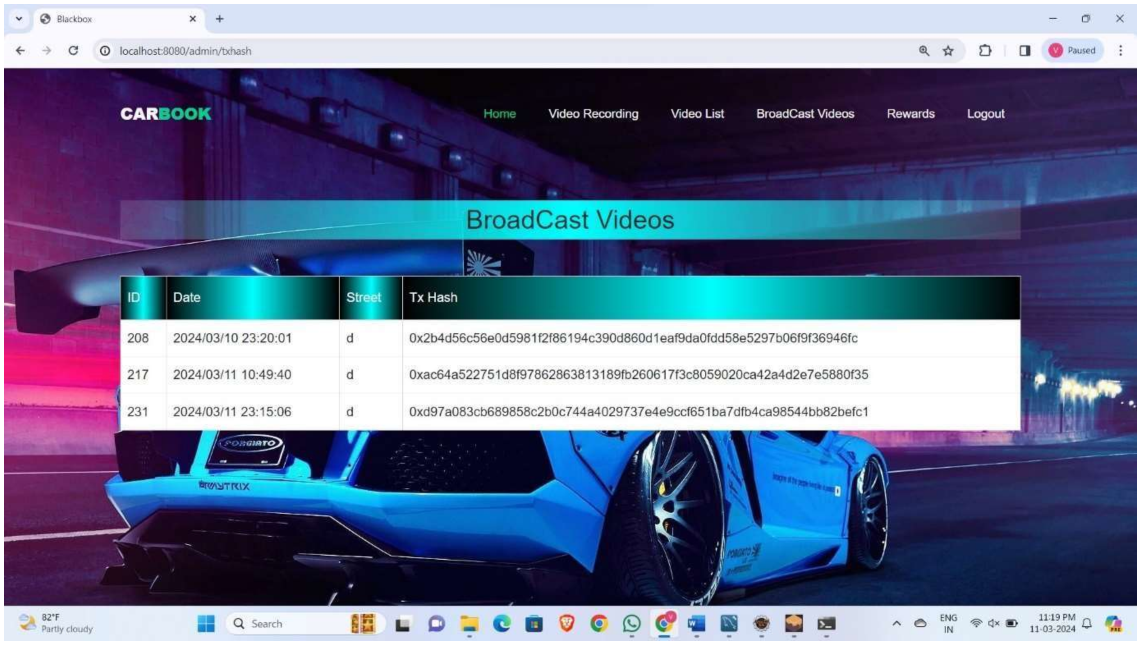
ID	Date	Street	Client Id	Action
197	2024/03/10 23:13:06	d	96	BroadCast
198	2024/03/10 23:13:16	d	96	
199	2024/03/10 23:13:06	d	96	BroadCast
200	2024/03/10 23:13:16	d	96	BroadCast
206	2024/03/10 23:19:41	d	96	BroadCast
207	2024/03/10 23:19:51	d	96	BroadCast
208	2024/03/10 23:20:01	d	96	

FIG 1.13 Client SearchVideo



ID	Date	Street	Client Id	Action
198	2024/03/10 23:13:16	d	96	
208	2024/03/10 23:20:01	d	96	
217	2024/03/11 10:49:40	d	96	
231	2024/03/11 23:15:06	d	96	Accept

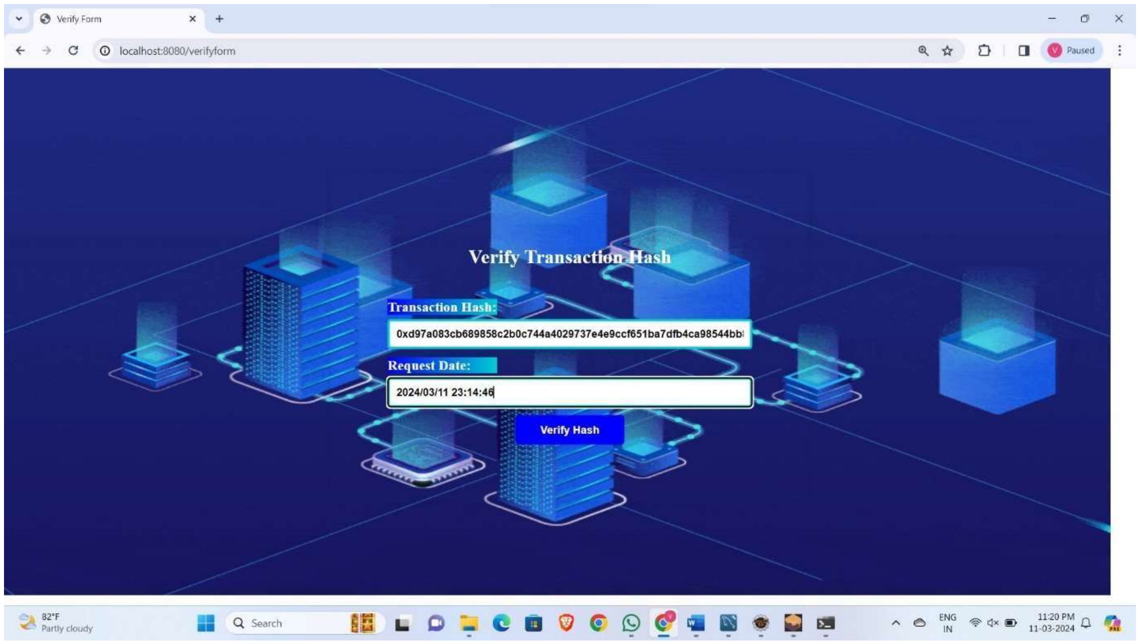
FIG 1.14 Accept Video



BroadCast Videos

ID	Date	Street	Tx Hash
208	2024/03/10 23:20:01	d	0x2b4d56c56e0d5981f2f86194c390d860d1eaf9da0fdd58e5297b06f9f36946fc
217	2024/03/11 10:49:40	d	0xac64a522751d8f97862863813189fb260617f3c8059020ca42a4d2e7e5880f35
231	2024/03/11 23:15:06	d	0xd97a083cb689858c2b0c744a4029737e4e9ccf651ba7dfb4ca98544bb82befc1

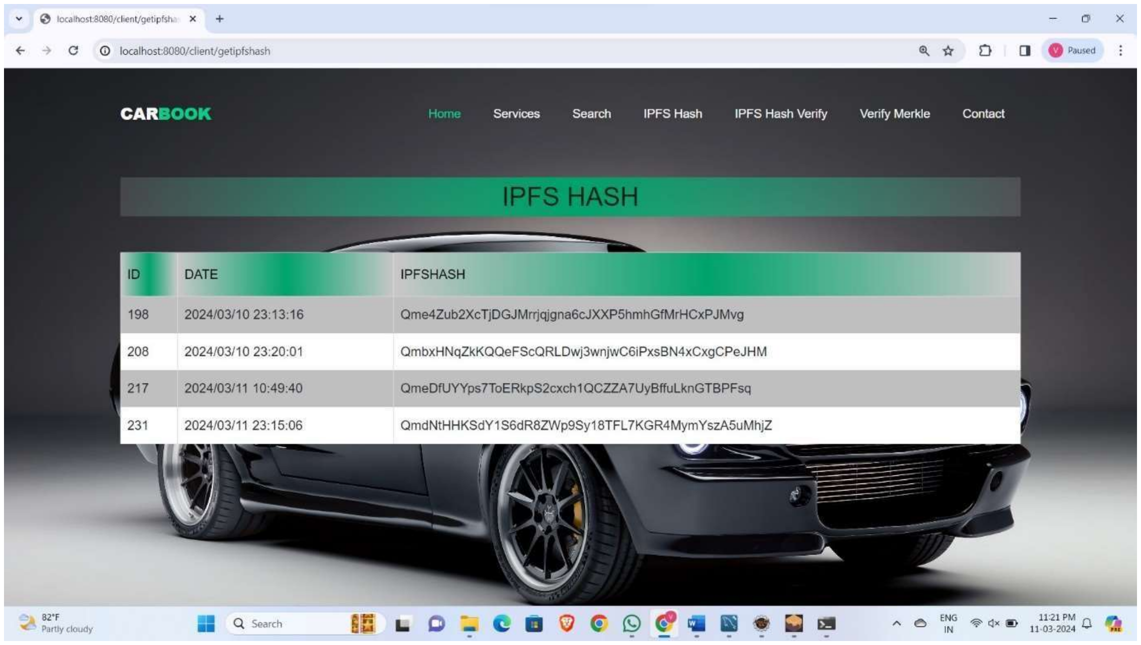
FIG 1.15 BroadCast Video



Transaction Hash: 0xd97a083cb689858c2b0c744a4029737e4e9ccf651ba7dfb4ca98544bb

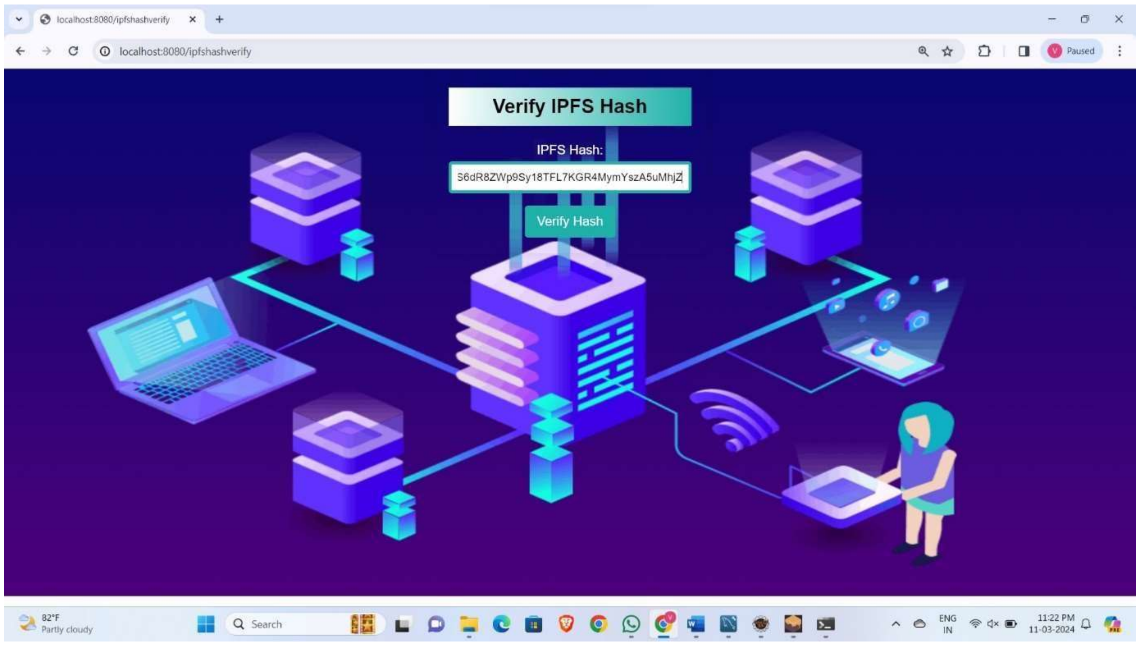
Request Date: 2024/03/11 23:14:46

FIG 1.16 Verify Transaction Hash



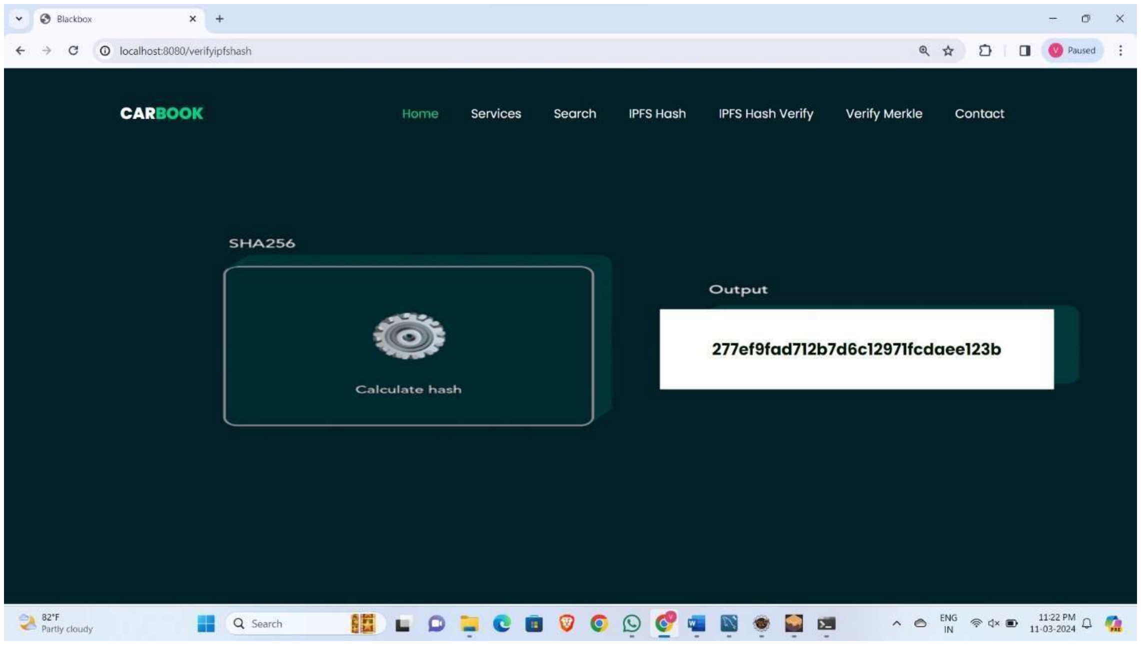
ID	DATE	IPFSHASH
198	2024/03/10 23:13:16	Qme4Zub2XcTjDGJMrrjqjgna6cJXXP5hmhGfMrHCxPJMvg
208	2024/03/10 23:20:01	QmbxHNqZkKQQeFScQRLDwj3wnjwC6iPxsBN4xCxgCPeJHM
217	2024/03/11 10:49:40	QmeDfUYYps7ToERkpS2cxch1QCZZA7UyBffuLknGTBPFsq
231	2024/03/11 23:15:06	QmdNtHHKSdY1S6dR8ZWp9Sy18TFL7KGR4MymYszA5uMhjZ

FIG 1.17 IPFS Hash



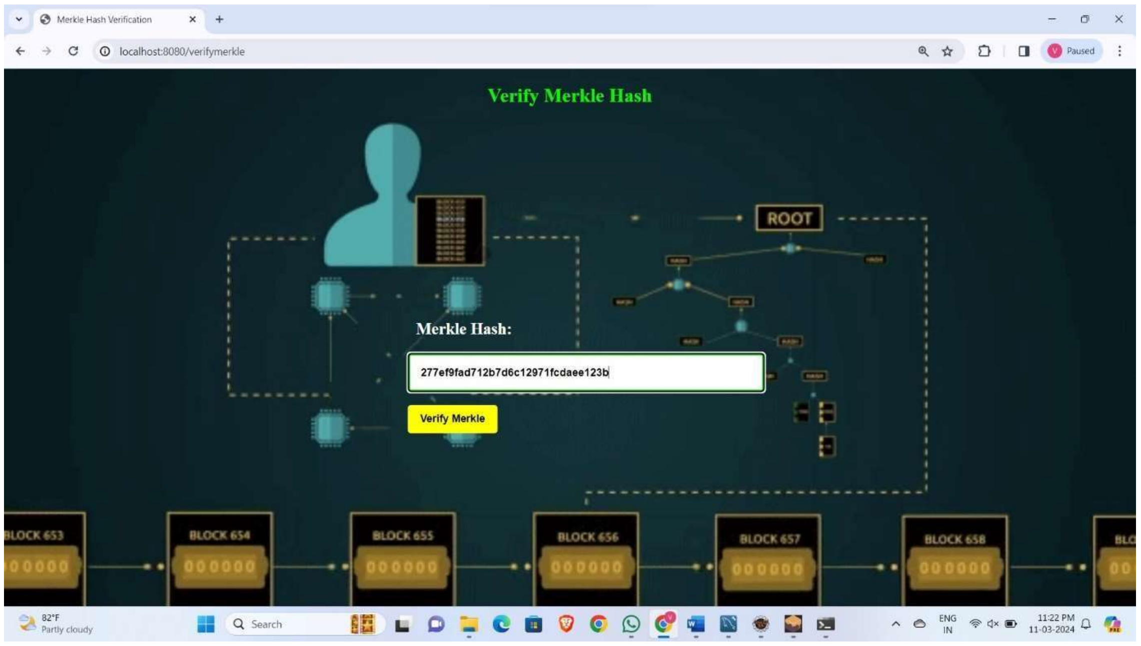
IPFS Hash: S6dR8ZWp9Sy18TFL7KGR4MymYszA5uMhjZ

FIG 1.18 Verify IPFS Hash



277ef9fad712b7d6c12971fcdaee123b

FIG 1.19 Calculate Hash



277ef9fad712b7d6c12971fcdaee123b

FIG 1.20 Verify Merkle Hash

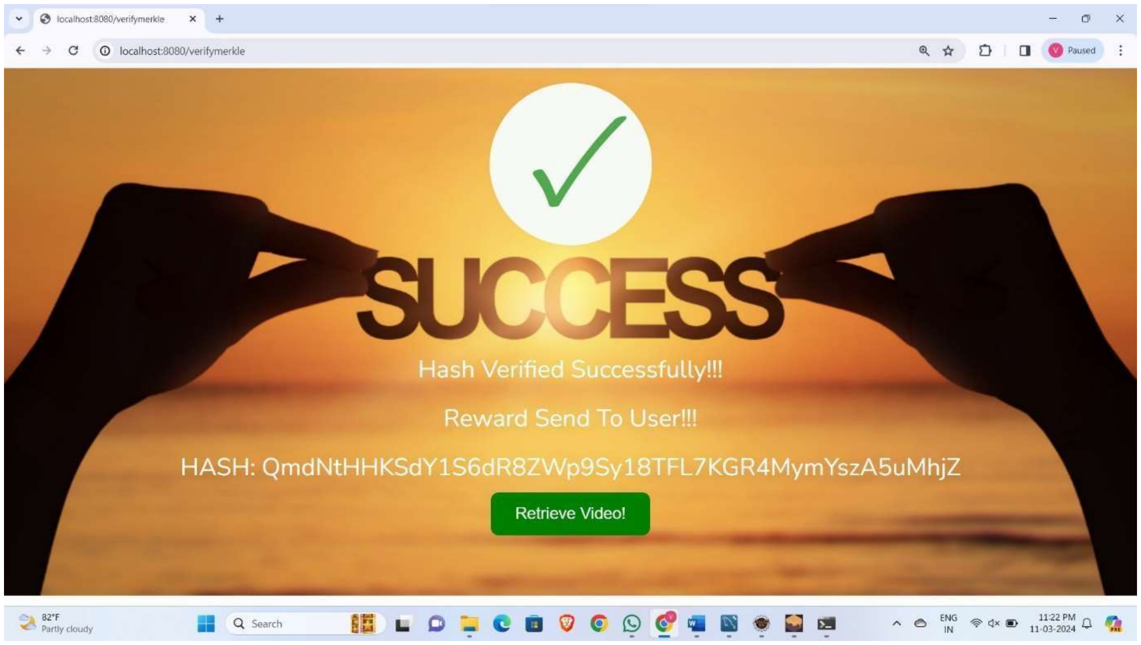

localhost:8080/verifymerkle
SUCCESS
Hash Verified Successfully!!!
Reward Send To User!!!
HASH: QmdNtHHKSdY1S6dR8ZWp9Sy18TFL7KGR4MymYszA5uMhjZ
Retrieve Video!

FIG 1.21 Hash Verified

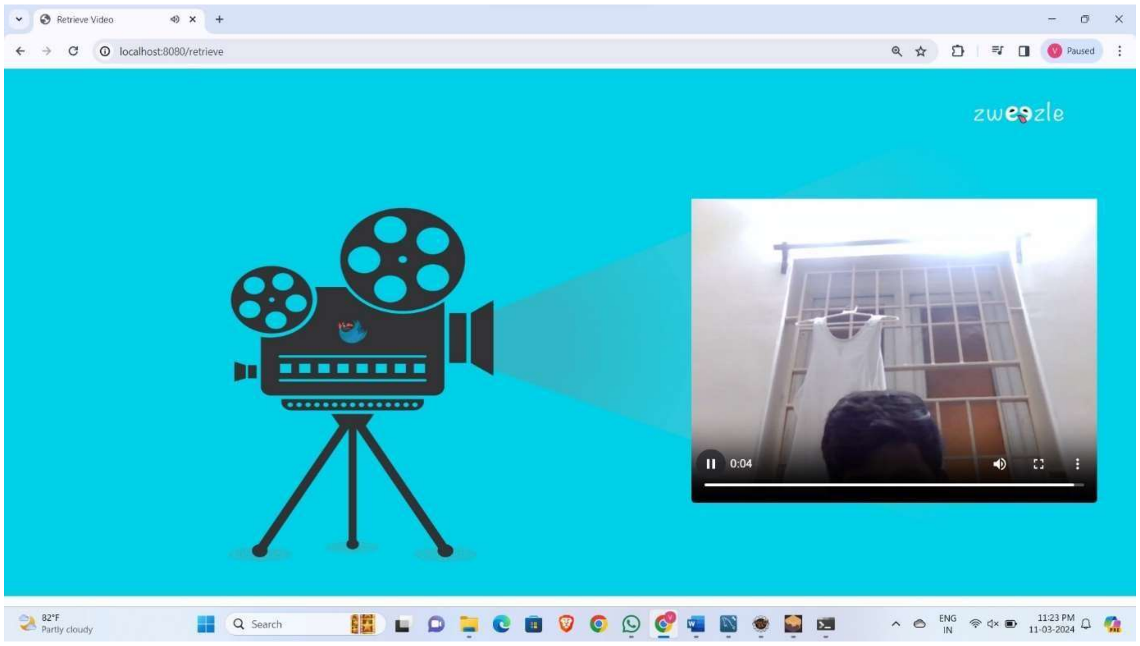


FIG 1.22 Video Monitering

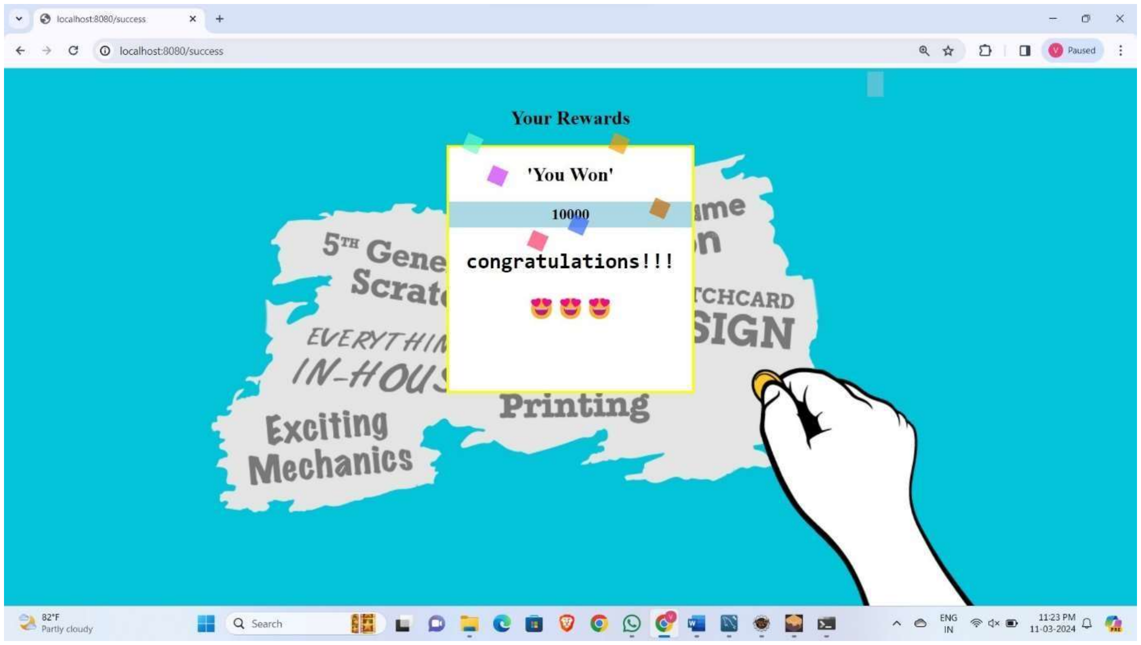


FIG 1.23 Rewards

A.3 PLAGIARISM REPORT

Authenticated by ANTIPLA plagiarism checker
Date of issuance 2024-03-23 02:34:05
Accessed via on www.antip.la

Result

The system found similarities with other sources, however, the text successfully passed the plagiarism check.

Analysis

Result	8 % .
Document title	conference paper.docx
Content hash	30dc5213294f39e54e01532ce9df7571
Date	2024-03-23 02:33:26
Check time	21 seconds
Character count	10,000
Special character count	21
Word count	1,567
Number of plagiarized words	123