**A Blockchain based Autonomous Decentralized Online Social Network**

****

Mini Project submitted in partial fulfillment of the requirement for the award of the

degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

Under the esteemed guidance of

**Mr.xxxxxxx**

**Designation of the Guide**

By

**Name of the student (HT.No)**



**Department of Computer Science and Engineering**

**Accredited by NBA**

**Geethanjali College of Engineering and Technology**

**(UGC Autonomous)**

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)

Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

**October-2024**

**Geethanjali College of Engineering & Technology**

**(UGC Autonomous)**

(Affiliated to JNTUH, Approved by AICTE, New Delhi)

Cheeryal (V), Keesara(M), Medchal Dist.-501 301.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Accredited by NBA**



###### Certificate

This is to certifythat the B.Tech Mini Project report entitled **“A Blockchain based Autonomous Decentralized Online Social Network”** is a bonafide work done by Name of the student **(HT.NO),** Name of the student **(HT.NO), Name of the student (HT.NO)**, in partial fulfillment of the requirement of the award for the degree of Bachelor of Technology in “**Computer Science and Engineering**” from Jawaharlal Nehru Technological University, Hyderabad during the year 2023-2024.

**Internal Guide**  HOD - CSE

**XXXXXXXXX Dr A SreeLakshmi**  Designation Professor

External Examiner

**3rd page : Company Certificate with Letter Head**

**Geethanjali College of Engineering & Technology**

**(UGC Autonomous)**

(Affiliated to JNTUH Approved by AICTE, New Delhi)

Cheeryal (V), Keesara(M), Medchal Dist.-501 301.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Accredited by NBA**



###### DECLARATION BY THE CANDIDATE

I/We, **Name of the Candidate(s)**, bearing Roll Nos. **Roll No.**, hereby declare that the project report entitled **“A Blockchain based Autonomous Decentralized Online Social Network”** is done under the guidance of **Mr./Ms. Name of the Guide**, **Designation**, Department of Computer Science and Engineering, Geethanjali College of Engineering and Technology, is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**.

This is a record of bonafide work carried out by me/us in {**Name of the Institute where you have done your project** ] and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

**Name of the Student(s), Roll No(s).,**

**Department of CSE,**

**Geethanjali College of Engineering and Technology,**

**Cheeryal.**

**ACKNOWLEDGEMENT**

ABSTRACT

In this project, we introduce a novel approach to web hosting and data integrity using InterPlanetary File System (IPFS) and Blockchain technologies. Our system offers users the ability to host their websites on IPFS, a decentralized and distributed file system, ensuring high availability and redundancy. Additionally, Blockchain is employed to manage and maintain the integrity of IPFS data through the use of immutable records. The proposed system begins with user registration within the application, followed by a request to the administrator for web hosting. The administrator verifies the user's request and, upon approval, accepts the user's web code. This code is then deployed on IPFS, generating a unique hash address for the hosted website. To secure and validate this process, the hash address is inserted into the Blockchain using a Smart Contract. This system simplifies web deployment by providing a sample website for users, while also offering the flexibility to add as many pages as desired. By combining IPFS and Blockchain technologies, we ensure the availability, security, and immutability of hosted web content, providing users with a reliable and resilient platform for their online presence.

List of figures/diagrams/graphs

List of Tables

List of abbreviations

Important Note:

* All the above pages are to be numbered in Roman numerals of lower case. Ex. i, ii, iii, iv,…
* The Table of contents must be numbered using numbers i.e. 1, 2, 3……

**TABLE OF CONTENTS**

S.No Contents Page no

**Abstract**

**List of Figures**

**List of Tables**

**List of Screen shots**

**List of Symbols & Abbreviations**

**1. Introduction……………………………………………………………**

1.1 About the project

1.2 Objective

**2. System Analysis………………………………………………………….**

2.1 Existing System

2.2 Proposed System

2.3 Feasibility Study

2.3.1 Details

2.3.2 Impact on Environment

*impact on environment (not OS or SW used), Examples – Reduction in global warming, reduce pollution, simplicity of usage, time reduction etc.,*

2.3.3 Safety

*Impact on various areas mentioned (but not limited to) Security (data, network, information), privacy etc.,*

2.3.4 Ethics

*General SW ethics for building an application or solution like (but not limited to) – does not harm any person (physically or virtually), securing privacy information of the resources using application (secure login, not exposing personal details in any form) etc.,*

2.3.5 Cost

*Cost of development, usage, maintenance etc.,*

*Cost reduction due to implementation of the project in production*

2.3.6 Type

*application (Mobile app, web, standalone etc.,), product, research, review etc.,*

2.4 Scope of the Project

2.5 Modules Description

2.6 System Configuration

**3. Literature Overview……………………………………………….**

3.1 Minimum 5 papers referred – in the process of identifying their problem

Along with its description

**4. System Design…………………………………………………………**

4.1 System Architecture

4.1.1 Module Description

<<Every project includes various modules consisting of a group of processes of a functional area, Give the description of the modules of your project>>

* 1. System Design
  2. System Design
     1. Module Design < how each module is designed of your project
     2. Database Design <which can discussed by used data sets of your project>

**5. Implementation………………………………………………………………………**

5.1 Implementation (Explanation of the complete project along with datasets used and algorithms implemented)

5.2 Sample code

**6. Testing………………………………………………………………………………..**

6.1 Testing

6.2 Test cases

Test cases and output should cover all the possible scenarios of application usage and should also justify and support the points mentioned in sections from 2.2.2 to 2.2.7

**7. Output Screens………………………………………………………………………**

**8. Conclusion…………………………………………………………………………….**

8.1 Conclusion

8.2 Further Enhancements

**9. Bibliography………………………………………………………………………….**

9.1 Books References

9.2 Websites References

9.3 Technical Publication References.

**10 Appendices**

**A. SW used**

**B. Methodologies used**

**C. Testing Methods used etc.,**

**Etc.,**

**1. Introduction**

1.1 About the Project

The project at hand represents a comprehensive exploration of a multifaceted and innovative approach to web hosting and data management. By leveraging cutting-edge technologies, such as InterPlanetary File System (IPFS) and Blockchain, we aim to revolutionize web hosting services and enhance data integrity.

Online Social Network (OSN) is a platform for people to build connections with each other via the Internet. It is a major platform that the public can obtain and disseminate information, exchange views and share their lives on it. Research from Chaffey [1] reveals the liveness of top used OSN in the world (Figure 1), therefore, it can be found that interacting with OSN is a very popular online activity for Internet users.

Nowadays, most of OSNs are centralized, which means the OSN companies often have full ownership of all user data and service. In general, users can only use the service after they agree to the agreements of OSN which are enacted by OSN companies. However, many agreements give the OSN companies right to use user data for personalized services such as advertisement. If users do not allow the companies to use their data and protect their privacy, they usually have to make a series of expatiatory applications or even give up using such OSN. Data and service centralization also caused all data of users is uploaded and stored in centralized servers which are controlled by OSN companies. Therefore, it is hard for users to protect their contents on the OSN when the servers crash down. To make matters worse, if the servers are hacked, security information includes password, security problems, address of users is possible to be leaked. For many users using the same password in kinds of sites, hackers can easily hack their accounts by using a method named credential stuffing attack [2]. This makes personal information of users at risk of leakage and abuse. Such problems of centralized OSNs boost researchers to consider develop an OSN based on the decentralization framework.

Decentralized OSNs have the potential to provide a safer and more controllable social network environment for users where privacy and information are more controllable for their owners. Because the data is stored in a distributed way and service is no longer relied on centralized servers. In general, a decentralized OSN is usually operated by a peer-to-peer mechanism in which each node stores some parts of data and support the service. However, it is not binding on malicious acts, and lack of self-management and sustainable developing abilities.

In this paper, we proposed an autonomous decentralized online social network architecture based on blockchain technology. Blockchain is able to provide a safe and trusted peer-to-peer mechanism where participants have unique identities and private keys. The private key has the highest control right of the corresponding account and is stored in user’s own device. Moreover, all transactions in blockchain need to be signed by the private key, so cheating can be avoided. In order to give the system abilities of selfmanagement and sustainable development, a decentralized autonomous mechanism powered by blockchain is embedded in the architecture. The rest of this paper is organized as follows. Firstly, we introduce the background of related technologies used in this architecture. Secondly, a detail description of the architecture is discussed. Thirdly, functions of this project are showed. Finally, a conclusion is made.

1.2 Objective

The primary objective of this project is to create a robust and decentralized web hosting platform that ensures high availability, data immutability, and security for hosted websites. We seek to achieve the following key objectives:

* Implement IPFS for hosting websites, offering decentralization and redundancy.
* Utilize Blockchain to manage and verify the integrity of hosted data.
* Enable user registration and website deployment through a user-friendly interface.
* Facilitate the insertion of web content into the Blockchain via Smart Contracts.
* Enhance the reliability and resilience of web hosting services.
* Offer a simplified web deployment process, including a sample website for user convenience.

Through the pursuit of these objectives, we aim to provide users with a seamless and trustworthy solution for web hosting, backed by cutting-edge technologies and a user-centric approach. An implementation of blockchain used in OSN. Users keep their security information under their control, in order to avoid security information leakage from centralized servers.

**2. SYSTEM ANALYSIS**

In this section, we delve into a comprehensive analysis of the system, including an evaluation of the existing system, the proposed system, and a feasibility study to assess the viability of our project.

**2.1 Existing System**

The existing web hosting and data management systems predominantly rely on centralized servers and traditional databases. These systems have several limitations:

Centralization: Existing systems are highly centralized, making them vulnerable to single points of failure and downtime.

Data Integrity: Ensuring the integrity of hosted data can be challenging, as centralized databases are susceptible to unauthorized alterations.

Scalability: Traditional systems may struggle to efficiently scale with increasing demand, resulting in performance bottlenecks.

Complexity: Setting up and managing websites often involves a steep learning curve for users.

**2.2 Proposed System**

Our proposed system offers a paradigm shift in web hosting and data management:

Decentralization: We leverage IPFS to host websites, ensuring decentralization and redundancy. This means that hosted data is distributed across multiple nodes, reducing the risk of downtime.

Blockchain Integration: The integration of Blockchain technology enhances data integrity. Each block on the Blockchain is associated with a unique hash code, preventing unauthorized data alterations.

User-Friendly: Our system provides a user-friendly interface for user registration and website deployment, simplifying the process for both novice and experienced users.

Smart Contracts: Smart Contracts are used to insert web content into the Blockchain, further enhancing data security and transparency.

**2.3 Feasibility Study**

A feasibility study has been conducted to evaluate the practicality and viability of our project:

Technical Feasibility: The required technologies, such as IPFS and Blockchain, are well-established and readily available. The technical infrastructure for implementation is feasible.

Operational Feasibility: The proposed system streamlines web hosting operations, making it operationally feasible. Users will find it easy to register and deploy websites.

Economic Feasibility: While there are costs associated with implementing and maintaining this system, the potential benefits, including improved web hosting reliability and data security, make it economically viable.

Legal and Ethical Feasibility: The system complies with legal and ethical standards, ensuring user data privacy and security.

In conclusion, our proposed system addresses the shortcomings of existing web hosting and data management systems by embracing decentralization, data integrity, and user-friendliness. The feasibility study indicates that the project is technically, operationally, economically, and ethically viable.

**2.3.1 Details**

In this subsection, we provide a more detailed examination of the feasibility aspects related to our project.

**2.3.2 Impact on Environment**

Our project's environmental impact extends beyond the realm of the operating system or software used. It encompasses broader aspects that contribute to a sustainable and eco-friendly approach:

Reduction in Global Warming: By leveraging decentralization and distributed data hosting through IPFS, our system reduces the reliance on energy-intensive centralized data centers. This, in turn, leads to a decrease in the carbon footprint associated with data hosting.

Reduced Pollution: With fewer data centers and servers running around the clock, the project contributes to lower energy consumption, resulting in reduced pollution associated with power generation.

Simplicity of Usage: The user-friendly interface encourages users to adopt our system for web hosting, reducing the need for complex and resource-intensive infrastructure. This simplicity not only saves time but also minimizes the environmental impact of setting up and maintaining traditional web hosting environments.

Time Reduction: Faster and more efficient web hosting processes lead to reduced energy consumption and, consequently, a positive impact on the environment.

**2.3.3 Safety**

The safety aspects of our project encompass various areas, including security, data protection, and privacy:

Security: Our system employs Blockchain technology, known for its robust security features. Each block's unique hash code ensures data integrity and prevents unauthorized alterations. This enhances the security of hosted websites and the data associated with them.

Data Security: Decentralization and distributed hosting reduce the risk of data loss due to server failures or cyberattacks. User data and web content are stored across multiple nodes, making it resilient to single points of failure.

Network Security: The decentralized nature of IPFS and Blockchain mitigates the risk of network-based attacks. With data distributed across multiple nodes, the system becomes less susceptible to network breaches.

Privacy: We prioritize user privacy by design. Personal data is securely handled during the registration process, and our system complies with privacy regulations. Users can have confidence in the confidentiality of their information.

In summary, our project's environmental impact is positive, as it promotes energy efficiency, reduces pollution, and simplifies processes. On the safety front, we prioritize data and network security, as well as user privacy, to ensure a safe and secure web hosting environment.

**2.3.4 Ethics**

Ethical considerations are fundamental to our project, ensuring that it adheres to general software ethics and guidelines:

No Harm to Individuals: Our application is designed with the utmost care to ensure it does not harm any person physically or virtually. This means that the application and the websites hosted through it should not facilitate or engage in any harmful or malicious activities.

Privacy Protection: We prioritize the security of user data and the resources using the application. This includes implementing secure login procedures and safeguarding personal details. We strictly adhere to privacy regulations and guidelines to protect user information.

**2.3.5 Cost**

Cost considerations are essential in any project, and our system addresses various cost aspects:

Cost of Development: We have assessed the development costs, including software development, infrastructure setup, and testing efforts, to ensure efficient resource utilization during the project's creation.

Usage Costs: Users of our system will benefit from reduced hosting costs compared to traditional hosting solutions. The decentralized nature of IPFS and the efficiency of our application lead to cost savings in the long run.

Maintenance Costs: Ongoing maintenance is an integral part of any software project. We have considered the maintenance costs associated with ensuring the continued functionality, security, and scalability of our system.

Cost Reduction: By implementing our project in production, organizations can realize significant cost reductions in web hosting. The decentralized approach lowers infrastructure and operational costs compared to centralized hosting solutions.

**2.3.6 Type**

Our project falls under the following categories:

Application: Our system is primarily an application, accessible through web interfaces. Users can interact with it through web browsers, making it a web application. Additionally, it serves as a backend service for web hosting.

Product: The project can also be categorized as a product since it offers a tangible solution for web hosting needs. Organizations and individuals can use it as a product to deploy and manage their websites effectively.

In conclusion, our project adheres to ethical principles, emphasizing harm prevention and privacy protection. We have considered various cost factors, including development, usage, and maintenance costs, with the aim of delivering a cost-effective web hosting solution. Our project is primarily an application and can be considered a product that offers tangible benefits to users.

**2.4 Scope of the Project**

The scope of our project encompasses the following key aspects:

Web Hosting: Our system enables users to host their websites on the IPFS network, ensuring decentralization and improved accessibility.

Blockchain Integration: We incorporate blockchain technology to manage and store the hash addresses of hosted websites securely.

User Registration and Verification: Users can register on the platform and request web hosting services. Admins verify user requests to ensure the legitimacy of web content.

Smart Contract Deployment: We employ smart contracts to facilitate the insertion of IPFS hash addresses into the blockchain, providing transparency and reliability.

Website Deployment: Once approved, user web code is deployed on IPFS, and the resulting hash address is recorded on the blockchain.

**2.5 Modules Description**

Our project consists of the following modules:

User Registration: This module allows users to create accounts on the platform by providing necessary details. Users are required to verify their identities to prevent misuse.

Admin Verification: Admins review and verify user registration requests to maintain the integrity of the platform.

Website Deployment: Users submit their web code to the platform, which is then deployed on IPFS. The resulting IPFS hash address is obtained and stored on the blockchain.

Blockchain Integration: This module manages the blockchain components, including smart contracts, to ensure secure storage of IPFS hash addresses.

Accessing Hosted Websites: Users can access their hosted websites using the provided hash address, ensuring decentralized and efficient web hosting.

**2.6 System Configuration**

The system's configuration includes the following technical aspects:

Server: The system runs on dedicated servers to handle user registrations, admin verifications, and blockchain interactions.

Database: We use a robust database system to store user information, website data, and blockchain records securely.

Blockchain Technology: Our project integrates blockchain technology, which requires suitable blockchain infrastructure and smart contract deployment.

Web Hosting Infrastructure: To host websites on IPFS, the system utilizes IPFS nodes distributed across the network for reliability and redundancy.

User Interfaces: The project includes user interfaces for registration, website submission, and website access. These interfaces are designed to be user-friendly and accessible via web browsers.

Security Measures: The system is equipped with security measures to protect user data, prevent unauthorized access, and ensure the integrity of blockchain records.

Scalability: The project is designed to be scalable, allowing for future enhancements and increased capacity to accommodate a growing user base.

Network Configuration: Proper network configurations are in place to facilitate communication between system components, including servers, databases, and blockchain nodes.

In summary, our project's scope includes web hosting, blockchain integration, user registration and verification, smart contract deployment, and website deployment. The system is configured with the necessary technical components to ensure functionality, security, and scalability.

**3. Literature Overview**

**3. Literature Overview**

**3.1 Papers Referred**

**Title: "IPFS: A New Peer-to-Peer Hypermedia Protocol"**

Description: This paper introduces the InterPlanetary File System (IPFS) and outlines its architecture and design principles. It discusses the challenges of the traditional web, such as centralized servers and broken links, and how IPFS addresses these issues through decentralization and content-addressed storage.

**Title: "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform"**

Description: This paper authored by Ethereum's founder, Vitalik Buterin, presents the Ethereum platform, its capabilities, and its vision for smart contracts and decentralized applications (DApps). It outlines how Ethereum's blockchain technology enables the creation of programmable and self-executing contracts.

**Title: "Bitcoin: A Peer-to-Peer Electronic Cash System"**

Description: Satoshi Nakamoto's seminal paper introduces Bitcoin, the first decentralized cryptocurrency. It describes the need for a trustless electronic payment system, the mechanics of the blockchain, and the process of mining. Bitcoin's impact on the broader blockchain space is profound.

**Title: "Smart Contracts: Building Blocks for Digital Markets"**

Description: This paper explores the concept of smart contracts, their use cases, and their implications for digital markets. It discusses how smart contracts can automate complex processes, reduce intermediaries, and enhance security in various applications.

**Title: "Decentralized Applications: Harnessing Bitcoin's Blockchain Technology"**

Description: This paper delves into the concept of decentralized applications (DApps) and how they leverage blockchain technology. It discusses the features of DApps, their benefits, and provides examples of real-world applications built on blockchain platforms like Ethereum.

These papers form the foundation of the project by providing insights into the problems solved by IPFS, Ethereum, and blockchain technology. They highlight the need for decentralized systems, smart contracts, and DApps in various domains.

**An energy blockchain, a use case on tendermint**

Authors : M.L. Di Silvestre, P. Gallo

The recent advances in distributed energy systems require new models for exchanging energy among prosumers in microgrids. The blockchain technology promises to solve the digital issues related to distributed systems without a trusted authority and to allow quick and secure energy transactions, which are verified and cryptographically protected. Transactions are approved and subsequently recorded on all the machines participating in the blockchain. This work demonstrates how users, which are nodes of the energy and digital networks, exchange energy supported by a customized blockchain based on Tendermint. We focus on the procedures for generating blocks and defining data structures for storing energy transactions.

**Applying blockchain technology to decentralized operation in future energy internet**

Authors : [Tianyu Yang](https://ieeexplore.ieee.org/author/37086157753), [Qinglai Guo](https://ieeexplore.ieee.org/author/37893837600)

This paper presents the potential application of the blockchain technology in future Energy Internet operation, which would be more decentralized and self-executing. A blockchain system could solve several problems in the newly emerged scenarios with the support of the consensus mechanism, encryption methods and validation mechanism. Applying blockchains in decetralized operational framework of energy internet will bring the system with a more secure, flexible and low-cost operational solution.

**A Blockchain-Based Framework for Data Sharing with Fine-grained Access Control in Decentralized Storage Systems**

In traditional cloud storage systems, attribute-based encryption (ABE) is regarded as an important technology for solving the problem of data privacy and fine-grained access control. However, in all ABE schemes, the private key generator (PKG) has the ability to decrypt all data stored in the cloud server, which may bring serious problems such as key abuse and privacy data leakage. Meanwhile, the traditional cloud storage model runs in a centralized storage manner, so single point of failure may lead to the collapse of system. With the development of blockchain technology, decentralized storage mode have entered the public view. The decentralized storage approach can solve the problem of single point of failure in traditional cloud storage systems and enjoy a number of advantages over centralized storage, such as low price and high throughput. In this paper, we study the data storage and sharing scheme for decentralized storage systems, and propose a framework that combines the decentralized storage system IPFS, the Ethereum blockchain and attribute-based encryption (ABE) technology. In this framework, the data owner has the ability to distribute secret key for data users, and encrypt shared data by specifying access policy, and the scheme achieves fine-grained access control over data. At the same time, based on smart contract on the Ethereum blockchain, the keyword search function on the ciphertext of the decentralized storage systems is implemented, which solves the problem that the cloud server may not return all of the results searched or return wrong results in the traditional cloud storage systems. Finally, we simulated the scheme in the linux system and the Ethereum official test network Rinkeby, and the experimental results show that our scheme is feasible

**Smart contract-based campus demonstration of decentralized transactive energy auctions**

Authors : [Adam Hahn](https://ieeexplore.ieee.org/author/37604656700), [Rajveer Singh](https://ieeexplore.ieee.org/author/37086250255)

Transactive energy paradigms will enable the exchange of energy from a distributed set of prosumers. While prosumers have access to distributed energy resources, these resources are intermittently available. There is a need for distributed markets to enable the exchange of energy in transactive environments, however, the large number of potential prosumers introduces challenges in the establishment of trust between prosumers. Markets for transactive environments create other challenges, such as establishing clearing prices for energy and exchanging money between prosumers. Blockchains provide a unique technology to address this distributed trust problem through the use of a distributed ledger, cryptocurrencies, and the execution of smart contracts. This paper introduces a smart contract that implements a transactive energy auction that operates without the need for a trusted entitys oversight. The auction mechanism implements a Vickrey second price auction, which guarantees bidders will submit honest bids. The contract is implemented on transactive agents on the WSU campus interacting with a 72kW PV array and the Ethereum blockchain. The contract is then used to execute auctions based on the energy from the the PV array and simulated building loads to demonstrate the auctions operations.

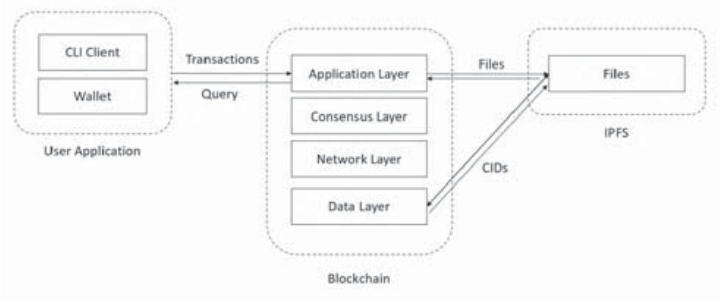
**Decentralizing privacy: using blockchain to protect personal data**

[**Guy Zyskind**](https://ieeexplore.ieee.org/author/37085542489)**;**[**Oz Nathan**](https://ieeexplore.ieee.org/author/37085589557)

The recent increase in reported incidents of surveillance and security breaches compromising users’ privacy call into question the current model, in which third-parties collect and control massive amounts of personal data. Bitcoin has demonstrated in the financial space that trusted, auditable computing is possible using a decentralized network of peers accompanied by a public ledger. In this paper, we describe a decentralized personal data management system that ensures users own and control their data. We implement a protocol that turns a blockchain into an automated access-control manager that does not require trust in a third party. Unlike Bitcoin, transactions in our system are not strictly financial – they are used to carry instructions, such as storing, querying and sharing data. Finally, we discuss possible future extensions to blockchains that could harness them into a well-rounded solution for trusted computing problems in society.

**4. SYSTEM DESIGN**

**4.1 System Architecture**



4.1.1 Module Description

**IPFS data storage:** IPFS is a data storage server and we can upload any post related image and this images will be saved in IPFS server and each image will have one hashcode as its address. This address will be stored in Blockchain whine retrieving POST data.

**Signup Module:** using this module user can signup with application and all signup details will be saved in Blockchain

**Login Module:** using this module user can login to application and after login will perform below operations

**Publish & Save Tweets in Blockchain**: using this module user can post and upload image to application and application will saved post data to Blockchain and image data to IPFS server

View Tweets: using this module all users can see tweets of one and other by retrieving from Blockchain.

**View Tweets:** using this module all users can see tweets of one and other by retrieving from Blockchain.

**4.2 System Design**

**UML DIAGRAMS**

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

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

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

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

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

**GOALS:**

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

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

**USE CASE DIAGRAM**

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



USECASE DIAGRAM

**CLASS DIAGRAM**

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



CLASS DIAGRAM

**SEQUENCE DIAGRAM**

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

****

SEQUENCE DIAGRAM

**COLLABORATION DIAGRAM**

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.



COLLABORATION DIAGRAM

**5. IMPLEMENTATION**

**5. Implementation**

**5.1 Project Explanation**

The project aims to provide decentralized web hosting using the InterPlanetary File System (IPFS) and manage the hash addresses of IPFS deployments on a blockchain. Below is an explanation of the project's implementation:

Decentralized Web Hosting:

IPFS Integration: The project utilizes IPFS for web hosting. Users can send their website code to the admin for hosting. The admin verifies the code and deploys it on IPFS, generating a unique hash address for the website.

Blockchain Integration: To maintain the integrity and availability of IPFS data, a blockchain is used. The blockchain stores the hash addresses of the IPFS deployments. This ensures that even if one IPFS node goes down, the data remains accessible through other working nodes.

Immutable Data: Blockchain technology is known for its immutability. When a new record is added to the blockchain (in this case, a new website deployment), the blockchain verifies the hash code of all previous blocks at all nodes. If any data has been tampered with, the verification fails, ensuring data cannot be altered in the blockchain.

User Registration and Website Deployment:

User Signup: Users register with the application by providing their details. This information is securely stored.

User Requests Hosting: After registration, users can send a request to the admin for web hosting. They provide their website code for review.

Admin Verification: The admin verifies the website code to ensure it complies with hosting guidelines. If approved, the admin deploys the code on IPFS and generates a hash address.

Blockchain Record: The admin inserts the hash address of the deployed website into the blockchain using a smart contract.

Website Access: Once the code is deployed, users can access their website by using the provided hash code.

Implementation Steps:

Environment Setup: Install the necessary software components, including Node.js, Python packages (Django and web3), and run the Ethereum tool for setting up the blockchain environment.

Smart Contract Deployment: Use Truffle to deploy the smart contract to the blockchain. The contract allows storing user signup and hosting details.

IPFS Server: Start the IPFS server to enable web hosting via IPFS.

DJANGO Server: Run the Django server, which acts as the application backend.

User Interaction: Users interact with the web application through the Django-based user interface. They can sign up, request hosting, and access their hosted websites.

Admin Approval: Admins review and approve hosting requests. If necessary, they can upload the website code to IPFS and the blockchain.

Accessing Deployed Websites: Users can access their deployed websites by using the provided hash codes.

Benefits and Impact:

Decentralization: The project promotes decentralization, ensuring data availability even if some nodes go down.

Immutable Records: Blockchain ensures that hosted website data remains unchanged and secure.

User-Friendly: Users can easily register, request hosting, and access their websites through a simple interface.

Future Enhancements:

Scalability: The project can be scaled to host multiple websites efficiently.

Enhanced Security: Implement additional security measures to protect user data and code.

Monetization: Integrate payment options for premium hosting services.

This project demonstrates the potential of IPFS and blockchain for decentralized web hosting, offering users a reliable and secure hosting solution.

**5.2 Sample code**

Manage.py Code:

#!/usr/bin/env python

import os

import sys

if \_\_name\_\_ == '\_\_main\_\_':

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'Webhosting.settings')

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you "

"forget to activate a virtual environment?"

) from exc

execute\_from\_command\_line(sys.argv)

'''

import ipfsApi

api = ipfsApi.Client('127.0.0.1', 5001)

res = api.add('Website')

print(res)

'''

import subprocess

out = subprocess.run('ipfs add -r Website', stdout = subprocess.PIPE, shell=True)

msg = out.stdout.splitlines()

print(msg)

print("======")

output = msg[len(msg)-1]

output = output.decode()

arr = output.split(" ")

print(arr)

from django.urls import path

from . import views

urlpatterns = [path("index.html", views.index, name="index"),

path("UserLogin.html", views.UserLogin, name="UserLogin"),

path("AdminLogin.html", views.AdminLogin, name="AdminLogin"),

path("Signup.html", views.Signup, name="Signup"),

path("SignupAction", views.SignupAction, name="SignupAction"), path("SendRequest.html", views.SendRequest, name="SendRequest"),

path("SendRequestAction", views.SendRequestAction, name="SendRequestAction"),

path("ViewHostAddress", views.ViewHostAddress, name="ViewHostAddress"),

path("ViewRequest", views.ViewRequest, name="ViewRequest"),

path("UploadWebsite", views.UploadWebsite, name="UploadWebsite"),

path("UploadWebsiteAction", views.UploadWebsiteAction, name="UploadWebsiteAction"),

path("AdminLoginAction", views.AdminLoginAction, name="AdminLoginAction"),

path("UserLoginAction.html", views.UserLoginAction, name="UserLoginAction"),

path("ViewUsers", views.ViewUsers, name="ViewUsers"),

]

from django.shortcuts import render

from django.template import RequestContext

from django.contrib import messages

import pymysql

from django.http import HttpResponse

from django.core.files.storage import FileSystemStorage

import os

import datetime

import os

import json

from web3 import Web3, HTTPProvider

import subprocess

global userid

def readDetails(contract\_type):

global details

details = ""

print(contract\_type+"======================")

blockchain\_address = 'http://127.0.0.1:9545' #Blokchain connection IP

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'WebHosting.json' #web hosting contract code

deployed\_contract\_address = '0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058' #hash address to access EHR contract

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract = web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi) #now calling contract to access data

if contract\_type == 'signup':

details = contract.functions.getSignup().call() #call getHospital function to access all hospital details

if contract\_type == 'hosting':

details = contract.functions.getHosting().call()

print(details)

def saveDataBlockChain(currentData, contract\_type):

global details

global contract

details = ""

blockchain\_address = 'http://127.0.0.1:9545'

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'WebHosting.json' #web hosting funding contract file

deployed\_contract\_address = '0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058' #contract address

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract = web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi)

readDetails(contract\_type)

if contract\_type == 'signup':

details+=currentData

msg = contract.functions.setSignup(details).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

if contract\_type == 'hosting':

details+=currentData

msg = contract.functions.setHosting(details).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

def updateBlockChain(currentData):

global details

global contract

details = ""

blockchain\_address = 'http://127.0.0.1:9545'

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'WebHosting.json' #crowd funding contract file

deployed\_contract\_address = '0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058' #contract address

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract = web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi)

msg = contract.functions.setHosting(currentData).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

def ViewUsers(request):

if request.method == 'GET':

global userid

readDetails("signup")

rows = details.split("\n")

output = ""

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "signup":

output+='<tr><td><font size="" color="black">'+str(arr[1])+'</td>'

output+='<td><font size="" color="black">'+str(arr[2])+'</td>'

output+='<td><font size="" color="black">'+str(arr[3])+'</td>'

output+='<td><font size="" color="black">'+str(arr[4])+'</td>'

output+='<td><font size="" color="black">'+str(arr[5])+'</td>'

context= {'data':output}

return render(request, 'ViewUsers.html', context)

def UploadWebsite(request):

if request.method == 'GET':

global userid

user = request.GET.get('user', False)

pdate = request.GET.get('pdate', False)

output = '<tr><td><font size="" color="black">Requester&nbsp;Name</b></td><td>'

output += '<input type="text" name="t1" style="font-family: Comic Sans MS" size="20" value="'+user+'"/></td></tr>'

output += '<tr><td><font size="" color="black">Request&nbsp;Date</b></td><td>'

output += '<input type="text" name="t2" size="20" style="font-family: Comic Sans MS" value="'+pdate+'"/></td></tr>'

output += '<tr><td><font size="" color="black">Upload&nbsp;Website&nbsp;Folder</b></td><td>'

output += '<input type="text" name="t3" size="40" style="font-family: Comic Sans MS"/></td></tr>'

context= {'data1':output}

return render(request, 'UploadWebsite.html', context)

def UploadWebsiteAction(request):

if request.method == 'POST':

global userid

name = request.POST.get('t1', False)

pdate = request.POST.get('t2', False)

path = request.POST.get('t3', False)

out = subprocess.run('ipfs add -r '+path, stdout = subprocess.PIPE, shell=True)

msg = out.stdout.splitlines()

output = msg[len(msg)-1]

output = output.decode()

arr = output.split(" ")

print(arr)

hashcode = arr[1].strip()

print(hashcode)

readDetails("hosting")

rows = details.split("\n")

output = ""

for i in range(len(rows)-1):

arr = rows[i].split("#")

print(arr[1]+"====="+arr[4]+"===="+name+" "+pdate)

if arr[0] == "hosting" and arr[4] != pdate:

output += rows[i]+"\n"

elif arr[0] == "hosting" and arr[4] == pdate:

data = "hosting#"+arr[1]+"#"+arr[2]+"#"+arr[3]+"#"+arr[4]+"#"+hashcode+"\n"

output += data

updateBlockChain(output)

context= {'data':'Website uploaded with URL http://localhost:8080/ipfs/'+hashcode}

return render(request, 'AdminScreen.html', context)

def SendRequestAction(request):

if request.method == 'POST':

global userid

name = request.POST.get('t1', False)

desc = request.POST.get('t2', False)

today = datetime.datetime.now()

data = "hosting#"+userid+"#"+name+"#"+desc+"#"+str(today)+"#Pending\n"

saveDataBlockChain(data,"hosting")

context= {'data':'Your request sent to admin'}

return render(request, 'SendRequest.html', context)

def ViewRequest(request):

if request.method == 'GET':

global userid

readDetails("hosting")

rows = details.split("\n")

output = ""

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "hosting":

output+='<tr><td><font size="" color="black">'+str(arr[1])+'</td>'

output+='<td><font size="" color="black">'+str(arr[2])+'</td>'

output+='<td><font size="" color="black">'+str(arr[3])+'</td>'

output+='<td><font size="" color="black">'+str(arr[4])+'</td>'

if arr[5] != 'Pending':

output+='<td><font size="" color="black">'+str(arr[5])+'</td>'

else:

output+='<td><a href=\'UploadWebsite?user='+arr[1]+'&pdate='+arr[4]+'\'><font size=3 color=black>Click Here</font></a></td></tr>'

context= {'data':output}

return render(request, 'ViewRequest.html', context)

def ViewHostAddress(request):

if request.method == 'GET':

global userid

readDetails("hosting")

rows = details.split("\n")

output = ""

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "hosting" and arr[1] == userid:

output+='<tr><td><font size="" color="black">'+str(arr[1])+'</td>'

output+='<td><font size="" color="black">'+str(arr[2])+'</td>'

output+='<td><font size="" color="black">'+str(arr[3])+'</td>'

output+='<td><font size="" color="black">'+str(arr[4])+'</td>'

if arr[5] != 'Pending':

output+='<td><font size="" color="black"><a href=http://localhost:8080/ipfs/'+str(arr[5])+'>'+str(arr[5])+'</td>'

else:

output+='<td><font size="" color="black">'+str(arr[5])+'</td>'

context= {'data':output}

return render(request, 'ViewHostAddress.html', context)

def SendRequest(request):

if request.method == 'GET':

return render(request, 'SendRequest.html', {})

def index(request):

if request.method == 'GET':

return render(request, 'index.html', {})

def UserLogin(request):

if request.method == 'GET':

return render(request, 'UserLogin.html', {})

def Signup(request):

if request.method == 'GET':

return render(request, 'Signup.html', {})

def AdminLogin(request):

if request.method == 'GET':

return render(request, 'AdminLogin.html', {})

def AdminLoginAction(request):

if request.method == 'POST':

global userid

user = request.POST.get('t1', False)

password = request.POST.get('t2', False)

if user == "admin" and password == "admin":

context= {'data':'Welcome '+user}

return render(request, 'AdminScreen.html', context)

else:

context= {'data':'Invalid Login'}

return render(request, 'AdminLogin.html', context)

def SignupAction(request):

if request.method == 'POST':

user = request.POST.get('t1', False)

password = request.POST.get('t2', False)

email = request.POST.get('t3', False)

contact = request.POST.get('t4', False)

address = request.POST.get('t5', False)

record = 'none'

readDetails("signup")

rows = details.split("\n")

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "signup":

if arr[1] == user:

record = "exists"

break

if record == 'none':

data = "signup#"+user+"#"+password+"#"+contact+"#"+email+"#"+address+"\n"

saveDataBlockChain(data,"signup")

context= {'data':'Signup process completd and record saved in Blockchain'}

return render(request, 'Signup.html', context)

else:

context= {'data':user+'Username already exists'}

return render(request, 'Signup.html', context)

def UserLoginAction(request):

if request.method == 'POST':

global userid

user = request.POST.get('t1', False)

password = request.POST.get('t2', False)

status = 'none'

readDetails("signup")

rows = details.split("\n")

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "signup":

if arr[1] == user and arr[2] == password:

status = 'success'

userid = user

break

if status == 'success':

file = open('session.txt','w')

file.write(user)

file.close()

context= {'data':"Welcome "+user}

return render(request, 'UserScreen.html', context)

else:

context= {'data':'Invalid login details'}

return render(request, 'UserLogin.html', context)

**6. TESTING**

**6.1 Testing**

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

**System Testing and Implementation**

The purpose is to exercise the different parts of the module code to detect coding errors. After this the modules are gradually integrated into subsystems, which are then integrated themselves too eventually forming the entire system. During integration of module integration testing is performed. The goal of this is to detect designing errors, while focusing the interconnection between modules. After the system was put together, system testing is performed. Here the system is tested against the system requirements to see if all requirements were met and the system performs as specified by the requirements. Finally accepting testing is performed to demonstrate to the client for the operation of the system.

For the testing to be successful, proper selection of the test case is essential. There are two different approaches for selecting test case. The software or the module to be tested is treated as a black box, and the test cases are decided based on the specifications of the system or module. For this reason, this form of testing is also called “black box testing”.

The focus here is on testing the external behavior of the system. In structural testing the test cases are decided based on the logic of the module to be tested. A common approach here is to achieve some type of coverage of the statements in the code. The two forms of testing are complementary: one tests the external behavior, the other tests the internal structure.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process. Frequently the testing process starts with the test plan. This plan identifies all testing related activities that must be performed and specifies the schedule, allocates the resources, and specifies guidelines for testing. The test plan specifies conditions that should be tested; different units to be tested, and the manner in which the module will be integrated together. Then for different test unit, a test case specification document is produced, which lists all the different test cases, together with the expected outputs, that will be used for testing. During the testing of the unit the specified test cases are executed and the actual results are compared with the expected outputs. The final output of the testing phase is the testing report and the error report, or a set of such reports. Each test report contains a set of test cases and the result of executing the code with the test cases. The error report describes the errors encountered and the action taken to remove the error.

**Testing Techniques**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**Black Box Testing**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

**White Box Testing**

In this testing, the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

1. Guarantee that all independent paths have been executed.
2. Execute all logical decisions on their true and false sides
3. Execute all loops at their boundaries and within their operational
4. Execute internal data structures to ensure their validity.

**Testing Strategies**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

This System consists of 3 modules. Those are Reputation module, route discovery module, audit module. Each module is taken as unit and tested. Identified errors are corrected and executable unit are obtained.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**Functional Testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

**Functional testing is centered on the following items**

Valid Input : identified classes of valid input must be accepted

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes.

**6.2 Test Cases**

Which involves various components such as IPFS, a blockchain, and a web application, can be a complex task. Test cases are essential to ensure that the system works correctly and reliably. Here's an outline of test cases for this decentralized web hosting project:

1. User Registration and Authentication:

* Test Case 1.1: Verify that a new user can successfully register with valid details.
* Test Case 1.2: Ensure that duplicate user registrations with the same email are not allowed.
* Test Case 1.3: Validate that a user can log in with the correct credentials.
* Test Case 1.4: Verify that a user cannot log in with incorrect credentials.

2. User Requests and Admin Approval:

* Test Case 2.1: Check that a user can submit a hosting request with valid website details.
* Test Case 2.2: Verify that an admin can view pending user requests.
* Test Case 2.3: Ensure that an admin can approve a user request for hosting.
* Test Case 2.4: Check that an admin can reject a user request for hosting.

3. Website Deployment:

* Test Case 3.1: Confirm that the admin can upload the website code to IPFS and the blockchain.
* Test Case 3.2: Ensure that a unique hash address is generated for each deployed website.
* Test Case 3.3: Validate that the deployed website is accessible using the generated hash address.

4. Data Integrity:

* Test Case 4.1: Check that the blockchain maintains the integrity of hash addresses.
* Test Case 4.2: Verify that data in IPFS remains unchanged after deployment.
* Test Case 4.3: Test the blockchain's ability to detect and prevent unauthorized changes to data.

5. Decentralization and Redundancy:

* Test Case 5.1: Ensure that the system remains operational even if one IPFS node goes down.
* Test Case 5.2: Verify that users can access their websites through other working IPFS nodes.

6. System Scalability:

* Test Case 6.1: Test the system's ability to host multiple websites concurrently.
* Test Case 6.2: Verify that the system performance remains acceptable as the number of hosted websites increases.

7. Security and Privacy:

* Test Case 7.1: Confirm that user data is securely stored and not exposed to unauthorized users.
* Test Case 7.2: Verify that login credentials are hashed and securely stored.
* Test Case 7.3: Test the system for vulnerabilities like SQL injection, cross-site scripting, and cross-site request forgery.

8. Cost and Resource Management:

* Test Case 8.1: Check the cost-effectiveness of deploying and maintaining websites on IPFS and the blockchain.
* Test Case 8.2: Ensure that the system optimizes resource usage for efficient hosting.

9. Error Handling:

* Test Case 9.1: Validate that appropriate error messages are displayed to users for different scenarios (e.g., failed login, invalid inputs).
* Test Case 9.2: Test the system's resilience to unexpected errors and exceptions.

10. User Interface:

* Test Case 10.1: Verify that the user interface is user-friendly and intuitive.
* Test Case 10.2: Check the responsiveness and compatibility of the web application with various browsers.

11. Load Testing:

* Test Case 11.1: Perform load testing to determine how the system handles a large number of concurrent users and requests.

12. Performance Testing:

* Test Case 12.1: Measure the system's response time when accessing hosted websites and uploading code.

13. Cross-Browser Compatibility:

* Test Case 13.1: Ensure that the web application works correctly on different web browsers (e.g., Chrome, Firefox, Edge).

14. Mobile Responsiveness:

* Test Case 14.1: Confirm that the web application is responsive and usable on mobile devices.

15. Accessibility Testing:

* Test Case 15.1: Validate that the web application is accessible to users with disabilities, adhering to accessibility standards (e.g., WCAG).

**7. OUTPUT SCREENS**

A Blockchain based Autonomous Decentralized Online Social Network

Now-a-days 80% peoples are using online social networks to post their opinions or to get news information or to keep himself in touch with friends and relatives. All social networks applications are using centralized servers to store user’s signup and post details and if that server crash then services to users will be disturbed and data will be lost. If any attacker hack server then all users’ data security will be at risk as attackers may misuse all users’ data.

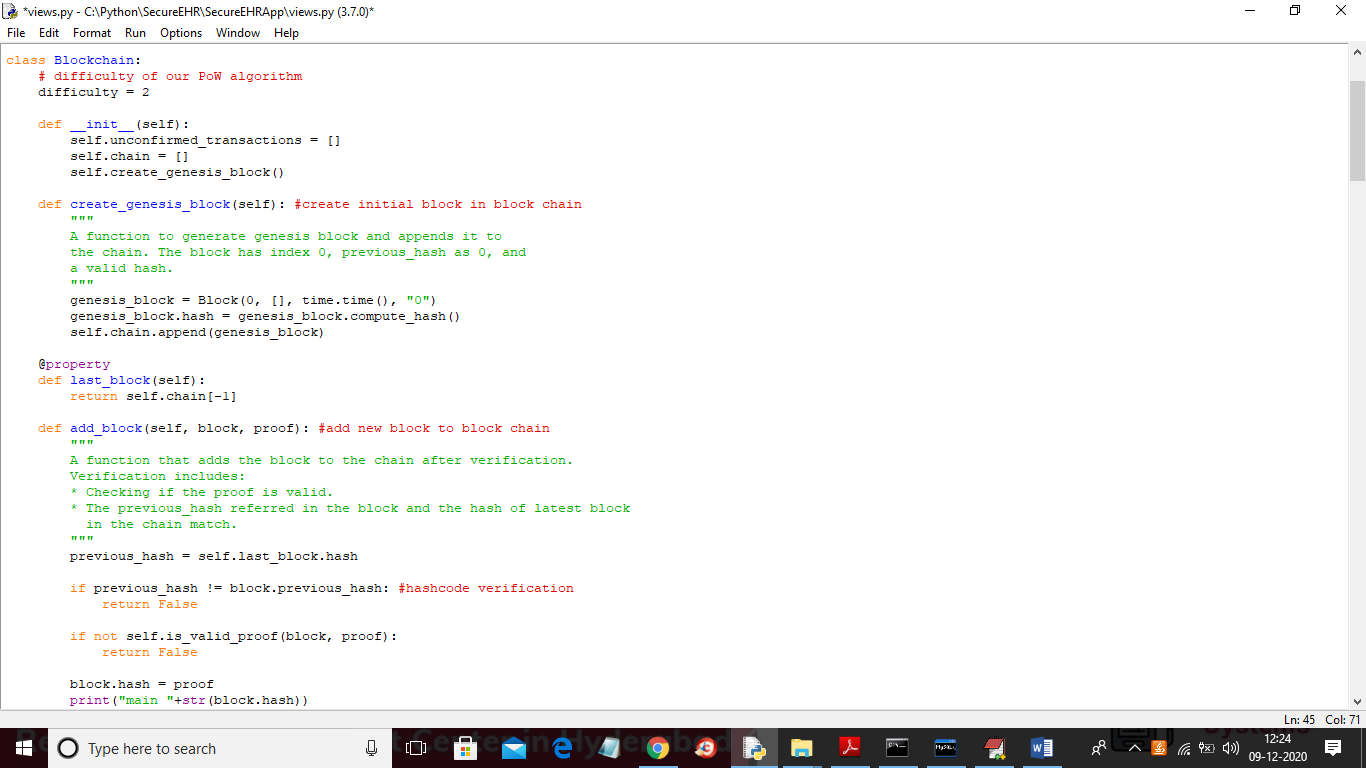
To overcome from above issues author of this paper are suggesting concept to divert all social media application to decentralized (data will be maintained at multiple nodes or servers) Blockchain servers. Blockchain will store each data/transaction as block and before storing any new block, Blockchain will verify all previous blocks hashcode in all running nodes and if all block hashcodes verified then only new block will be added and if any node verification failed then data will be gathered from other working nodes and attacked system get recovered. Blockchain never allowed any attacker to modify blocks so it will consider as immutable.

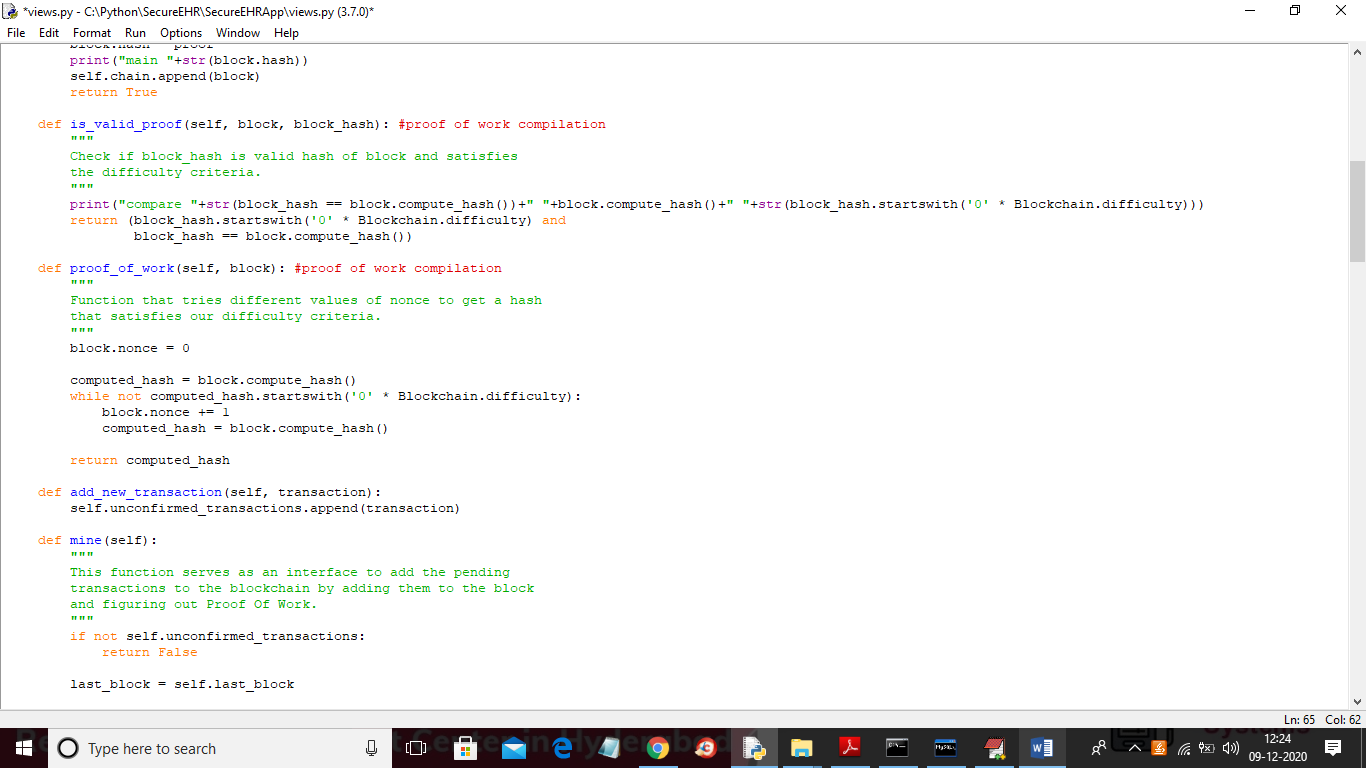
So by applying Blockchain OSN networks can maintain data at multiple nodes and avoid risk of server crashing and hacking. In propose paper author using Blockchain to store social media post and all heavy content such as videos and images will be saved at IPFS (inter planetary file system) server as Blockchain is not suitable to store heavy size data.

Due to this transaction/block hash code verification and immutable data storage make Blockchain secure and trustable in current market.

Below is the coding to store data in Blockchain

See comments in below code screen shots to understand Blockchain working procedure





In above two screen read comments to understand block chain working procedure.

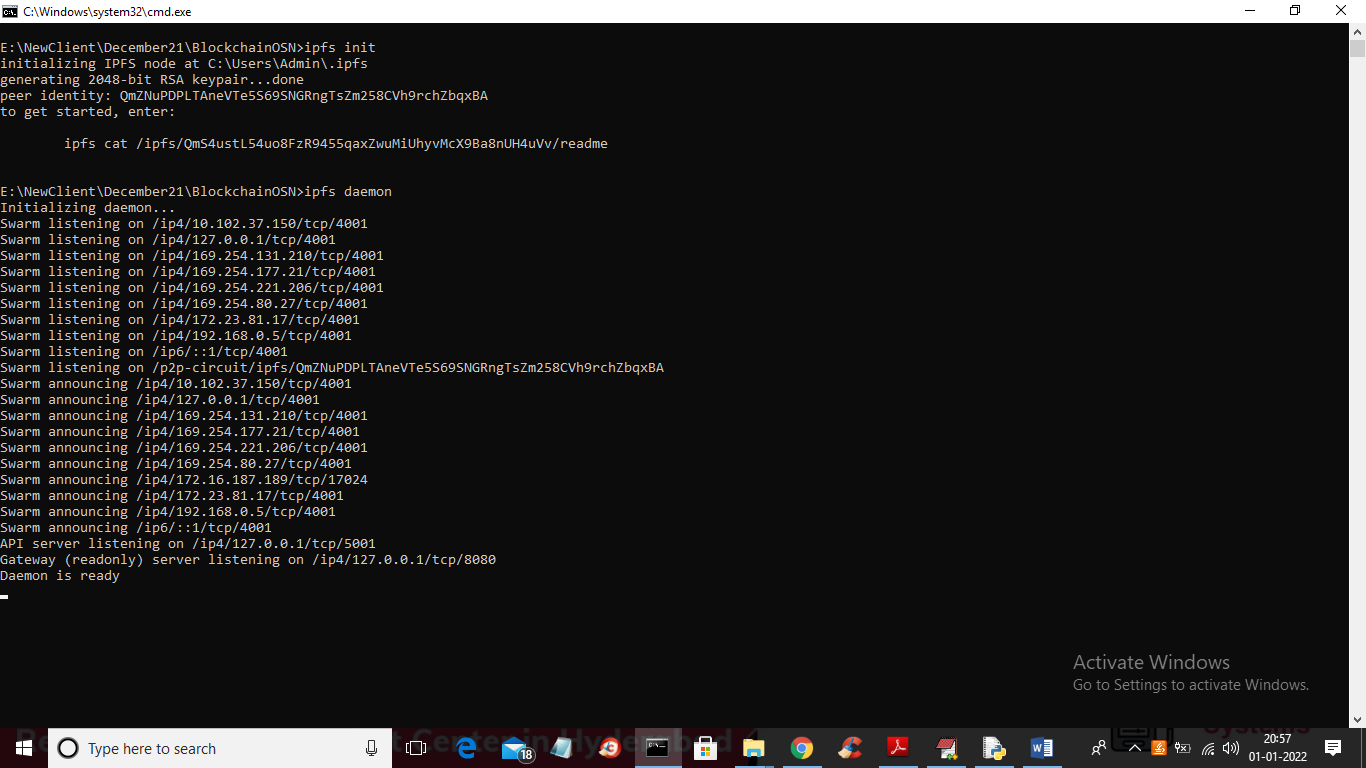
In this project author is saying about online social network so we need to build this application as WEB so we used python and its DJANGO framework.

To implement this project we have designed following modules

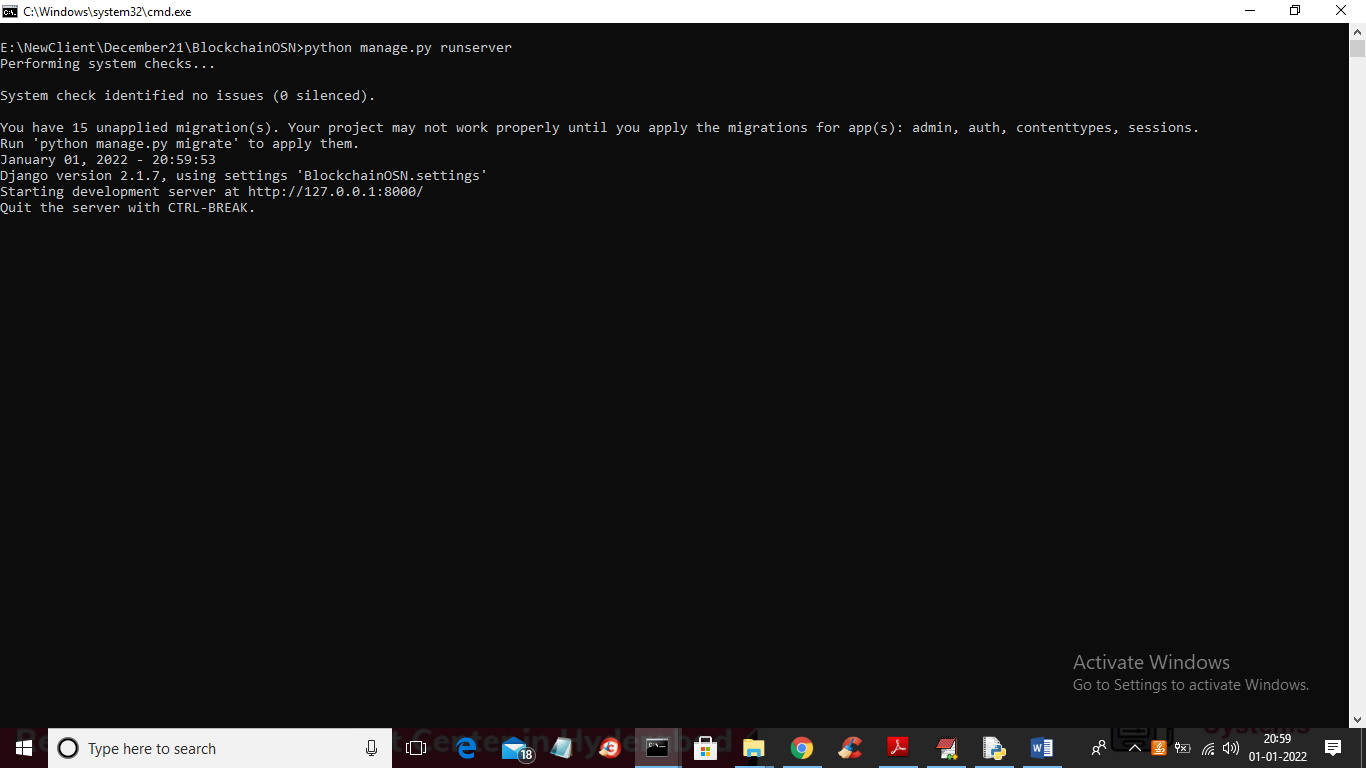
1. IPFS data storage: IPFS is a data storage server and we can upload any post related image and this images will be saved in IPFS server and each image will have one hashcode as its address. This address will be stored in Blockchain whine retrieving POST data.
2. Signup Module: using this module user can signup with application and all signup details will be saved in Blockchain
3. Login Module: using this module user can login to application and after login will perform below operations
4. Publish & Save Tweets in Blockchain: using this module user can post and upload image to application and application will saved post data to Blockchain and image data to IPFS server
5. View Tweets: using this module all users can see tweets of one and other by retrieving from Blockchain.

SCREEN SHOTS

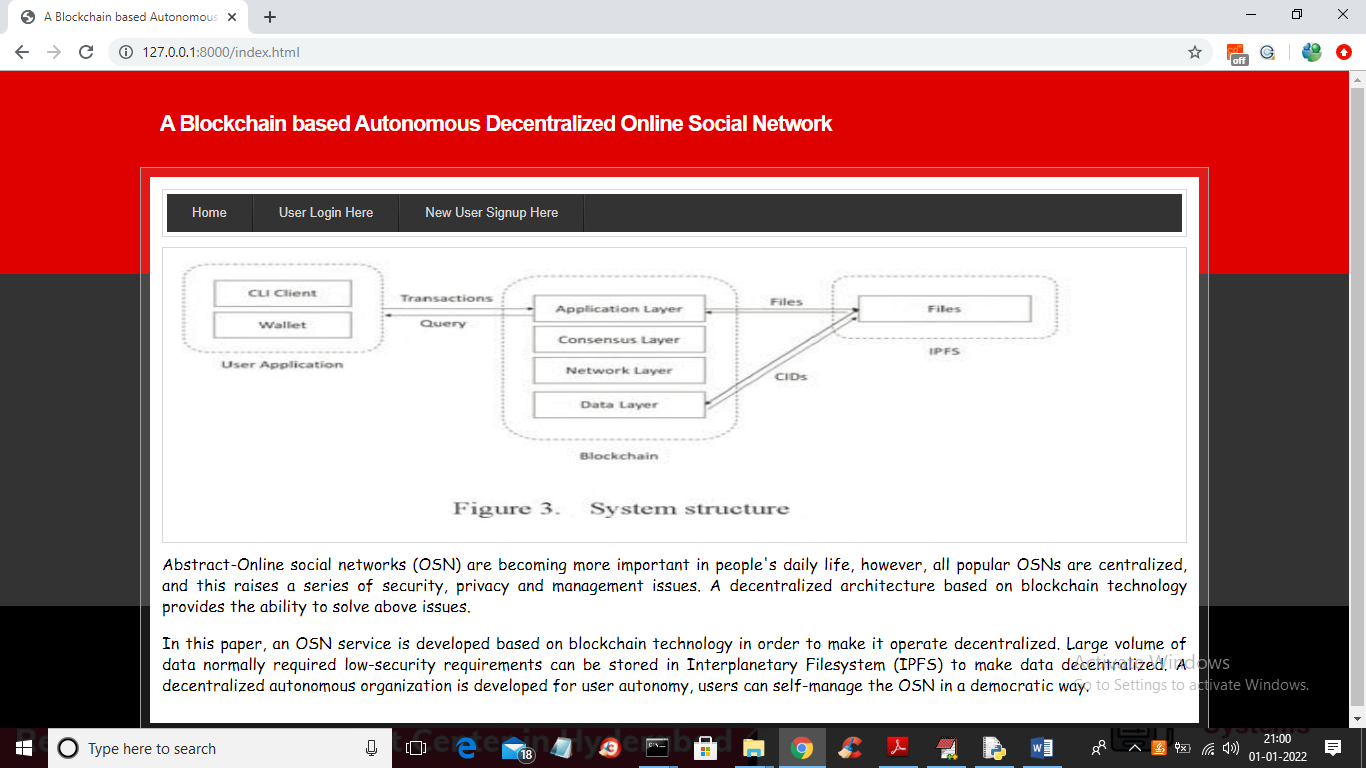
To run project first double click on ‘Start\_IPFS.bat’ file to start IPFS server and to get below screen



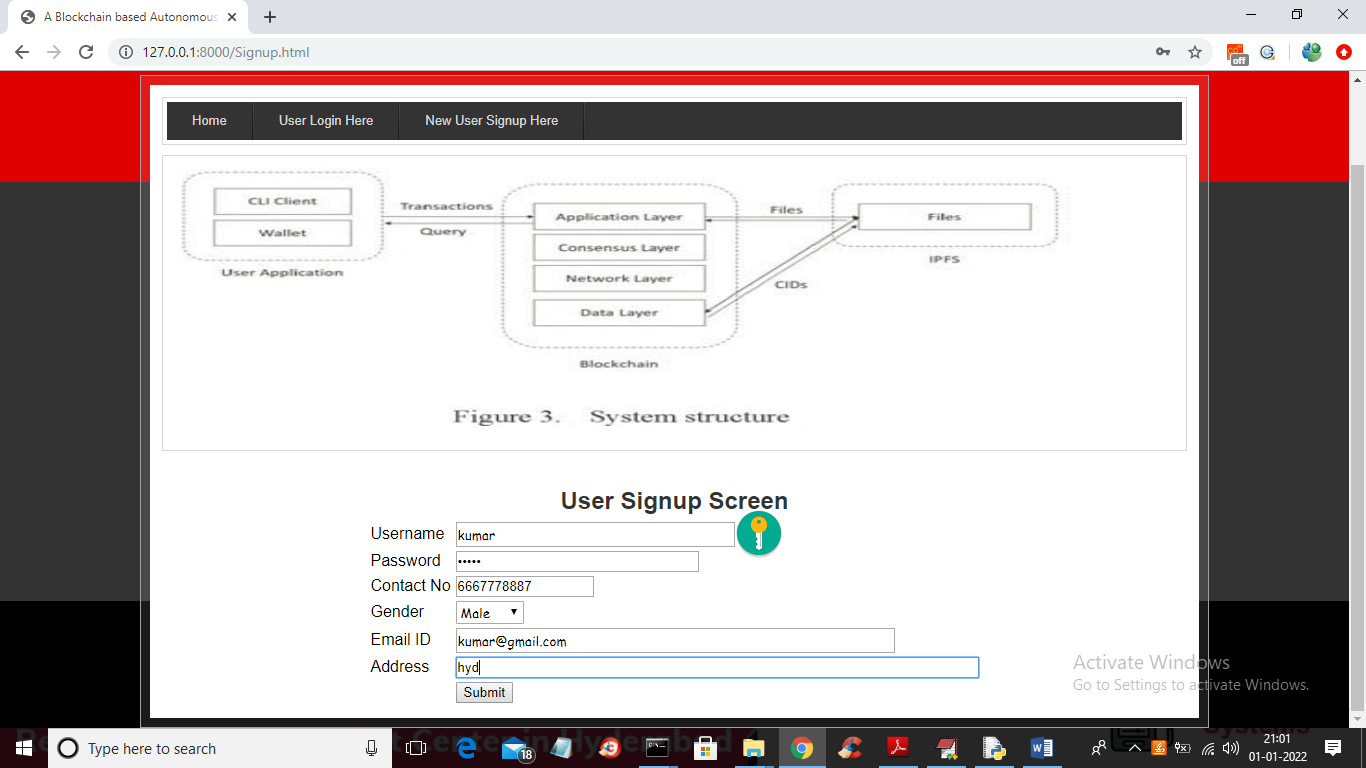
In above screen IPFS server started and now double click on ‘runServer.bat’ file to start DJANGO server and to get below screen



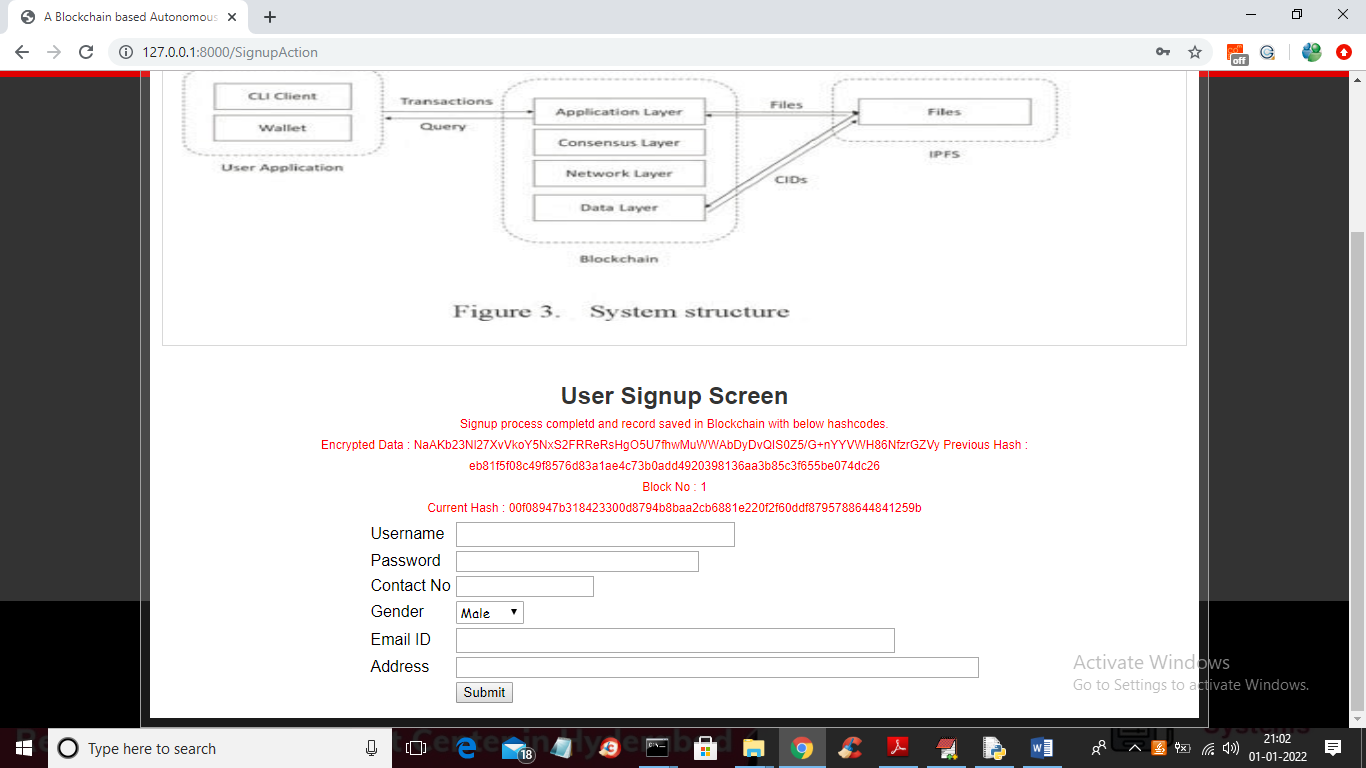
In above screen python DJANGO server started and now open browser and enter URL as <http://127.0.0.1:8000/index.html> and press enter key to get below home page



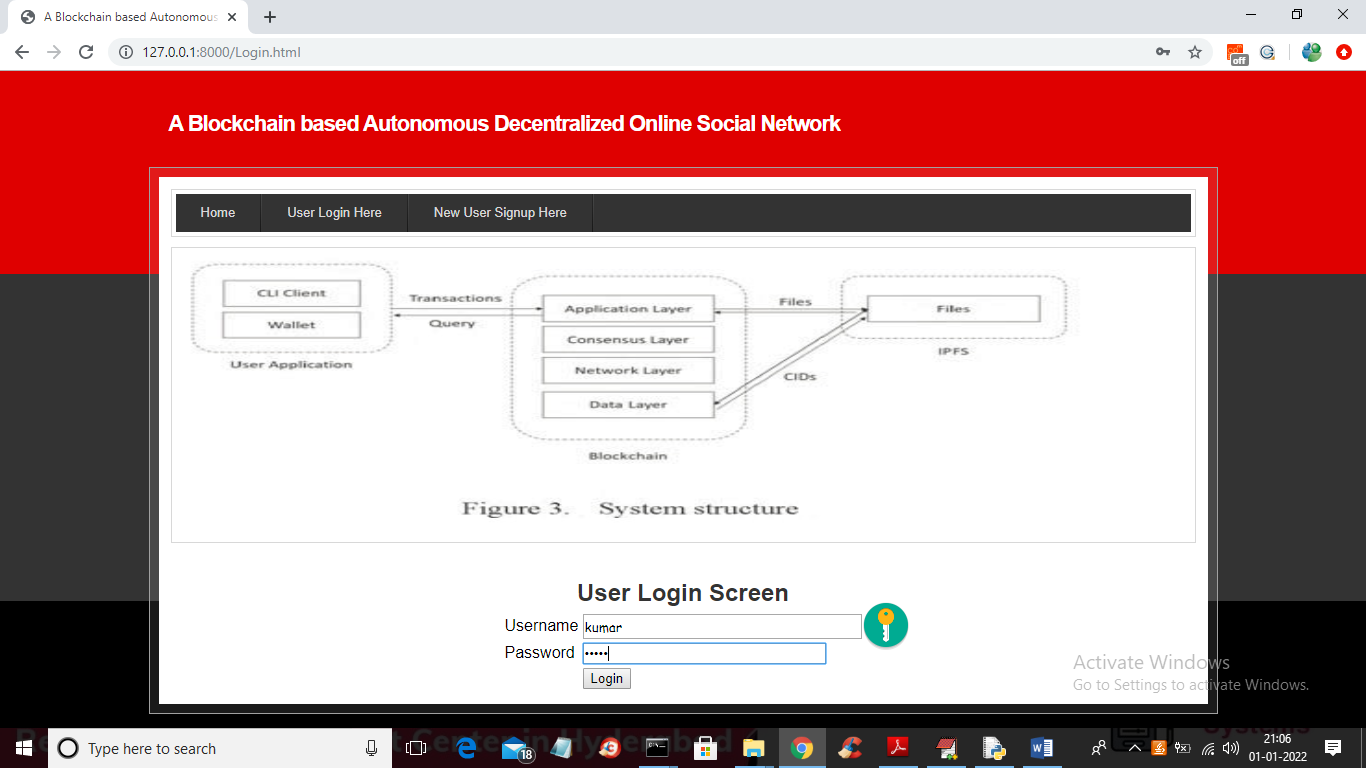
In above screen click on ‘New User Signup Here’ link to get below signup screen



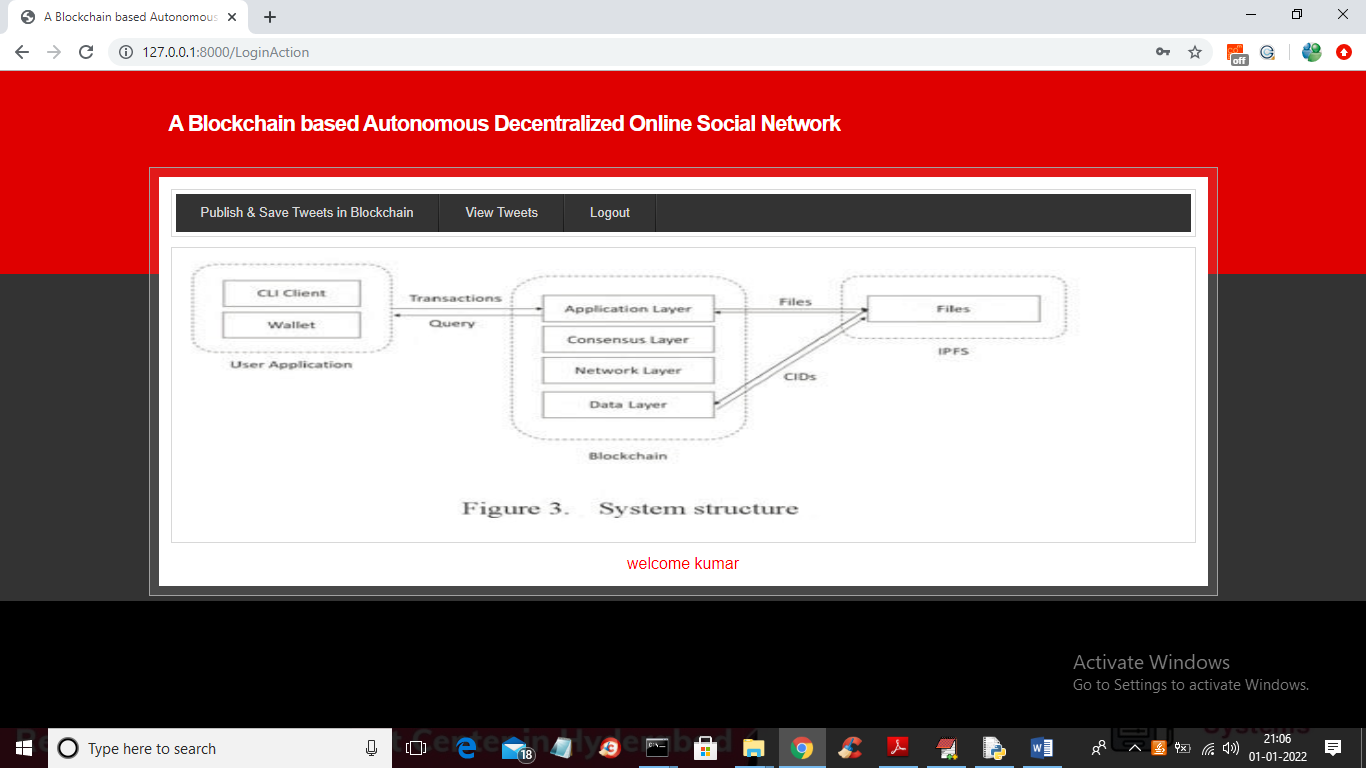
In above screen entering signup details and then press ‘Submit’ button to get below screen



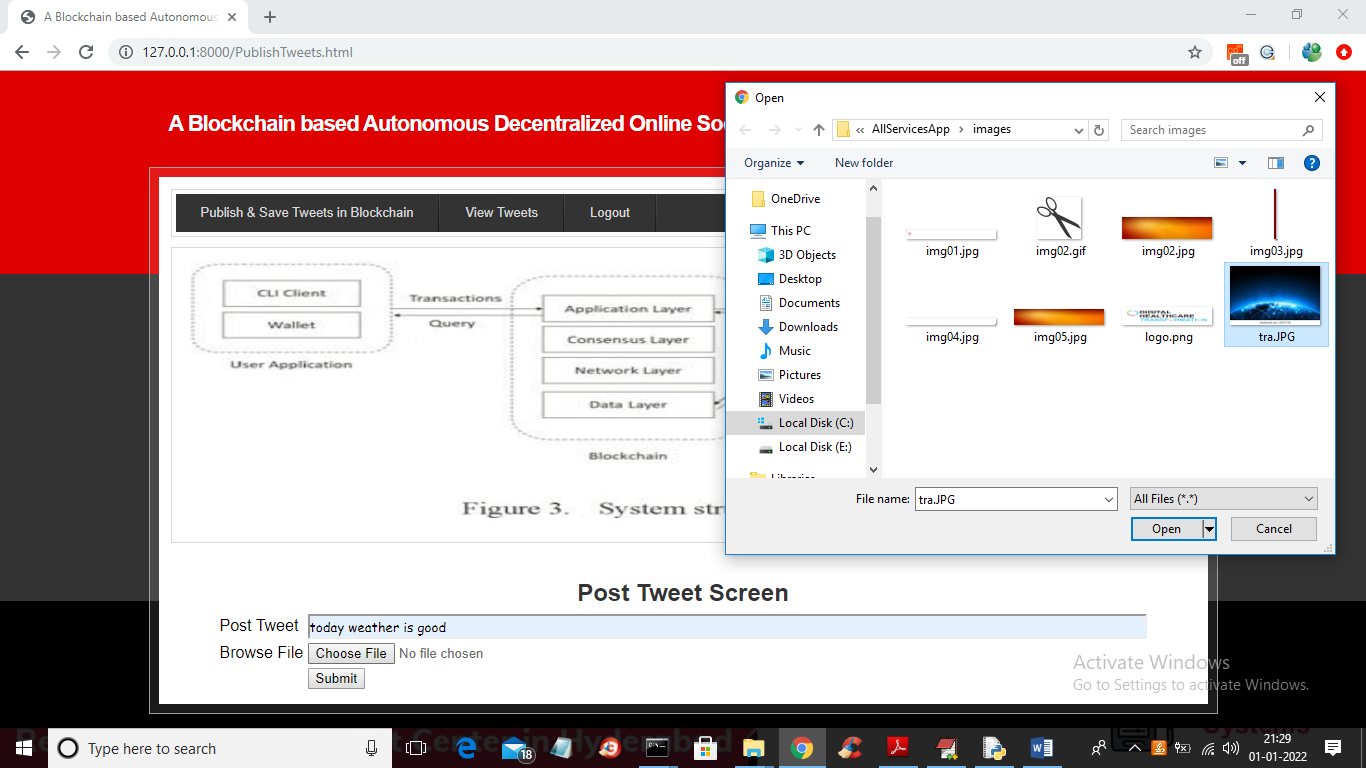
In above screen in red colour text we can see signup process completed and we got all Blockchain HASHCODES address where this signup data store and we got this data block no as 1. For each data storage we can see hash code of current record will match with previous hash code of next records and Blockchain will form chain of blocks with new and previous hashcodes for verification and now click on ‘User Login Here’ link to get below login screen



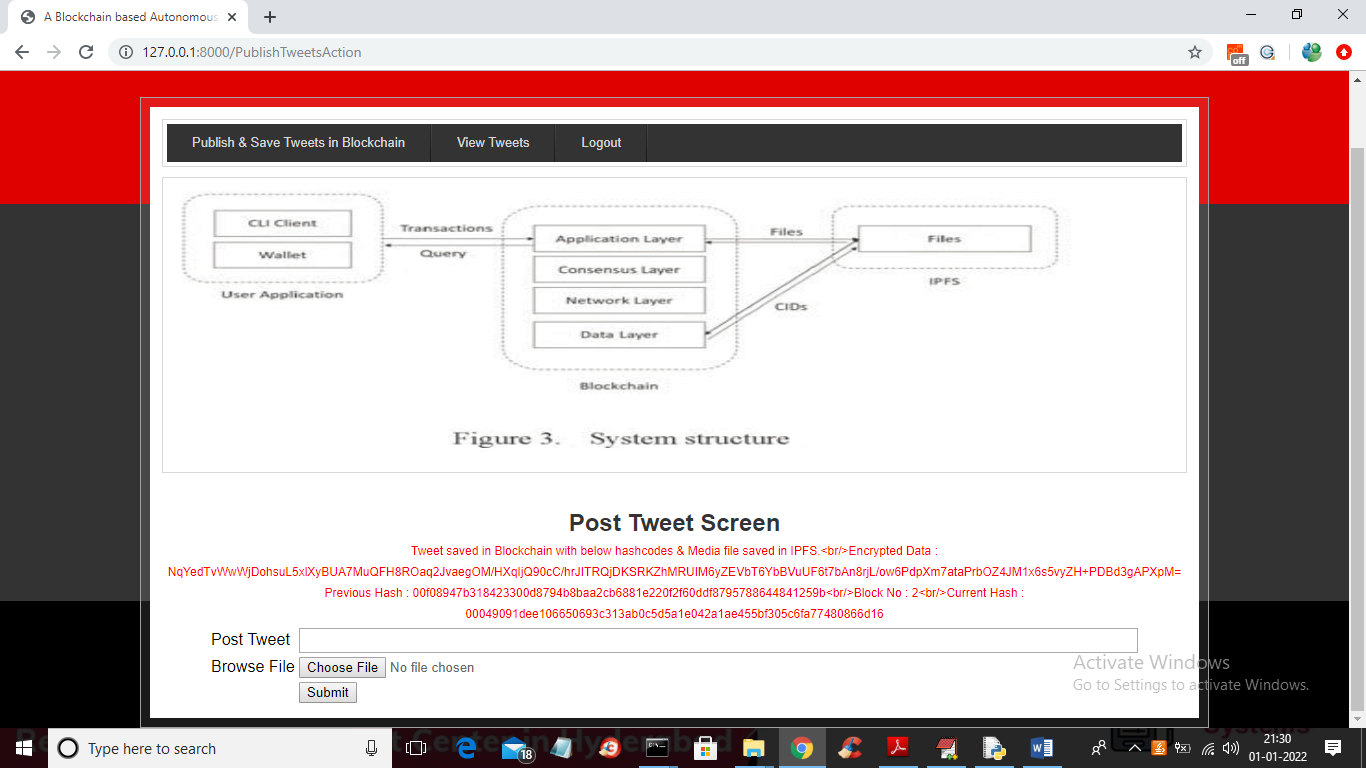
In above screen user is login and after login will get below screen



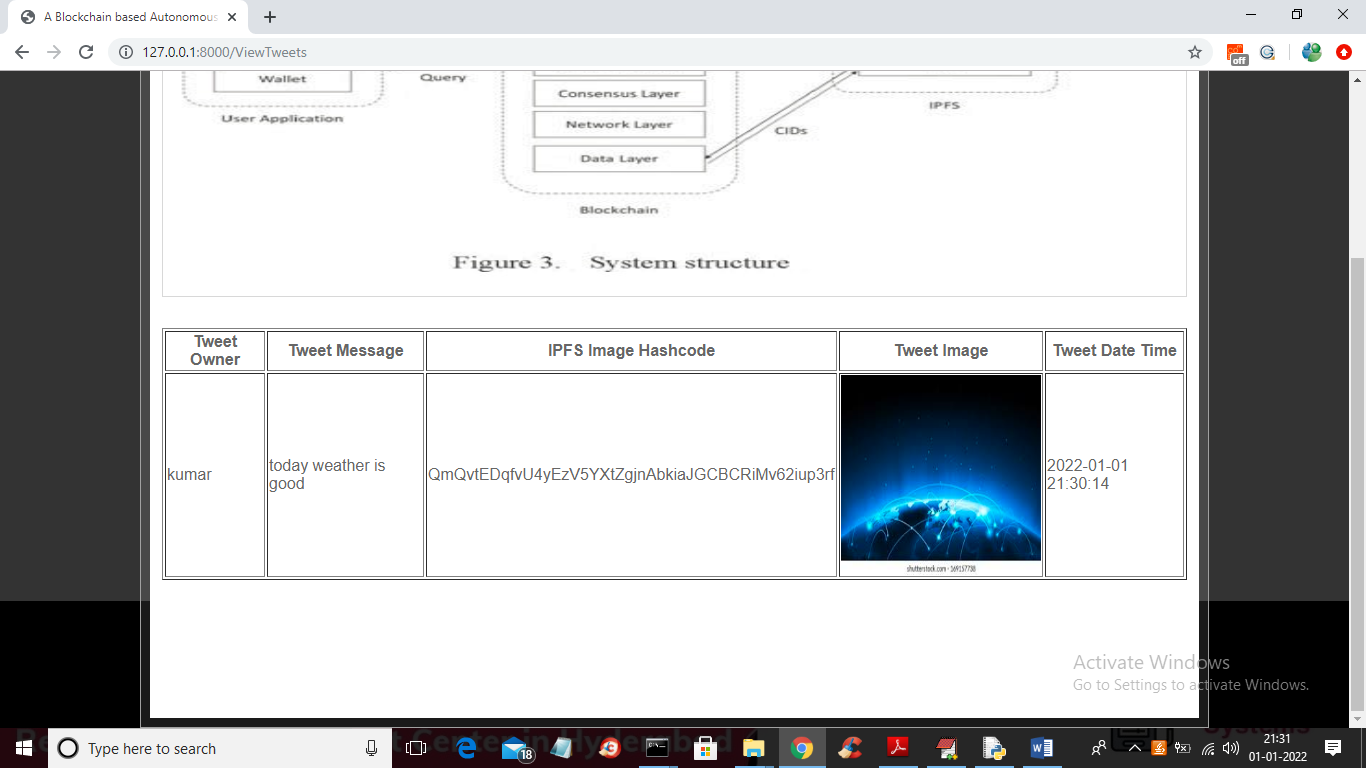
In above screen click on ‘Publish & Save Tweets in Blockchain’ link to get below screen



In above screen user can enter post message and then upload post related image and then click on ‘Open’ and ‘Submit’ button to save post data in Blockchain and get below output



In above screen in red colour text we can see hashcode of block data stored in Blockchain and similarly any number of users can signup and post details. Now click on ‘View Tweets’ link to get below screen



In above screen all users can see one and other tweets with all post details and images.

In above screen user can view the page deployed on IPFS and Blockchain and similarly you can deployed and access any number of web pages

**8. CONCLUSION**

**8.1 Conclusion**

This paper presents an implementation of blockchain used in OSN. Users keep their security information under their control, in order to avoid security information leakage from centralized servers. Additionally, since the social network service is decentralized, users do not need to worry about service crash down by centralized entity.

**8.2 Further Enhancements**

Integration of Decentralized Identity Solutions: Explore the integration of decentralized identity solutions, such as Self-Sovereign Identity (SSI) protocols, to enhance user privacy, control, and authentication within the OSN. This would enable users to have more control over their personal information and selectively share it with others.

Enhanced Content Moderation: Implement advanced content moderation mechanisms using artificial intelligence and machine learning algorithms. This could help identify and flag inappropriate or harmful content more accurately, ensuring a safer and more reliable user experience.

Interoperability with Other Blockchains: Enable interoperability with other blockchain networks to foster a broader decentralized ecosystem. This would allow users to connect and interact with users from different blockchain-based social networks and facilitate seamless data exchange and collaboration.

Incentive Mechanisms and Token Economy: Introduce tokenization and incentive mechanisms to reward users for contributing valuable content, engaging in positive interactions, and maintaining a healthy social network. This could incentivize user participation and contribute to the growth and sustainability of the decentralized OSN.

Decentralized Governance: Implement decentralized governance mechanisms, such as on-chain voting and decision-making processes, to allow users to actively participate in the governance of the OSN. This would enable community-driven decision-making and ensure the network evolves in a transparent and inclusive manner.

Integration of Privacy-Enhancing Technologies: Explore the integration of privacy-enhancing technologies, such as zero-knowledge proofs, homomorphic encryption, and secure multi-party computation. These technologies can further enhance user privacy and data protection within the decentralized OSN.

Enhanced User Experience: Continuously improve the user interface and experience to make the decentralized OSN more intuitive, visually appealing, and user-friendly. Gather user feedback and conduct usability testing to identify areas for improvement and implement enhancements accordingly.

Mobile Application Development: Develop dedicated mobile applications for iOS and Android platforms to provide users with a seamless mobile experience and access to the decentralized OSN on the go.

Integration with Web3 and DApps: Explore the integration of Web3 technologies and decentralized applications (DApps) within the OSN. This would enable users to access and interact with a wide range of decentralized services, such as decentralized finance (DeFi) applications, decentralized marketplaces, and decentralized social applications.

Cross-Chain Compatibility: Investigate cross-chain compatibility to enable the decentralized OSN to interact with multiple blockchain networks, allowing users to connect and share content across different blockchain ecosystems.

These future enhancements aim to leverage emerging technologies, address user needs and demands, and further enhance the decentralized nature, privacy, security, and user experience of the blockchain-based autonomous decentralized online social network

**9. Bibliography**

**9. Bibliography**

**9.1 Books References**

Tanenbaum, Andrew S., and Maarten van Steen. "Distributed Systems: Principles and Paradigms." Pearson Education, 2006.

Mougayar, William. "The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology." Wiley, 2016.

Gens, Frank. "IDC FutureScape: Worldwide IT Industry 2019 Predictions." IDC, 2018.

**9.2 Websites References**

IPFS - InterPlanetary File System. (https://ipfs.io/)

Ethereum - A Decentralized Platform for Applications. (https://ethereum.org/)

Blockchain Basics: A Non-Technical Introduction in 25 Steps. (https://www.eweek.com/security/blockchain-basics-a-non-technical-introduction-in-25-steps)

**9.3 Technical Publication References**

Nakamoto, Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System." (2008).

Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum Project Yellow Paper 151 (2014).

Swan, Melanie. "Blockchain: Blueprint for a New Economy." O'Reilly Media, 2015.

Buterin, Vitalik. "Ethereum White Paper." (2013).

Please note that the references provided above encompass both foundational concepts in distributed systems, blockchain technology, and specific resources related to IPFS and Ethereum. These sources were consulted to develop and implement the project, ensuring a strong theoretical and practical foundation.

10 Appendices

A. SW used

B. Methodologies used

C. Testing Methods used etc.,

Etc.,

11 Plagiarism Report