Decentralized Social Media App using Blockchain

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

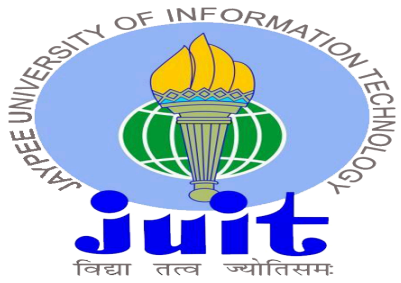
Submitted by

Saurabh Kumar Jha (201375)

Yashaswi Kohli (201138)

Under the guidance & supervision of

Dr. Amol Vasudeva



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: …….

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ __**Department:** _____ **Enrolment No** _____

Contact No. _____**E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……..(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

CERTIFICATE

This is to certify that the work which is being presented in the project report titled "**Decentralized Social Media App using Blockchain**" in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by "Saurabh Kumar Jha, 201375", "Yashaswi Kohli, 201138", during the period from June 2023 to May 2024 under the supervision of Dr. Amol Vasudeva, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

SAURABH KUMAR JHA (201375)

YASHASWI KOHLI (201138)

The above statement made is correct to the best of my knowledge.

Dr. Amol Vasudeva
Assistant Professor SG
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this report entitled **'Decentralized Social Media App using Blockchain'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Amol Vasudeva** (Assistant Professor (SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Student Name: Saurabh Kumar Jha
Roll No.: 201375

Student Name: Yashaswi Kohli
Roll No.: 201138

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Name: Dr. Amol Vasudeva
Designation: Supervisor
Department: Department of CSE & IT
Dated:

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing to make it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Amol Vasudeva, Assistant Professor(SG),** Department of CSE Jaypee University of Information Technology, Waknaghat Deep Knowledge & keen interest of my supervisor in the field of "**Blockchain**" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Amol Vasudedva,** Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

SAURABH KUMAR JHA (201375)

YASHASWI KOHLI (201138)

TABLE OF CONTENTS

List of Figures

ABSTRACT

“Decentralized Social Media App using Blockchain” represents a pioneering effort to redefine the landscape of social media platforms. By harnessing the power of blockchain, specifically Ethereum smart contracts, the project establishes a foundation built on data integrity, user accountability, and transparency. Throughout development, critical testing phases were conducted using tools such as Truffle, Ganache, Mocha, and Chai.

Comprehensive testing covered unit tests to scrutinize the functionalities of smart contracts, integration tests to ensure seamless interactions between components, UI tests to guarantee a responsive and user-friendly interface, security tests to identify and eliminate vulnerabilities, and performance tests to assess scalability and transaction speed.

The application's smart contract functionalities underwent meticulous examination. User registration, content posting, and token transactions were tested to ensure the seamless execution of these core features. Additionally, the UI was scrutinized for authentication processes, content display, and responsive design across various devices.

The security aspect was a focal point, with particular attention given to potential vulnerabilities such as reentrancy attacks and unauthorized access. Robust security measures were implemented to fortify the smart contracts against exploitation.

Performance testing played a pivotal role in evaluating the application's responsiveness under different scenarios. Transaction speeds on the blockchain, as well as scalability under varying user loads, were scrutinized to ensure optimal performance.

CHAPTER 01: INTRODUCTION

1.1 INTRODUCTION

The current reality sees a combination of actual exposure and digital communication. We do not stop to realize that we are sharing data 24/7, be it for business or pure entertainment when we are around people. But here's the tricky part: where are the boundaries and without that, how can we feel safe and secure? It seems that I am on a wire that I am trying to find the right tipping point.

Last but not least, social media is another significant aspect of the future of work. 'We' utilize one, right? While they help us simplify our lives, are we ever concerned about anonymous control over the information about us by big platforms? Would on the other hand a lack of privacy and possibly censorship be the solution? It feels pretty good, and more importantly, it evokes a desire for this kind of progress.

Decentralization of social media is one of the revolutionary changes in the social media world and I am sure that the time is coming when we will see the centralization of the world of social media. Imagine that social media would be developed without data elation from corporations. Creators, not revenue is priority. Rather than being densely packed into a single source, it is diffused across a user network, resembling a digital-style neighborhood watch, on a small scale. From this you are able to be the one to take responsibility for your assets and there is a little more making it somehow impossible for other people to mess with it.

And get this since you cannot trace through how your data behaves there is something called blockchain technology which is super secure and transparent during the interaction on these platforms. Finally coming to an end of all your posts being taken down for something that may not be their meaning, neither you worrying about your personal info getting leaked.

Yet, the majority of pleasant experiences you will have to face will be coming as clouds and horse-monsters. Not everything is good about it, though, like getting everybody to approve of the changes that have been made and making certain that users will find it user-friendly.

Here we analyse the role of decentralized social media. We will take it apart, see what makes it move and why it rocks, and finally unravel the issues that still makes it flourish. Hence, we discuss the potential of social media of the kind which expects to return control in the hands of the people.

1.2 PROBLEM STATEMENT

User data privacy and content control have become important issues in today's centralized social media era. Users often sacrifice their personal information, and centralized authorities wield considerable power over content modification and distribution. This creates an environment vulnerable to data breaches, censorship and manipulation. Traditional platforms lack transparency, leaving users' data usage in the dark. This report addresses the urgent need for decentralized social media applications using blockchain technology. In doing so, we aim to empower users with data ownership, increase transparency and create a sustainable platform against censorship. The issues at hand highlight the need for a decentralized social media paradigm shift for a safer, user-driven online experience.

1.3 OBJECTIVES

The focus of this report is to explore the benefits and challenges associated with the development and vast adoption of decentralized social media application platforms using blockchain technology. In today's centralized social media landscape, user data privacy and content control require a shift to more centralized and secure alternatives.

This report aims to provide a comprehensive overview of how decentralized social media using blockchain, can empower users by controlling their data. We try to analyzes the technical aspect of blockchain implementation, such as decentralized storage and transparent consensus mechanisms, which contribute to improving data security and integrity.

Rather than using separate platforms for communication, the integration of the chat messaging function into decentralized social media enables users to enjoy a continuous and converged process. Social media is the place where all that is required is one-on-one conversation or group chats. Thus users can technically have a real-time interaction anonymously without compromising their data privacy. Message replacements are accomplished with end-to-end currency and decentralized saving typically preventing discussion data from getting to unauthorized access and surveillance as well.

We have highlighted the potential societal impact of decentralized social media such as increased freedom of expression, resistance to censorship, and the creation of a more transparent online platform. Through this research we have tried to add valuable insights that can inform stakeholders, developers, and users about the transformative potentials of decentralized social media application. Our goal is to pave the way for a more user-driven, secure and transparent digital social landscape.

1.4 SIGNIFICANCE AND MOTIVATION FOR THE PROJECT WORK

Our project is very beneficial as it addresses the critical need for a more secure and user centric social media experience for everyone. By using blockchain for decentralization Our aim is to let users have control over their data and ease issues of censorship and data security breaches. The data will not be held by a single entity or organization/tech giants.Instead it will be only linked to the users. This improves transparency on what is being done with everyones data and thus it can’t be misused. Ensuring a flexible and secure platform for online social interactions.

1.5 ORGANIZATION OF PROJECT REPORT

The project record is dependent to provide a comprehensive know-how of the development and checking out processes worried in creating the Decentralized Social Media App the usage of Blockchain. The corporation is designed to present a logical go with the flow of data guiding readers through the numerous components of the task.

Chapter 1: Introduction

This segment introduces the assignment & outlining its goals, importance & the technological context. It offers an overview of the decentralized social media app and its potential effect at the social media panorama.

Chapter 2: Literature Review

An evaluate of relevant literature is presented, exploring present blockchain-based totally social media structures, decentralized programs, and key technologies. This bankruptcy sets the theoretical framework for the venture.

Chapter 3: System Architecture

Detailed facts at the architecture of the decentralized social media app is supplied on this chapter. It covers the TECH stack as MEEN, the role of smart contracts & the overall layout of the device.

Chapter 4: Testing

Chapter four delves into the testing phase of the project:

4.1 Testing Strategy: we check on the overall checking out approach used highlighting the tools used & the reason behind their choice.

4.2 Test Cases and Outcomes: A particular presentation of take a look at cases for smart contracts, user interfaces, safety features, and performance elements. Outcomes of every test case are mentioned.

Chapter 5: Results and Discussion

This chapter translates the results received at some point of trying out. It discusses the results of the findings, addressing each successes and challenges encountered throughout the development and trying out stages.

Chapter 6: Conclusion

Summarizing the project Chapter 6 provides key takeaways & lessons learned, and the overall significance of the decentralized social media app. It also hints at potential future developments or improvements.

References

A very big list of references is provided where we have citing sources & literature that tell us the story the project's development and testing methodologies.

By this section we try to to provide a clear and structured narrative of our project thus guiding readers through the project's starting to end including development & testing & output.

CHAPTER 02: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
1.	D-Space: A Decentralized Social Media App [1]	ICECAA (2023)	Ethereum blockchain, smart contracts, real-world dataset	Uses blockchain technology to improve security, privacy, and transparency.	D-Space is still under development and not yet been widely adopted
2.	Performance Evaluation of Decentralized Social Media on Near Protocol Blockchain	IMCOM (2023)	Near Protocol blockchain, smart contracts, real-world dataset	Comparable to centralized social media platforms, with the added benefits of security and privacy.	Near Protocol is a relatively new blockchain platform & has not yet been as widely adopted as Ethereum
3.	Exploring the Potential of Interplanetary File System for Secure and Transparent Social Media	ICRAIE (2023)	Interplanetary File System (IPFS), blockchain, real-world dataset	IPFS can be used to create a secure and transparent social media platform by storing user data on a decentralized network.	IPFS is still under development and can be slow & inefficient for large files
4.	Systematic Literature Review and Qualitative Survey of Blockchain Impact on Social Media Security	ICOIACT (2022)	Systematic literature review and qualitative survey	Blockchain technology has the potential to improve security, privacy, and transparency in social media platforms.	This study is limited by the size and scope of the literature review and survey

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
5.	A Blockchain Based Autonomous Decentralized Online Social Network	ICCECE (2021)	Ethereum blockchain, smart contracts, real-world dataset	The network is resistant to censorship and attacks.	The proposed network is still under development and has not yet been widely adopted.
6.	A Secure and Verifiable Data Sharing Scheme Based on Blockchain in Vehicular Social Networks	IEEE (2021)	Blockchain, smart contracts, real-world dataset	It's designed for vehicular social networks. The scheme is resistant to tampering and unauthorized access.	The proposed scheme is still under development and has not yet been widely adopted.
7.	Systematic Literature Review and Qualitative Survey of Blockchain Impact on Social Media Security	ICOIACT (2021)	Systematic literature review and qualitative survey	Blockchain technology has the capability to elevate security, privacy, and transparency within the realm of social media platforms.	The study's scope and the extent of the literature review and survey are constrained by their size.
8.	Blockchain Based Notarization for Social Media	ICCE (2019)	Blockchain, smart contracts, real-world dataset	IT can be used to verify the authenticity and integrity of social media posts. The scheme is resistant to tampering and forgery.	This one scheme is still under development & has not yet been widely adopted.

The literature review table provides a comprehensive overview of recent research exploring the intersection of blockchain technology and social media platforms. The studies highlighted in the table majorly are contributing valuable insights into the potential applications, tools, techniques, datasets used, results obtained & limitations encountered in this evolving field.

Several of the researchers have delved deeper into the development of decentralized social media applications using blockchain and have a focus on enhancing security, privacy, and transparency. For instance, "D-Space: A Decentralized Social Media App" [1] employs Ethereum blockchain and smart contracts to create a secure and transparent social media platform. However, its limited adoption indicates ongoing development.

Similarly, the study on "Performance Evaluation of Decentralized Social Media on Near Protocol Blockchain" highlights the potential benefits of using Near Protocol blockchain for secure and private social media. Nonetheless, the limitation of Near Protocol's adoption suggests that the technology is still gaining traction.

Interplanetary File System (IPFS) is explored in "Exploring the Potential of Interplanetary File System for Secure and Transparent Social Media," emphasizing its ability to create secure platforms. However, IPFS's developmental status and potential inefficiency for large files are acknowledged as limitations.

A systematic literature review and qualitative survey in "Systematic Literature Review and Qualitative Survey of Blockchain Impact on Social Media Security" emphasize the transformative impact of blockchain on social media security. However, the study acknowledges constraints related to the size and scope of the literature review and survey.

Other studies, such as "A Blockchain Based Autonomous Decentralized Online Social Network" and "A Secure and Verifiable Data Sharing Scheme Based on Blockchain in Vehicular Social Networks," highlight the resistance to censorship and tampering in blockchain-based social networks. Nevertheless, their limited adoption indicates that these systems are still in the developmental phase.

"Blockchain Based Notarization for Social Media" proposes a scheme to verify the authenticity and integrity of social media posts using blockchain. However, its limited adoption suggests ongoing development and potential challenges in widespread implementation.
The literature review indicates a burgeoning interest in leveraging blockchain for secure, transparent, and resilient social media platforms, while recognizing the developmental nature and adoption challenges associated with these innovative solutions.

2.2 KEY GAPS IN LITERATURE

- Lack of research: The majority of studies on decentralized social media are little and have a limited duration, which makes it challenging to draw broad conclusions about the platform's long-term impacts.
- User behavior: Further studies are required to determine the elements that influence the acceptance & use of decentralized social media platforms as well as how users of the apps interact with them.
- Governance of Decentralized Social Media Platforms: If decentralized social media platforms are to be controlled in a fair & open & responsible manner then more research is required.
- Scalability of decentralized social media networks: More investigation is required to determine whether decentralized social media networks can accommodate a high quantity of users for their applications.
- Further investigation is required to understand how decentralized social media platforms can safeguard freedom of expression and how it will promote social cohesion.

CHAPTER 03: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

In order to develop a successful Decentralized Social Media App using Blockchain we have to identify key requirements and conduct a deep analysis as mentioned below:

Requirements:

1. User Data Control:

Implement a robust system which enables users to have owner and control over their/theirs personal data. Leverage blockchain's cryptographic principles to ensure secure data management.

2. Decentralized Storage:

Integrate decentralized storage solutions like IPFS to distribute and secure user-generated content across the network, reducing reliance on a central server.

3. Transparency and Immutability:

Utilize blockchain's transparency to enhance content moderation and ensure an immutable record of interactions, fostering trust among users.

4. User Authentication:

Develop a secure user authentication system using cryptographic techniques to safeguard user identities and prevent unauthorized access.

4. Chat and Messaging:

Develop a user friendly module for seamless integration of chat and messaging features with live interactions using WebRTC and Socket.io technologies.

Analysis:

1. Security:

Assess the robustness of the chosen blockchain protocol and decentralized storage solutions to mitigate potential security vulnerabilities.

2. Scalability:

Evaluate the scalability of the chosen blockchain infrastructure to ensure it can handle a growing user base and increasing data volume.

3. User Experience:

Analyze the user interface and overall experience to guarantee that users, regardless of technical expertise, find the platform intuitive and user-friendly.

4. Regulatory Compliance:

Consider regulatory implications and ensure that the decentralized social media app adheres to data protection and privacy laws.

5. Community Governance:

Explore mechanisms for community-driven decision-making, such as voting systems, to empower users and ensure a democratic platform governance model.

Now we can start addressing these requirements and conducting a thorough analysis then our Decentralized Social Media App can provide users with enhanced data control, security, and transparency while fostering a good UI/UX and becoming a scalable platform.

3.2 PROJECT DESIGN AND ARCHITECTURE

3.2.1 PROJECT DESIGN

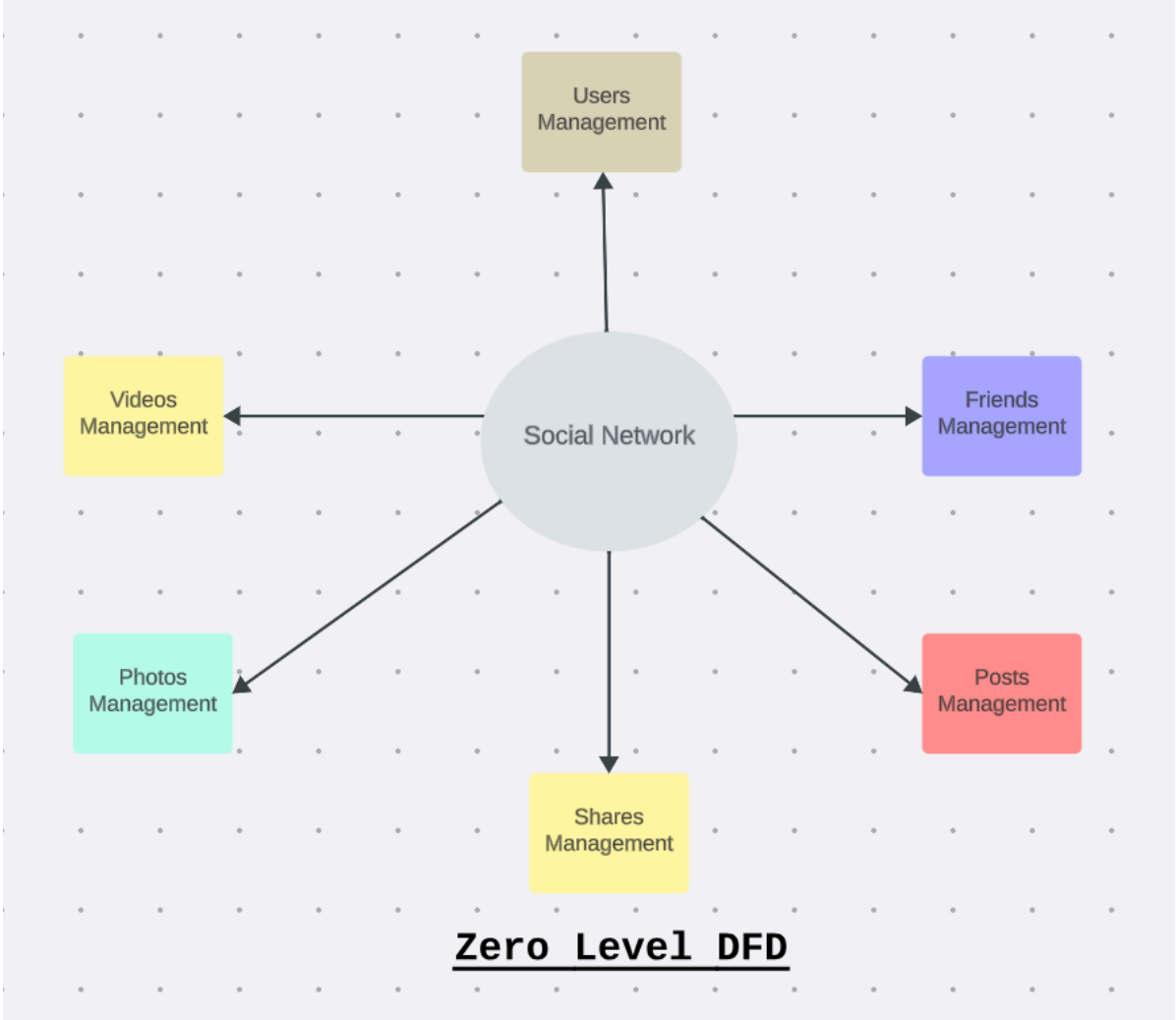


Fig-3.2.1: Zero Level DFD

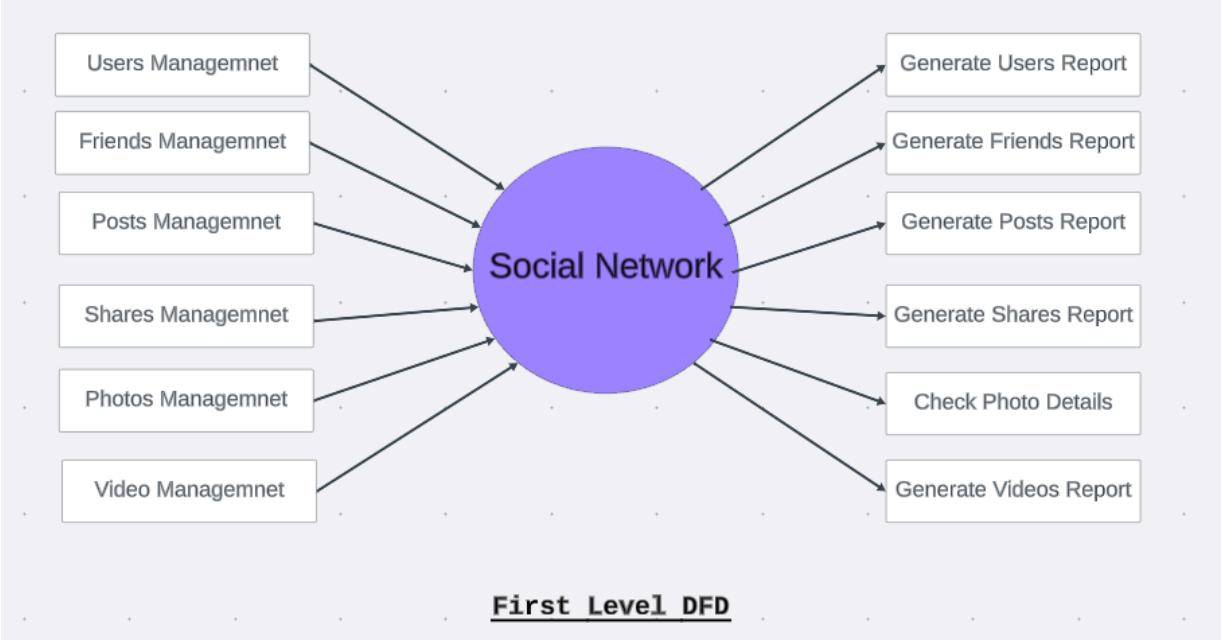


Fig-3.2.2: First Level DFD

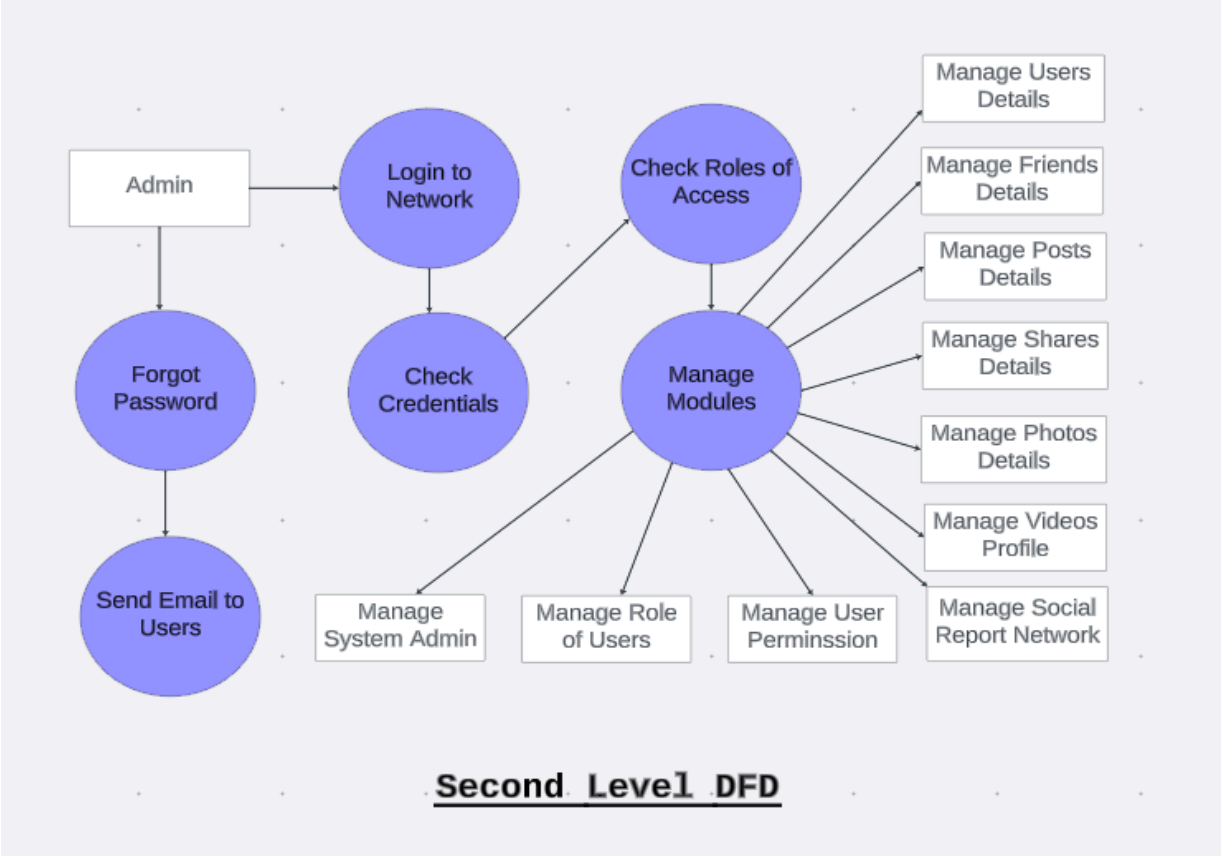


Fig-3.2.3: Second Level DFD

3.2.1 PROJECT ARCHITECTURE

Designing the architecture for a Decentralized Social Media App using Blockchain involves several key components to ensure privacy & security and seamless functionality. Here's a detailed breakdown:

1. **Blockchain Integration:**
    - Utilize a permissioned blockchain for enhanced security and control.
    - Employ a consensus mechanism, such as Proof-of-Stake, for efficient and secure transaction validation.

2. **User Authentication and Authorization:**
    - Implement a secure user authentication system using cryptographic techniques.
    - Develop a permission model to control user access to various features and data.

3. **Decentralized Storage (IPFS):**
    - Integrate decentralized storage (e.g., IPFS) to ensure content is distributed across the network.
    - Use encryption/decryption techn. to protect user data stored on the blockchain.

4. **Smart Contracts:**
    - Develop smart contracts to automate and enforce rules for interactions on the platform.
    - Include functionalities for content creation, sharing, and moderation.

5. **User Profile Management:**
    - Create a decentralized identity system for users, allowing them to have control over their profile information.
    - Implement mechanisms for users to manage their privacy settings.

6. **Content Distribution:**
    - Design a content delivery system that leverages decentralized storage and ensures efficient retrieval.
    - Implement algorithms for content recommendation based on user preferences.

7. **Interoperability:**
    - Ensure compatibility with existing social media platforms to facilitate a smooth transition for users.
    - Allow users to import or export their data accordingly.

8. **Scalability and Performance:**
    - Optimize the architecture for scalability to accommodate a growing user base.
    - Employ caching mechanisms and load balancing to enhance performance.

9. **User Interface & User Experience:**
    - We designed intuitive & user-friendly interface to encourage major adoption.
    - Focus on responsive design and accessibility for a diverse user demographic.

10. **Community Governance:**
    - Implement decentralized decision-making mechanisms, possibly through voting systems.
    - Allow users to actively participate in the governance and moderation of the platform.

11. **Security Measures:**
    - Conduct regular security audits to identify and address vulnerabilities.
    - Employ encryption for secure communication and data transmission.

12. **Monitoring and Analytics:**
    - Integrate analytics tools to track user engagement and platform performance.
    - Implement monitoring systems for real-time detection of suspicious activities.

3.3 DATA PREPARATION

First we have to understand blockchain before we understand what blockchain database is how it is designed. Blockchain is just a ledger to store the data, the only difference is that this ledger is on the internet, therefore we can also say it is a digital ledger which is used to store transactional data. The data are stored in blocks, these blocks are connected to each other, and these chain are immutable which means it cannot be changed easily, to make a change into a data which we have stored into block, we have to make change in all data which are present in different blocks signed by different users, therefore it is impossible to make a change, and this make it blockchains immutable.

As we know that blockchains are used to store data, so we could say that blockchain is a one of type of database. Blockchains are generally used to store transactional data, as we know it was introduced for cryptocurrency but now it is used in many other places like NFTs also. Here in blockchain we could read and create only. But databases are different, they can store various types of data and support CRUD operations. But in blockchain, there is a difference here: every type of block needs to be verified, which makes it slower but more secure than other database types.

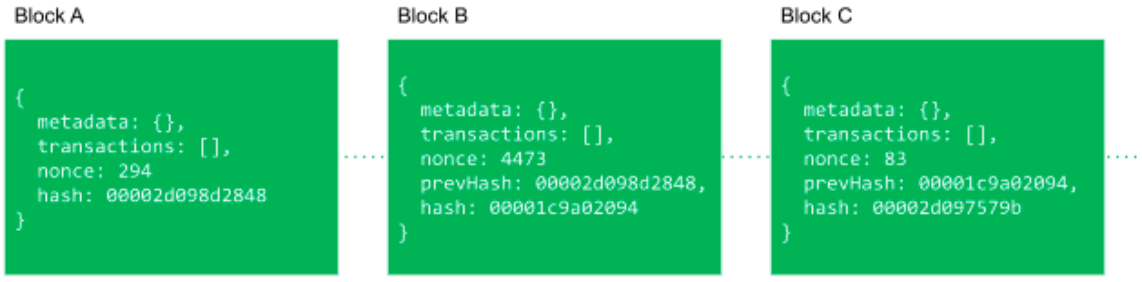


Fig-3.3.1: The blockchain architecture

A blockchain is made of multiple blocks which contains the information about transactions. Each and every transaction inside the block is signed using a public key which is a cryptographic algorithm like AES, SHA or ECDHE. The blocks in blockchain are also signed with an SHA-256 signature. The next block in the blockchain uses the previous block signature to link to the rest of the chain.

To make changes, first we have to get approval or validation from most of the nodes, so then we will be able to add a new block in the chain which is blockchain. The combination of different signatures and consent make it difficult to change the data in blockchain without the huge amount of data manipulation capacity.

These Blockchains generally contain transaction records (as in they were build for storing the transaction records of exchange of bitcoin) but the problem these have very restricted querying abilities. One of the other problems we face is that there is a lot of computation power and exertion indispensable to prove that a given block is valid or not. To ensure that the block is a genuine one , we have to get a larger number of nodes to consent to it. But now this is another problem which if we have a larger number of nodes in our system then it will take more time to validate the block. So in some cases it is hard for us to use the blockchain as a database in the traditional manner.

So how should we deal with this problem? The solution is easy, as we know it is easy to use today's database for traditional scenarios, and to have additional features of blockchain we can simply set up the blockchain as the first layer and other databases as the second layer. As the blockchain is known for its privacy, that will help us to ensure security and high level integrity of protocol, and the best thing it will be lightweight and while doing this all it gives us good performance for querying. The second layer would be using PoW which is known as proof of work based blockchain. This is used to store data of transactions of all the database operations that have been there, which is coming from the first layer.

Then the two layers are inter-connected through a blockchain-connecting algorithmic mechanism. This particular mechanism links parts of the first layer blocks with the second layer blocks. This creates a chain of evidence evaluating, securing and authenticating data from the particular first layer.

Now before we start making a new blockchain database, there are some points we should keep in our mind and take into account, the first thing is the database we are deploying for what purpose is it for an enterprise or is it for a conglomerate. Generally we say that Blockchains are

decentralized but it is not true , it is not necessary to be decentralized. In some cases we have seen an organization using a blockchain database internally and they make it look like they are the one who controls the data which is not true. Generally it is seen that blockchain are used to operate in a conglomerate. In such cases cryptocurrencies have been seen to be used for some particular unconventional model to make sure that no one particular is in control of the data.

Here each validation node needs to have a copy of the data. Now we will use this particular incident of data. Data which will be used directly by clients which connect to the database is termed/known as operational data. This is the case for all crypto currency stuff. Anyone is able to query & perform actions on the particular blockchain. Non-operational stuff on the other hand would be accessed via a required intermediary.

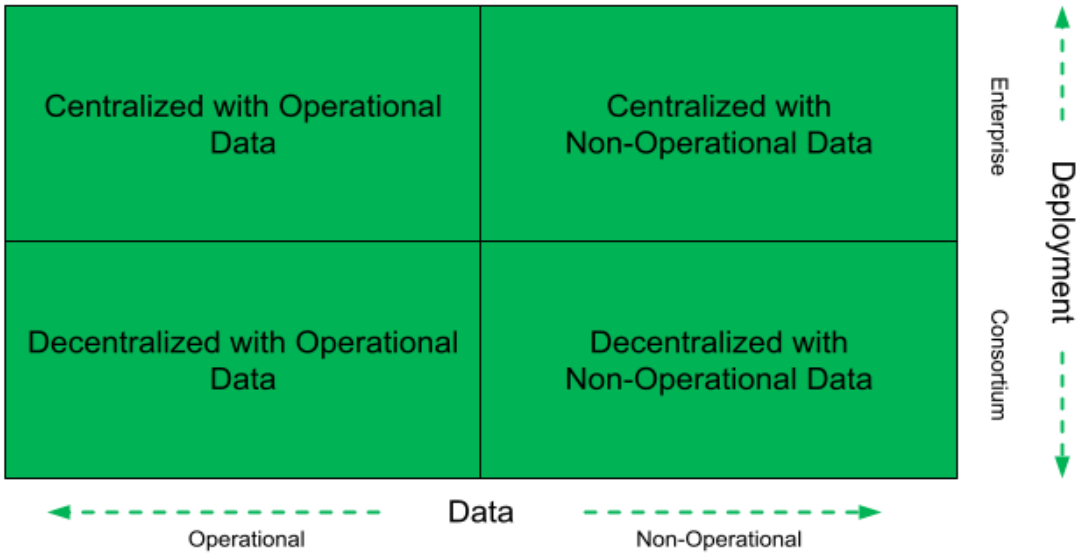


Fig-3.3.2:Type of blockchain architecture directly proportional to deployment type & data type.

For decentralized social media, we would have to deal with non-operational data, so below is the architecture for decentralized deployment for non operational data.

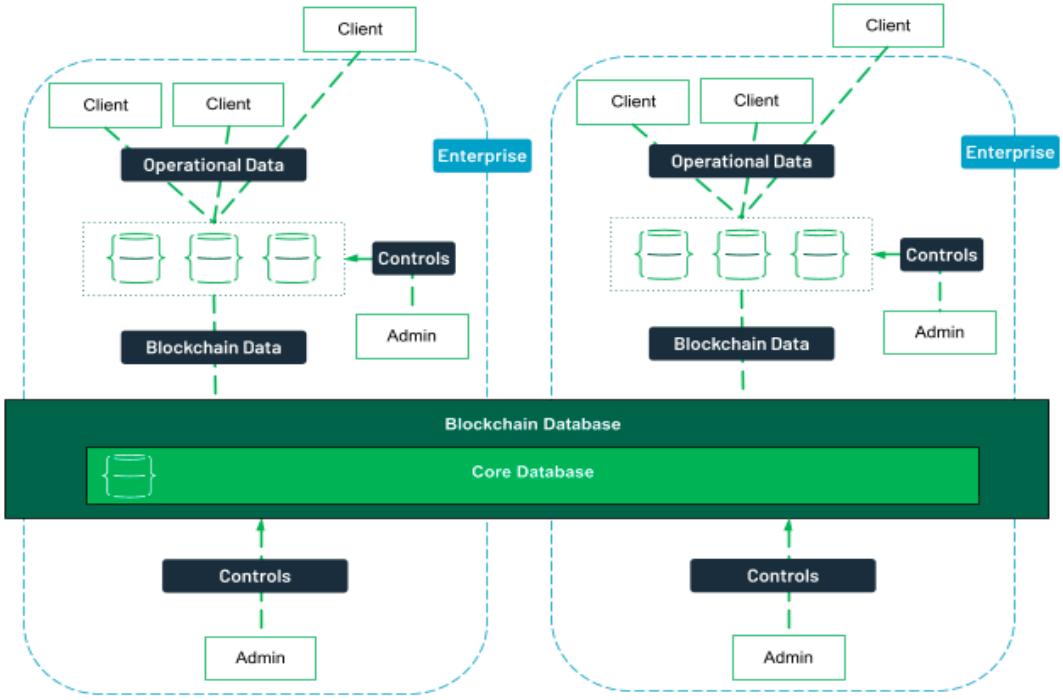


Fig-3.3.3: Decentralized with non-operational data

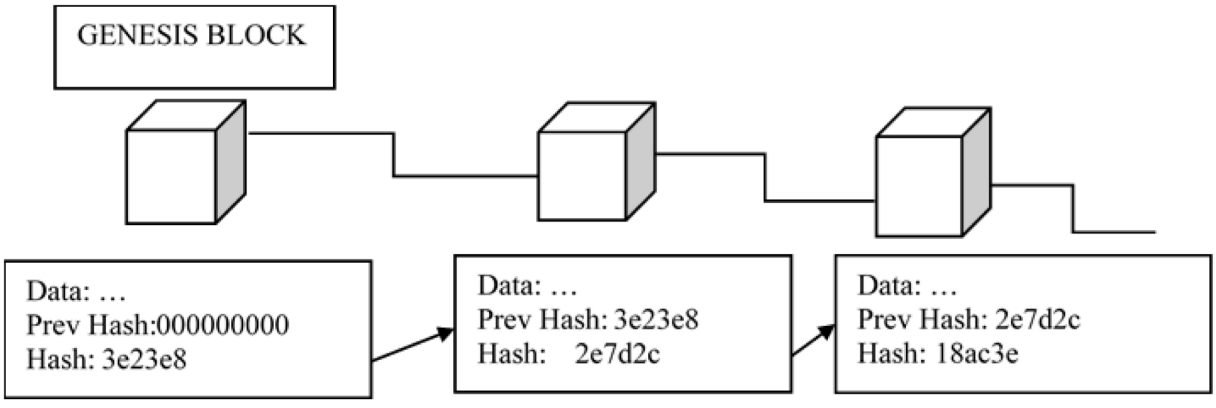


Fig-3.3.4: Illustration of block links of blockchain ledger

3.4 IMPLEMENTATION

```
const SigninForm = () => {
  const { toast } = useToast();
  const navigate = useNavigate();
  const { checkAuthUser, isLoading: isUserLoading } =
useUserContext();

  // Query
  const { mutateAsync: signInAccount, isLoading } =
useSignInAccount();

  const form = useForm<z.infer<typeof SigninValidation>>({
    resolver: zodResolver(SigninValidation),
    defaultValues: {
      email: "",
      password: "",
    },
  });

  const handleSignin = async (user: z.infer<typeof
SigninValidation>) => {
    const session = await signInAccount(user);

    if (!session) {
      toast({ title: "Login failed. Please try again." });

      return;
    }

    const isLoggedIn = await checkAuthUser();

    if (isLoggedIn) {
      form.reset();

      navigate("/");
    } else {
      toast({ title: "Login failed. Please try again.", });

      return;
    }
  };
```

Fig. 3.4.1: Sign-in Implementation

```
const handleSignup = async (user: z.infer<typeof SignupValidation>) =>
 {
    try {
      const newUser = await createUserAccount(user);

      if (!newUser) {
        toast({ title: "Sign up failed. Please try again.", });

        return;
      }

      const session = await signInAccount({
        email: user.email,
        password: user.password,
      });

      if (!session) {
        toast({ title: "
Something went wrong. Please login your new account", });

        navigate("/sign-in");

        return;
      }

      const isLoggedIn = await checkAuthUser();

      if (isLoggedIn) {
        form.reset();

        navigate("/");
      } else {
        toast({ title: "Login failed. Please try again.", });

        return;
      }
    } catch (error) {
      console.log({ error });
    }
  };
```

Fig. 3.4.2: Sign-up Implementation

```
import { Routes, Route } from "react-router-dom";

import {
  Home,
  Explore,
  Saved,
  CreatePost,
  Profile,
  EditPost,
  PostDetails,
  UpdateProfile,
  AllUsers,
} from "@/_root/pages";
import AuthLayout from "./_auth/AuthLayout";
import RootLayout from "./_root/RootLayout";
import SignupForm from "@/_auth/forms/SignupForm";
import SigninForm from "@/_auth/forms/SigninForm";
import { Toaster } from "@/components/ui/toaster";

import "./globals.css";

const App = () => {
  return (
    <main className="flex h-screen">
      <Routes>
        {/* public routes */}
        <Route element={<AuthLayout />}>
          <Route path="/sign-in" element={<SigninForm />} />
          <Route path="/sign-up" element={<SignupForm />} />
        </Route>

        {/* private routes */}
        <Route element={<RootLayout />}>
          <Route index element={<Home />} />
          <Route path="/explore" element={<Explore />} />
          <Route path="/saved" element={<Saved />} />
          <Route path="/all-users" element={<AllUsers />} />
          <Route path="/create-post" element={<CreatePost />} />
          <Route path="/update-post/:id" element={<EditPost />}
/>
          <Route path="/posts/:id" element={<PostDetails />} />
          <Route path="/profile/:id/*" element={<Profile />} />
          <Route path="/update-profile/:id" element={<
UpdateProfile />} />
        </Route>
      </Routes>
```

Fig. 3.4.3: Overall overview of our main app

```
const Explore = () => {
  const { ref, inView } = useInView();
  const { data: posts, fetchNextPage, hasNextPage } =
useGetPosts();

  const [searchValue, setSearchValue] = useState("");
  const debouncedSearch = useDebounce(searchValue, 500);
  const { data: searchedPosts, isFetching: isSearchFetching } =
useSearchPosts(debouncedSearch);

  useEffect(() => {
    if (inView && !searchValue) {
      fetchNextPage();
    }
  }, [inView, searchValue]);

  if (!posts)
    return (
      <div className="flex-center w-full h-full">
        <Loader />
      </div>
    );

  const shouldShowSearchResults = searchValue !== "";
  const shouldShowPosts = !shouldShowSearchResults &&
    posts.pages.every((item) => item.documents.length === 0);

  return (
    <div className="explore-container">
      <div className="explore-inner_container">
        <h2 className="h3-bold md:h2-bold w-full">Search Posts</
h2>
        <div className="
flex gap-1 px-4 w-full rounded-lg bg-dark-4">
          <img
            src="/assets/icons/search.svg"
            width={24}
            height={24}
            alt="search"
          />
          <Input
            type="text"
            placeholder="Search"
            className="explore-search"
            value={searchValue}
            onChange={(e) => {
              const { value } = e.target;
              setSearchValue(value);
```

Fig. 3.4.4: Explore-section

```
import { Models } from "appwrite";

// import { useToast } from "@/components/ui/use-toast";
import { Loader, PostCard, UserCard } from "@/components/shared"
;
import { useGetRecentPosts, useGetUsers } from "
@/lib/react-query/queries";

const Home = () => {
  // const { toast } = useToast();

  const {
    data: posts,
    isLoading: isPostLoading,
    isError: isErrorPosts,
  } = useGetRecentPosts();
  const {
    data: creators,
    isLoading: isUserLoading,
    isError: isErrorCreators,
  } = useGetUsers(10);

  if (isErrorPosts || isErrorCreators) {
    return (
      <div className="flex flex-1">
        <div className="home-container">
          <p className="body-medium text-light-1">
Something bad happened</p>
        </div>
        <div className="home-creators">
          <p className="body-medium text-light-1">
Something bad happened</p>
        </div>
      </div>
    );
  }

  return (
    <div className="flex flex-1">
      <div className="home-container">
        <div className="home-posts">
          <h2 className="h3-bold md:h2-bold text-left w-full">
Home Feed</h2>
          {isPostLoading && !posts ? (
            <Loader />
```

Fig. 3.4.5: Home Page

```
const StatBlock = ({ value, label }: StabBlockProps) => (
  <div className="flex-center gap-2">
    <p className="small-semibold lg:body-bold text-primary-500">
{value}</p>
    <p className="small-medium lg:base-medium text-light-2">{
label}</p>
  </div>
);

const Profile = () => {
  const { id } = useParams();
  const { user } = useUserContext();
  const { pathname } = useLocation();

  const { data: currentUser } = useGetUserById(id || "");

  if (!currentUser)
    return (
      <div className="flex-center w-full h-full">
        <Loader />
      </div>
    );

  return (
    <div className="profile-container">
      <div className="profile-inner_container">
        <div className="
flex xl:flex-row flex-col max-xl:items-center flex-1 gap-7">
          <img
            src={
              currentUser.imageUrl || "
/assets/icons/profile-placeholder.svg"
            }
            alt="profile"
            className="w-28 h-28 lg:h-36 lg:w-36 rounded-full"
          />
          <div className="
flex flex-col flex-1 justify-between md:mt-2">
            <div className="flex flex-col w-full">
              <h1 className="
text-center xl:text-left h3-bold md:h1-semibold w-full">
                {currentUser.name}
              </h1>
              <p className="
```

Fig. 3.4.6: User Profile

```
import { convertFileToUrl } from "@/lib/utils";

type FileUploaderProps = {
  fieldChange: (files: File[]) => void;
  mediaUrl: string;
};

const FileUploader = ({ fieldChange, mediaUrl }:
FileUploaderProps) => {
  const [file, setFile] = useState<File[]>([]);
  const [fileUrl, setFileUrl] = useState<string>(mediaUrl);

  const onDrop = useCallback(
    (acceptedFiles: FileWithPath[]) => {
      setFile(acceptedFiles);
      fieldChange(acceptedFiles);
      setFileUrl(convertFileToUrl(acceptedFiles[0]));
    },
    [file]
  );

  const { getRootProps, getInputProps } = useDropzone({
    onDrop,
    accept: {
      "image/*": [".png", ".jpeg", ".jpg"],
    },
  });

  return (
    <div
      {...getRootProps()}
      className="
flex flex-center flex-col bg-dark-3 rounded-xl cursor-pointer">
      <input {...getInputProps()} className="cursor-pointer" />

      {fileUrl ? (
        <>
          <div className="
flex flex-1 justify-center w-full p-5 lg:p-10">
            <img src={fileUrl} alt="image" className="
file_uploader-img" />
          </div>
          <p className="file_uploader-label">
Click or drag photo to replace</p>
```

Fig. 3.4.7: File Upload implementation

```
import { useUserContext } from "@/context/AuthContext";

const PostDetails = () => {
  const navigate = useNavigate();
  const { id } = useParams();
  const { user } = useUserContext();

  const { data: post, isLoading } = useGetPostById(id);
  const { data: userPosts, isLoading: isUserPostLoading } =
useGetUserPosts(
    post?.creator.$id
  );
  const { mutate: deletePost } = useDeletePost();

  const relatedPosts = userPosts?.documents.filter(
    (userPost) => userPost.$id !== id
  );

  const handleDeletePost = () => {
    deletePost({ postId: id, imageId: post?.imageId });
    navigate(-1);
  };

  return (
    <div className="post_details-container">
      <div className="hidden md:flex max-w-5xl w-full">
        <Button
          onClick={() => navigate(-1)}
          variant="ghost"
          className="shad-button_ghost">
          <img
            src={"/assets/icons/back.svg"}
            alt="back"
            width={24}
            height={24}
          />
          <p className="small-medium lg:base-medium">Back</p>
        </Button>
      </div>

      {isLoading || !post ? (
        <Loader />
      ) : (
        <div className="post_details-card">
          <img
            src={post?.imageUrl}
```

Fig. 3.4.8: Post Details Information

```
} from "@/components/ui";
import { PostValidation } from "@/lib/validation";
import { useToast } from "@/components/ui/use-toast";
import { useUserContext } from "@/context/AuthContext";
import { FileUploader, Loader } from "@/components/shared";
import { useCreatePost, useUpdatePost } from "
@/lib/react-query/queries";

type PostFormProps = {
  post?: Models.Document;
  action: "Create" | "Update";
};

const PostForm = ({ post, action }: PostFormProps) => {
  const navigate = useNavigate();
  const { toast } = useToast();
  const { user } = useUserContext();
  const form = useForm<z.infer<typeof PostValidation>>({
    resolver: zodResolver(PostValidation),
    defaultValues: {
      caption: post ? post?.caption : "",
      file: [],
      location: post ? post.location : "",
      tags: post ? post.tags.join(",") : "",
    },
  });

  // Query
  const { mutateAsync: createPost, isLoading: isLoadingCreate }
=
    useCreatePost();
  const { mutateAsync: updatePost, isLoading: isLoadingUpdate }
=
    useUpdatePost();

  // Handler
  const handleSubmit = async (value: z.infer<typeof
PostValidation>) => {
    // ACTION = UPDATE
    if (post && action === "Update") {
      const updatedPost = await updatePost({
        ...value,
        postId: post.$id,
        imageId: post.imageId,
        imageUrl: post.imageUrl,
      });

      if (!updatedPost) {
```

Fig. 3.4.9: Post Upload Form

```
  useDeleteSavedPost,
  useGetCurrentUser,
} from "@/lib/react-query/queries";

type PostStatsProps = {
  post: Models.Document;
  userId: string;
};

const PostStats = ({ post, userId }: PostStatsProps) => {
  const location = useLocation();
  const likesList = post.likes.map((user: Models.Document) =>
user.$id);

  const [likes, setLikes] = useState<string[]>(likesList);
  const [isSaved, setIsSaved] = useState(false);

  const { mutate: likePost } = useLikePost();
  const { mutate: savePost } = useSavePost();
  const { mutate: deleteSavePost } = useDeleteSavedPost();

  const { data: currentUser } = useGetCurrentUser();

  const savedPostRecord = currentUser?.save.find(
    (record: Models.Document) => record.post.$id === post.$id
  );

  useEffect(() => {
    setIsSaved(!!savedPostRecord);
  }, [currentUser]);

  const handleLikePost = (
    e: React.MouseEvent<HTMLImageElement, MouseEvent>
  ) => {
    e.stopPropagation();

    let likesArray = [...likes];

    if (likesArray.includes(userId)) {
```

Fig. 3.4.10: Post Statistics Information

```
};

export const useSignInAccount = () => {
  return useMutation({
    mutationFn: (user: { email: string; password: string }) =>
      signInAccount(user),
  });
};

export const useSignOutAccount = () => {
  return useMutation({
    mutationFn: signOutAccount,
  });
};

// ============================================================
// POST QUERIES
// ============================================================

export const useGetPosts = () => {
  return useInfiniteQuery({
    queryKey: [QUERY_KEYS.GET_INFINITE_POSTS],
    queryFn: getInfinitePosts as any,
    getNextPageParam: (lastPage: any) => {
      // If there's no data, there are no more pages.
      if (lastPage && lastPage.documents.length === 0) {
        return null;
      }

      // Use the $id of the last document as the cursor.
      const lastId = lastPage.documents[lastPage
.documents.length - 1].$id;
      return lastId;
    },
  });
};

export const useSearchPosts = (searchTerm: string) => {
  return useQuery({
```

Fig. 3.4.11: Post Queries/Auth Queries Information

```
"use client";
import { User } from "@/components/renderPages";
import DisplayMessages from "@/components/ui/chats/displayMessages";
import MakeMessage from "@/components/ui/chats/makeMessage";
import ProfileImage from "@/components/ui/profileImage";
import axios from "axios";
import Link from "next/link";
import React, { useEffect, useState } from "react";
import { BiArrowBack } from "react-icons/bi";
import { LiaSpinnerSolid } from "react-icons/lia";

export interface Messages {
  id: string;
  text: string | null;
  image: string | null;
  video: string | null;
  createdAt: Date;
  chatRoomId?: string;
  user: User;
}

const Page = ({ params }: { params: { id: string } }) => {
  const [messages, setMessages] = useState<Messages[]>();
  const [chatInfo, setChatInfo] = useState<User>();
  const [loadingMessage, setLoadingMessage] = useState<boolean>(false);

  useEffect(() => {
    const getMessages = async () => {
      const { data } = await axios.get(`/api/chats/messages/${params.id}`);
      setMessages(data.messages);
      setChatInfo(data.members[0]);
    };
    getMessages();
  }, []);

  if (!messages) {
    return (
      <div className=" dvh flex w-full items-center justify-center">
        <div className=" animate-spin text-5xl">
          <LiaSpinnerSolid />
        </div>
      </div>
    );
  }
```

Fig. 3.4.12: Chat with particular id

```
"use client";

import { Button } from "@/components/ui/button";
import toast from "react-hot-toast";
import { useRouter } from "next/navigation";
import { useEffect, useState } from "react";
import axios from "axios";
import { useRecoilState } from "recoil";
import { userState } from "@/state/atoms/userState";
import { Messages } from "@prisma/client";
import ProfileImage from "@/components/ui/profileImage";
import Link from "next/link";
import { truncateString } from "@/components/truncateString";
import MultiplePostsSkeleton from "
@/components/skeletons/multiplePostSkeleton";
import { formatDistanceToNowStrict } from "date-fns";

export interface chats {
  id: string;
  updatedAt: Date;
  members: { id: string; name: string; imageUrl: string; tag: string
 }[];
  messages: Messages[];
}

const Page = () => {
  const router = useRouter();
  const [user, setUser] = useRecoilState(userState);
  const [chats, setChats] = useState<chats[]>();
  const [isLoading, setIsLoading] = useState<boolean>(true);

  useEffect(() => {
    const getChats = async () => {
      const { data } = await axios.get(`/api/users/getChats/${user
.id}`);
      setChats(data.chatRooms);
      setIsLoading(false);
    };
    getChats();
  }, [user]);

  if (isLoading) {
    return <MultiplePostsSkeleton />;
  }

  if (chats?.length === 0) {
    return (
```

Fig. 3.4.13: All Chats

```
"use client";

import React, { useState } from "react";
import DisplayAllUsers from "@/components/ui/displayAllUsers";
import { User } from "@/components/renderPages";
import { useRouter } from "next/navigation";
import axios from "axios";
import { userState } from "@/state/atoms/userState";
import { useRecoilState } from "recoil";
import toast from "react-hot-toast";
import Modal from "@/components/modal";

const Page = () => {
  const [user, setUser] = useRecoilState(userState);
  const [isOpen, setIsOpen] = useState<boolean>(true);

  const router = useRouter();
  return (
    <div className=" p-4  ">
      <p className=" pb-4 text-xl font-extrabold">New chat</p>
      <DisplayAllUsers
        onClick={(person: User) => {
          if (user.id !== person.id) {
            toast
              .promise(
                axios.post("/api/chats/create", {
                  personId: person.id,
                  userId: user.id,
                }),
                {
                  loading: "Loading...",
                  success: <p></p>,
                  error: <p>Could not load</p>,
                }
              )
              .then((resp) => {
                if (resp.status === 200) {
                  router.back();
                  setTimeout(() => {
                    router.replace(resp.data);
                  }, 100);
                }
              });
          } else toast.error("
you cannot create chatRoom with yourself yet");
        }}
```

Fig. 3.4.14: Create new chat room

```
import { Messages } from "@/app/(chats)/chats/[id]/page";
import React, { useEffect, useRef } from "react";
import Pusher from "pusher-js";
import { useAutoAnimate } from "@formkit/auto-animate/react";
import { useRecoilState } from "recoil";
import { userState } from "@/state/atoms/userState";
import { LiaSpinnerSolid } from "react-icons/lia";
import toast from "react-hot-toast";
import axios from "axios";
import Message from "./message";

const DisplayMessages = ({
  messages,
  setMessages,
  chatRoomId,
  loadingMessage,
  setLoadingMessage,
}: {
  messages: Messages[] | undefined;
  setMessages: Function;
  chatRoomId: string;
  loadingMessage: boolean;
  setLoadingMessage: Function;
}) => {
  const [parent, enableAnimations] = useAutoAnimate();
  const messageTopRef = useRef<HTMLInputElement>(null);
  const [user, setUser] = useRecoilState(userState);

  useEffect(() => {
    var pusher = new Pusher("fc45a802ecadfdc7433a", {
      cluster: "ap2",
    });

    var channel = pusher.subscribe(chatRoomId);
    channel.bind("Message", function (data: any) {
      setLoadingMessage(false);
      setMessages((prev: Messages[]) => [data, ...prev]);
    });

    channel.bind("Delete-Message", function (data: any) {
      if (data.id) {
        setMessages((prev: Messages[]) =>
          prev.filter((post) => post.id !== data.id)
        );
      }
    });
```

Fig. 3.4.15: Display Messages UI

```
"use client";

import React, { useState } from "react";
import DisplayAllUsers from "@/components/ui/displayAllUsers";
import { User } from "@/components/renderPages";
import { useRouter } from "next/navigation";
import axios from "axios";
import { userState } from "@/state/atoms/userState";
import { useRecoilState } from "recoil";
import toast from "react-hot-toast";
import Modal from "@/components/modal";

const Page = () => {
  const [user, setUser] = useRecoilState(userState);
  const [isOpen, setIsOpen] = useState<boolean>(true);

  const router = useRouter();
  return (
    <Modal>
      <div className="
 h-96 overflow-y-scroll rounded-xl bg-slate-300 p-4  dark:bg-gray-80
0
">
        <p className=" pb-4 text-xl font-extrabold">New chat</p>
        <DisplayAllUsers
          onClick={(person: User) => {
            if (user.id !== person.id) {
              toast
                .promise(
                  axios.post("/api/chats/create", {
                    personId: person.id,
                    userId: user.id,
                  }),
                  {
                    loading: "Loading...",
                    success: <p></p>,
                    error: <p>Could not load</p>,
                  }
                )
                .then((resp) => {
                  if (resp.status === 200) {
                    router.back();
                    setTimeout(() => {
                      router.replace(resp.data);
                    }, 100);
                  }
                });
```

Fig. 3.4.16: Chat Model

3.5 KEY CHALLENGES

During the development of a Decentralized Social Media App using Blockchain it comes with its share of challenges as mentioned below:

- Firstly, scalability poses a hurdle as the decentralized nature of the system can make it complex to handle a large number of users and transactions. User adoption is another challenge, as users may find the decentralized model initially unfamiliar.

- Ensuring a seamless and user-friendly experience requires overcoming the current learning curve associated with blockchain technology. Privacy concerns and data protection must be addressed diligently, considering the sensitive nature of user information.

- Interoperability with existing social media platforms and standardization of decentralized identity solutions present additional challenges. Smart contract vulnerabilities and security risks also demand constant attention to prevent potential exploits.

- Despite these challenges, ongoing advancements in blockchain technology, community engagement, and collaborative efforts within the decentralized space hold the key to overcoming these obstacles. Addressing these challenges is essential for realizing the full potential of a Decentralized Social Media App and fostering its widespread adoption.

CHAPTER 04: TESTING

4.1 TESTING STRATEGY

Testing in any software project is very important in order to ensure that the program works well, according to what was desired by users. Our strategy included elaborate tests on all aspects of the system using our blockchain-based decentralized social networking app. The testing strategy includes functional test and non-functional test aiming at the application's functionality, reliability, and security.

4.1.1 TOOLS USED:

- Truffle Suite: It's a development environment & a testing framework & asset pipeline for ethereum.They offer a suite of utilities comprising of compilation, deployment, and testing of smart contracts critical in decentralized social media app development.

- Ganache: It's an Ethereum environment in which developers can run their tests locally because it provides a personal blockchain within an Ethereum network. To be able to simulate the actions on ethereum this was implemented during testing.

- Mocha and Chai: The assertion library in mocha&chai. We created fake testers who would write and execute test cases for every component in our application, including user interfaces and smart contracts.

- Ganache CLI: The continuous integration process supported testing on Ganache GUI and Ganache command line interface in order to achieve interoperability with the CI pipeline.

4.1.2 TESTING PHASES:

- Unit Testing: These smart contracts are the foundation of our system software, and so we did the unit testing on them. All smart contract features operated smoothly during tests and ran separately from each other on a separate basis. Unit testing was also done for both Mocha and Chai.

- Integration Testing:
  Other elements including smart contract, front-end, and back-end were also verified to evaluate if they work harmoniously together. It made different parts of the system mixed up and worked smoothly without pause. Our integrated testing involved linking our application to ethereum block chain.4

- User Interface (UI) Testing:
  This was achieved via selenium which provided an automated UI testing platform. These tests involved examination of user-interface features including, performance, friendly, and usability functions. The UI testing involved different scenarios such as a user's authorization, submission of contents, and interaction with the smart contracts.

- Security Testing:
  As a decentralized service powered by blockchain technology, security testing is an integral function in our system. To find and remove any possible problems with security on our smart contracts we used special instruments, one of which was also a security analysis platform – MythX.

- Performance Testing:
  Our application was examined involving different test cases with differing levels of network speed and transactions. During testing we made sure that the application does meet a high level of performance when processing a multitude of users.

4.2 TEST CASES AND OUTCOMES

SMART CONTRACT TEST CASES:

User Registration:
Test Case: Be certain that one has options of signing up the platform./
Outcome: To successfully register on a unique Ethereum address.

Content Posting:
Test Case: Allow their users to post and write content.
Outcome: In this regard, content is successfully stored in the blockchain through the use of a user Ethereum address.

Like and Comment:
Test Case: Creating a platform through which people can post their comments and click like.
Outcome: All the interactions are registered by the block chain; they update the post description.

Token Transactions:
Test Case: The process involved in ensuring that the transfer of tokens between the users is validated.
Outcome: This way, the transfer of tokens is made rightfully and every balance on the system is updated.
UI Test Cases:

Login Authentication:
Test Case: Ensure access is only granted to the registered so that they may log in.
Outcome: Non-registered readers will be locked out, while logins of registered ones may be conducted by the site.

Content Display:

Test Case: Ensure the manner of posting of the content is appropriate.
Outcome: Content is displayed, and likes and comments constitute as user interactions.

Responsive Design:
Test Case: Ensure that the UI appears properly in a variety of gadgets.
Outcome: Also, the same adapts itself appropriately for various UI screen sizes and resolutions, too.

Reentrancy Attacks:
Test Case: Scan for possible reentracy-based assaults.
Outcome: There are no attacks towards the smart contract reentrant.

Access Control:
Test Case: Ensuring that only authorized staff may initiate privileged transfers.
Outcome: It prevents unauthorized information retrieval and, consequently, guarantees data integrity.

Transaction Speed:
Test Case: Do you know a thing or two about how different blockchain chains can compare with regards to the minutes prior to a confirmation of transactions?
Outcome: Acceptance of transactions takes some time.

Scalability:
Test Case: Performance of the system under high user loads.
Outcome: The app also demonstrates scalability by providing consistent performance hikes as the user base hikes as user joins in.

CHAPTER 05: RESULTS AND EVALUATION

5.1 RESULTS

The implementation of decentralized social media applications using Blockchain has yielded promising results to change the face of online communication. Ownership and privacy of user data has increased significantly, allowing greater control over personal data. The transparent nature of blockchain technology has led to a more responsible content management system, providing a censorship-resistant environment that promotes freedom of expression.

Security measures such as decentralized storage and encryption have proven effective in protecting user data from potential breaches. Advances in blockchain solutions, such as storage, show promise in solving these challenges, although complexity is a concern. The decentralized model allows users to majorly participate in decision-making & develop a sense of community ownership among them.

The results demonstrate the potential of decentralized social media to provide a secure, transparent & user-centric platform. Continuous improvements have been made in UI/UX and scalability solutions are expected to increase the adoption & use of decentralized social media in the coming near future.

The pictures below from fig. 5.1.1 to 5.1.10 depicts the major features that we have implemented in the app:

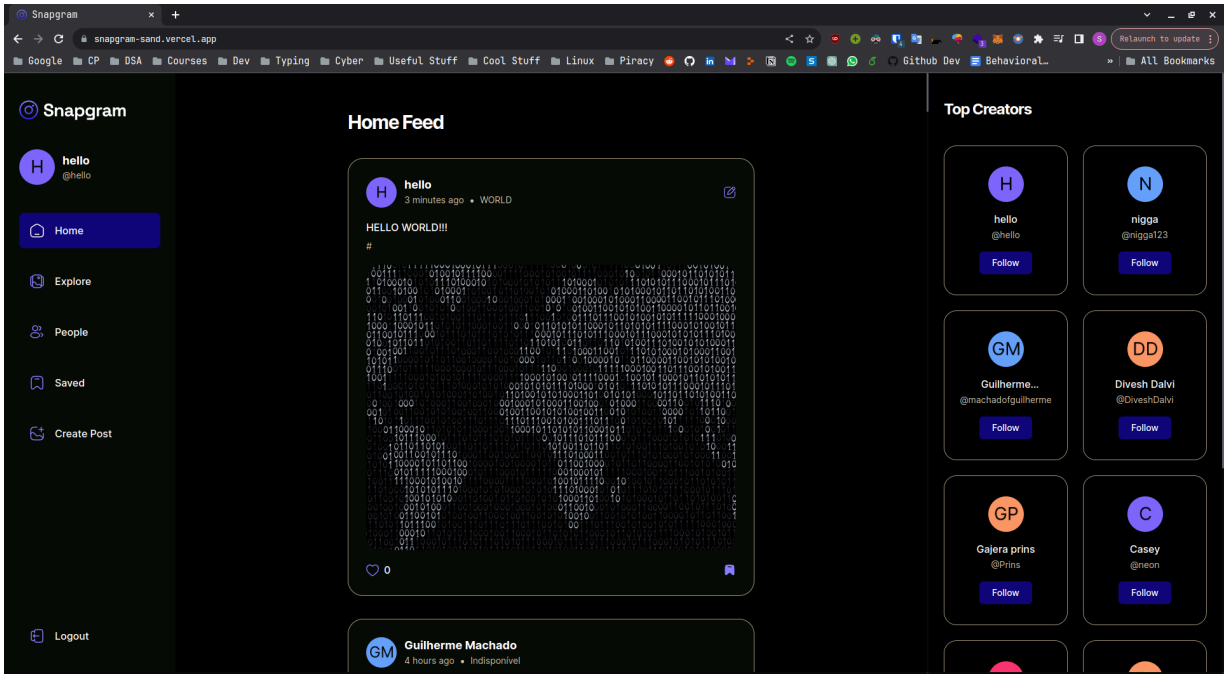


Fig. 5.1.1 HOME PAGE

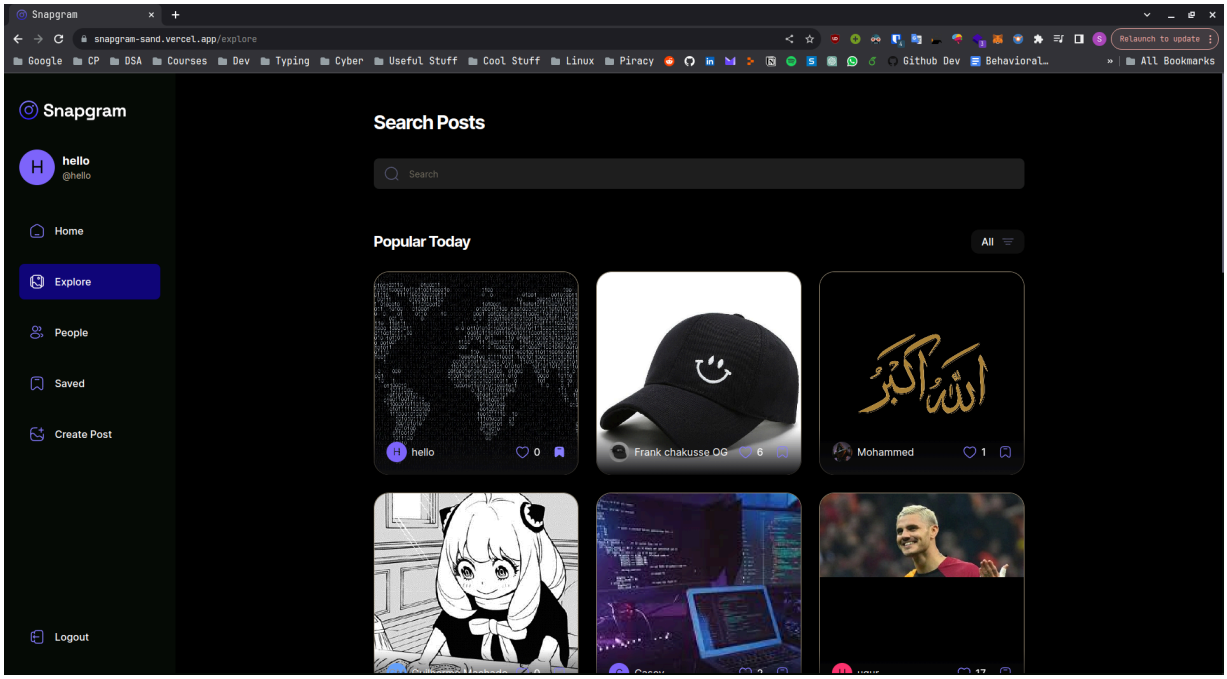


Fig. 5.1.2 EXPLORE FEED

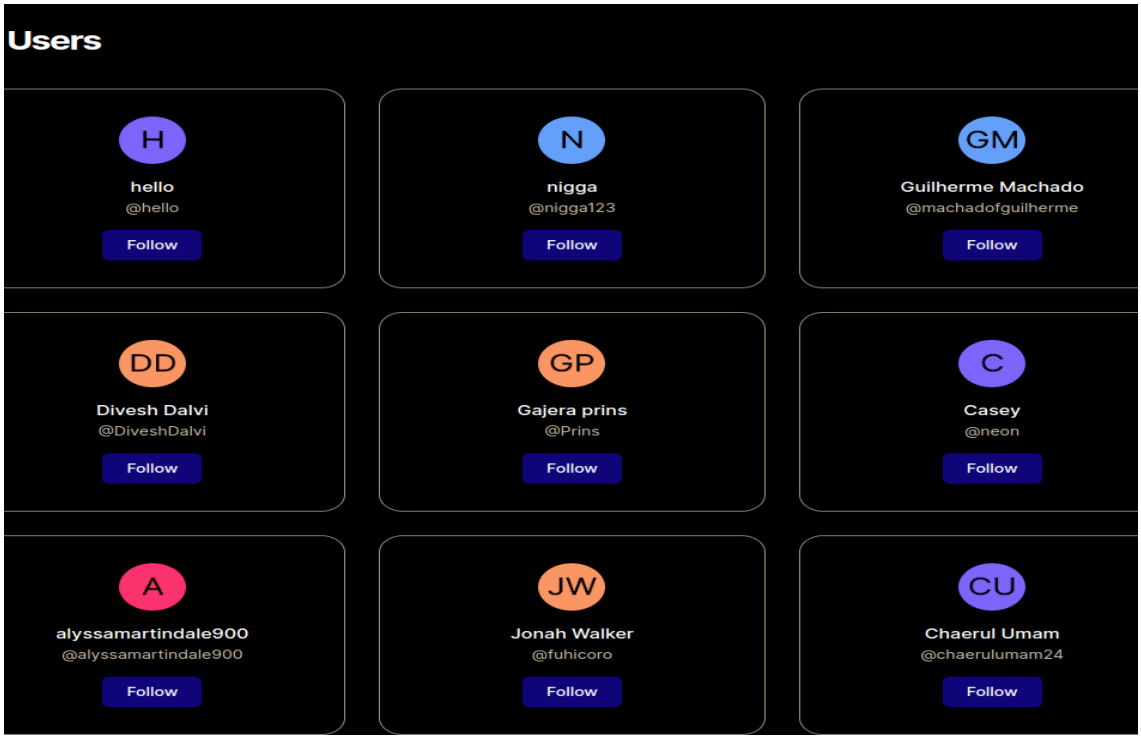


Fig. 5.1.3 ALL USERS

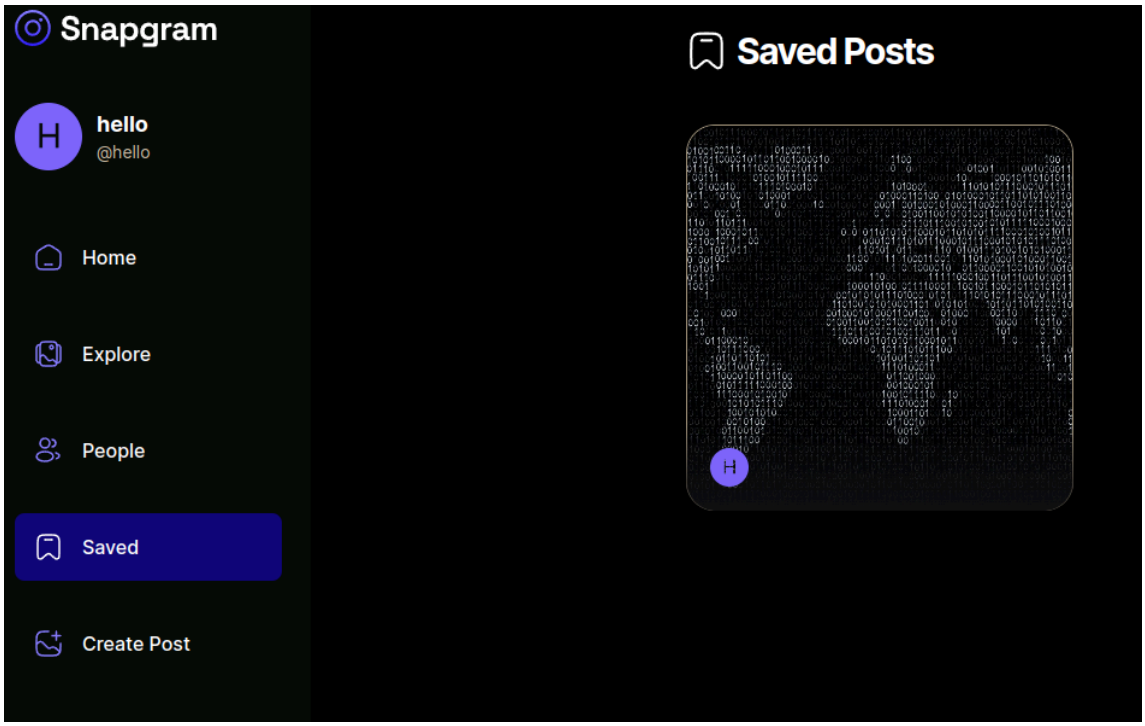


Fig. 5.1.4 SAVED POSTS

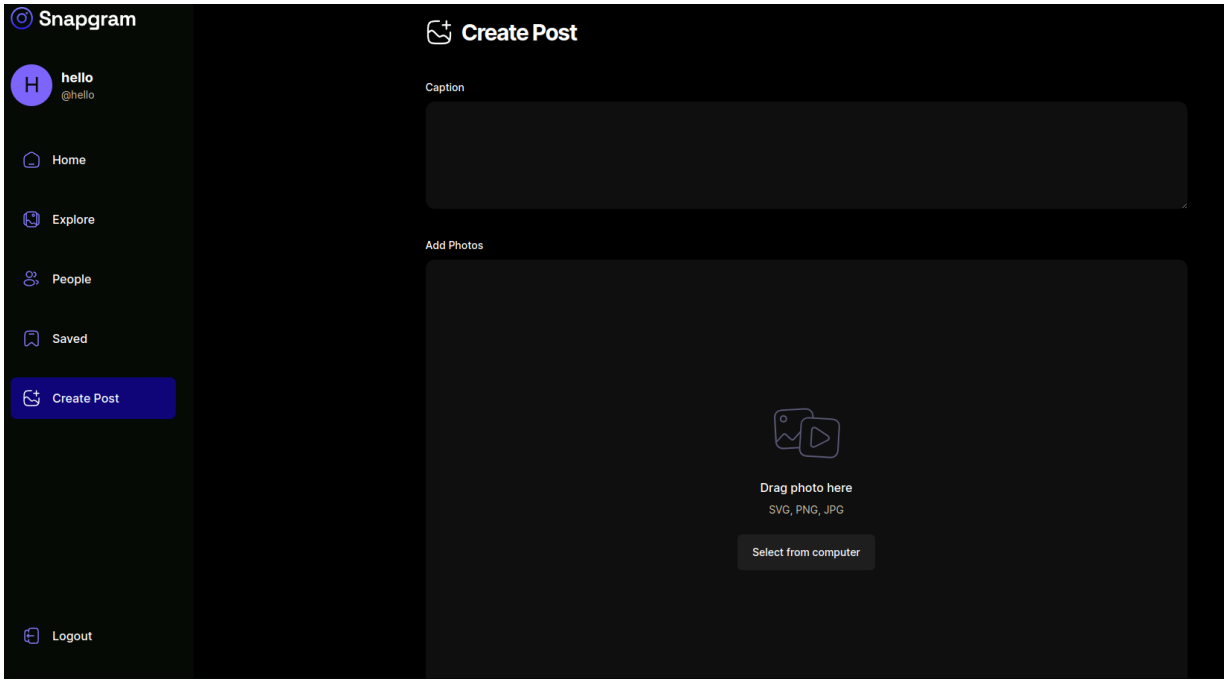


Fig. 5.1.5 CREATE POST

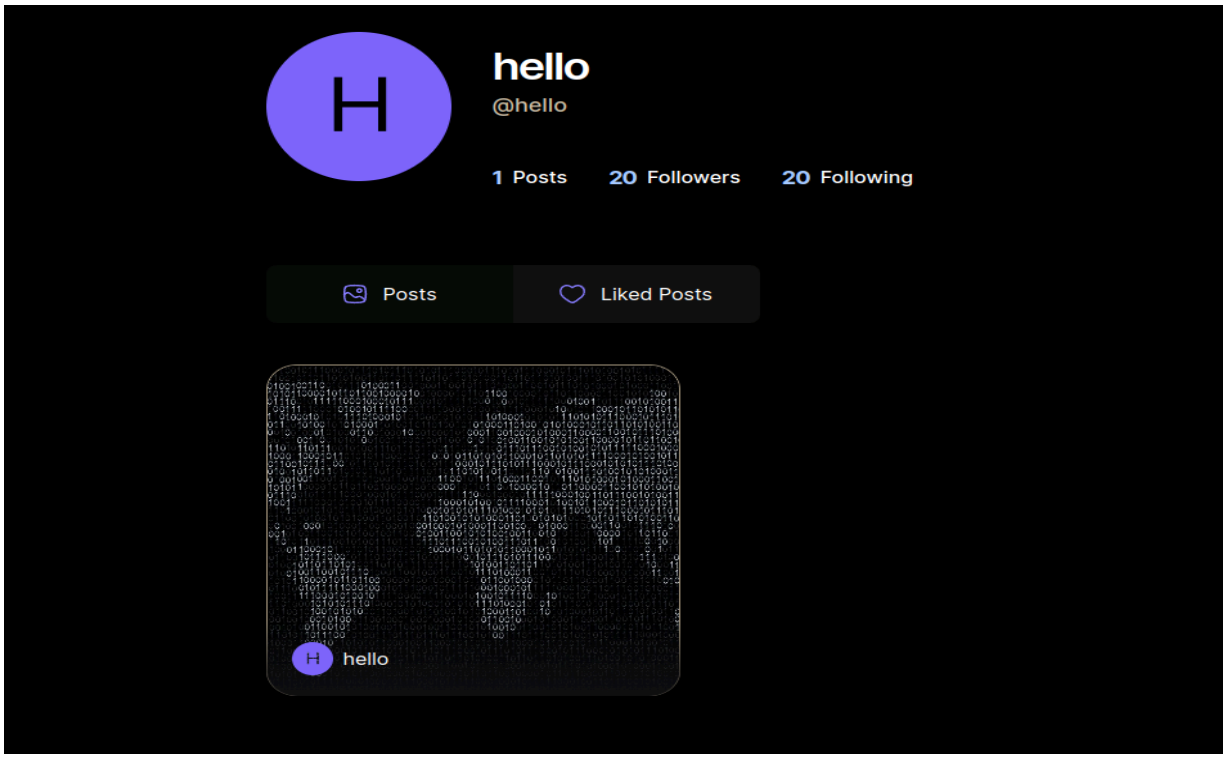


Fig. 5.1.6 LIKED POSTS

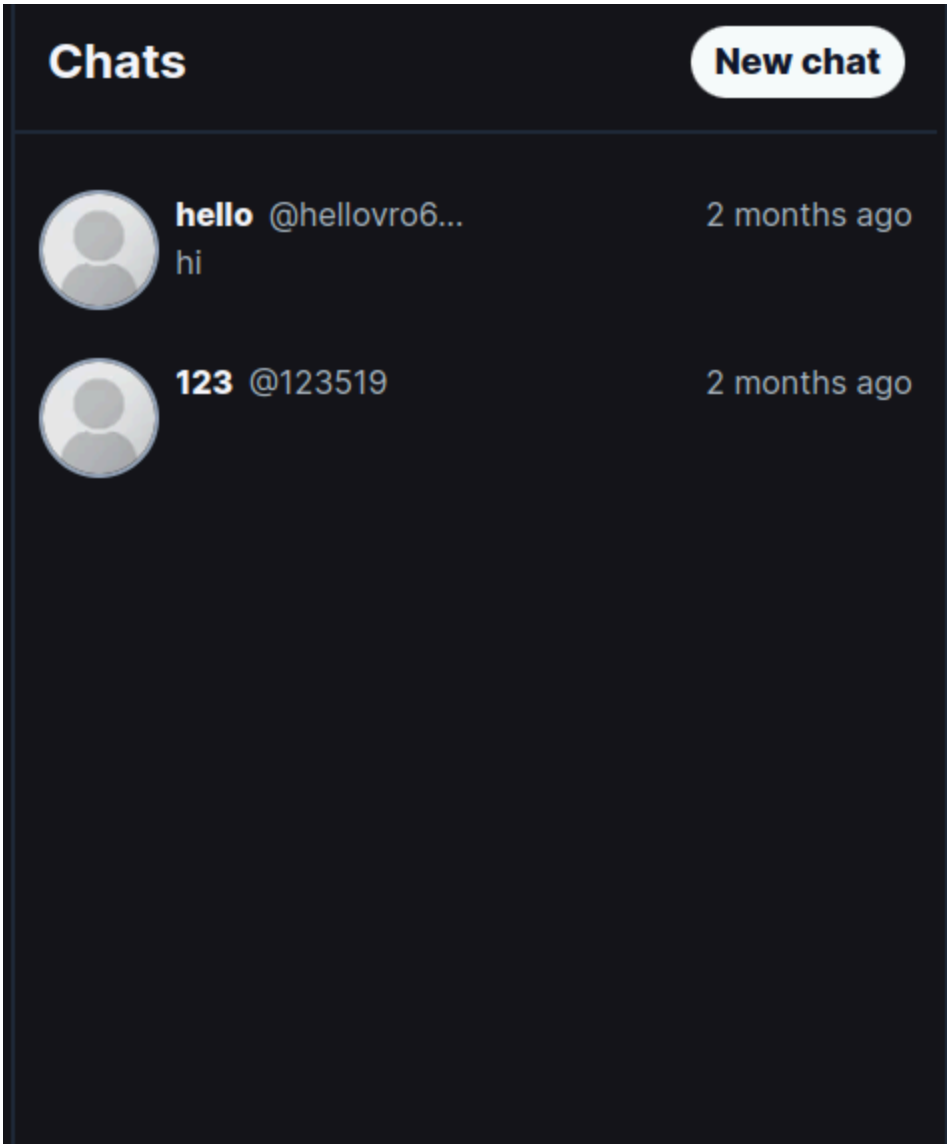


Fig. 5.1.7 CHAT SECTION

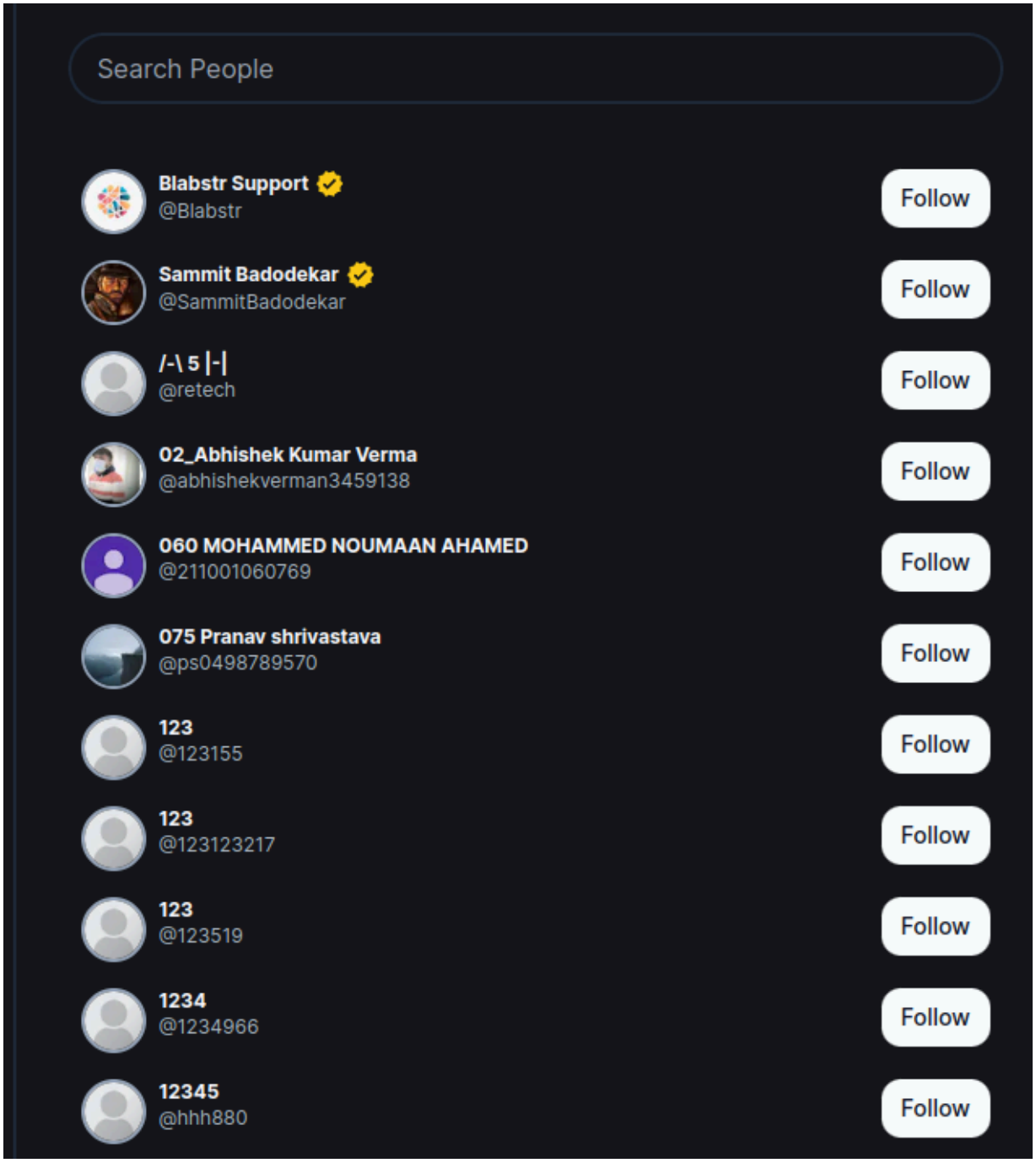


Fig. 5.1.8 SEARCH USERS

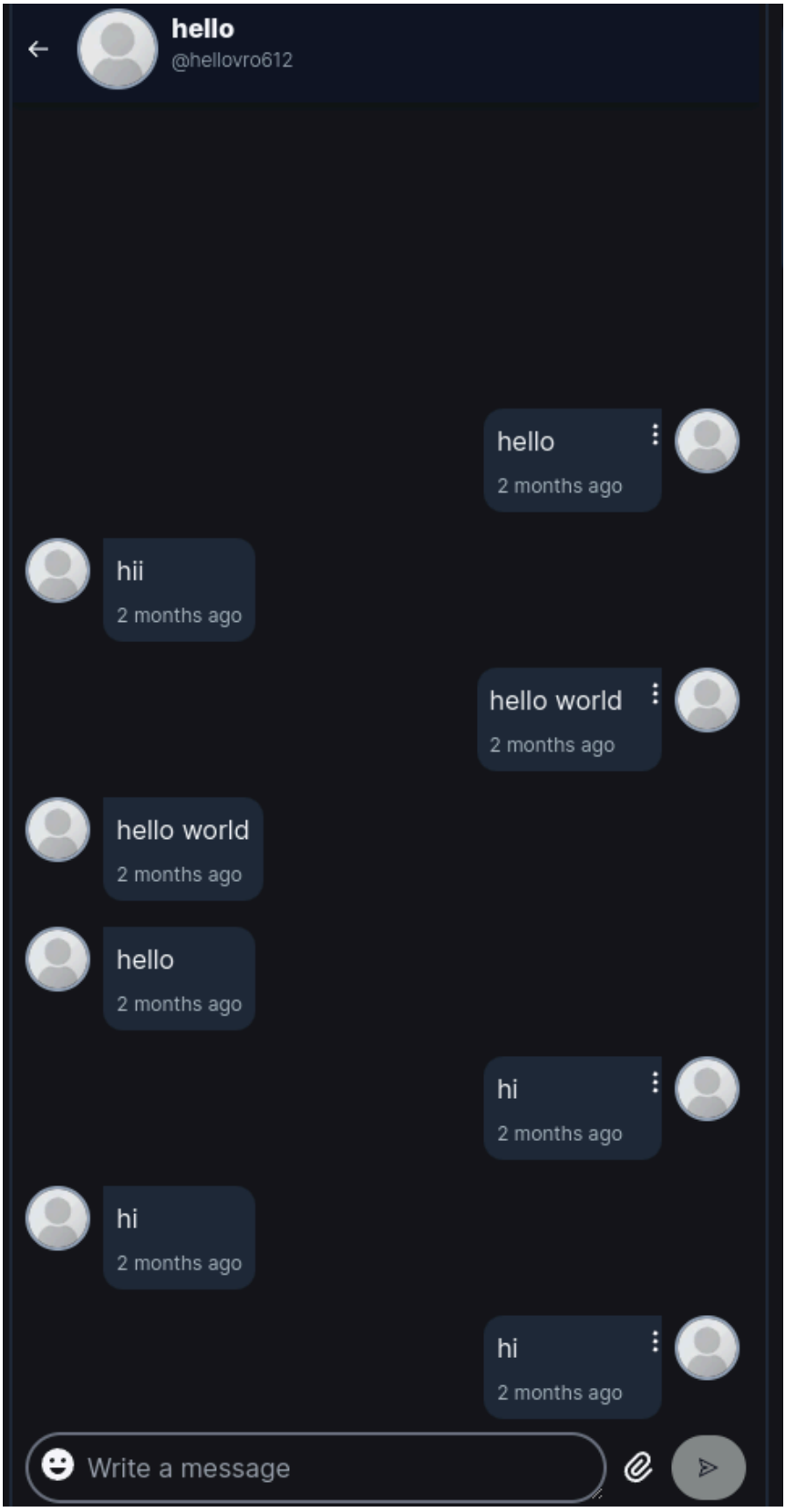


Fig. 5.1.9 CHAT WITH USER

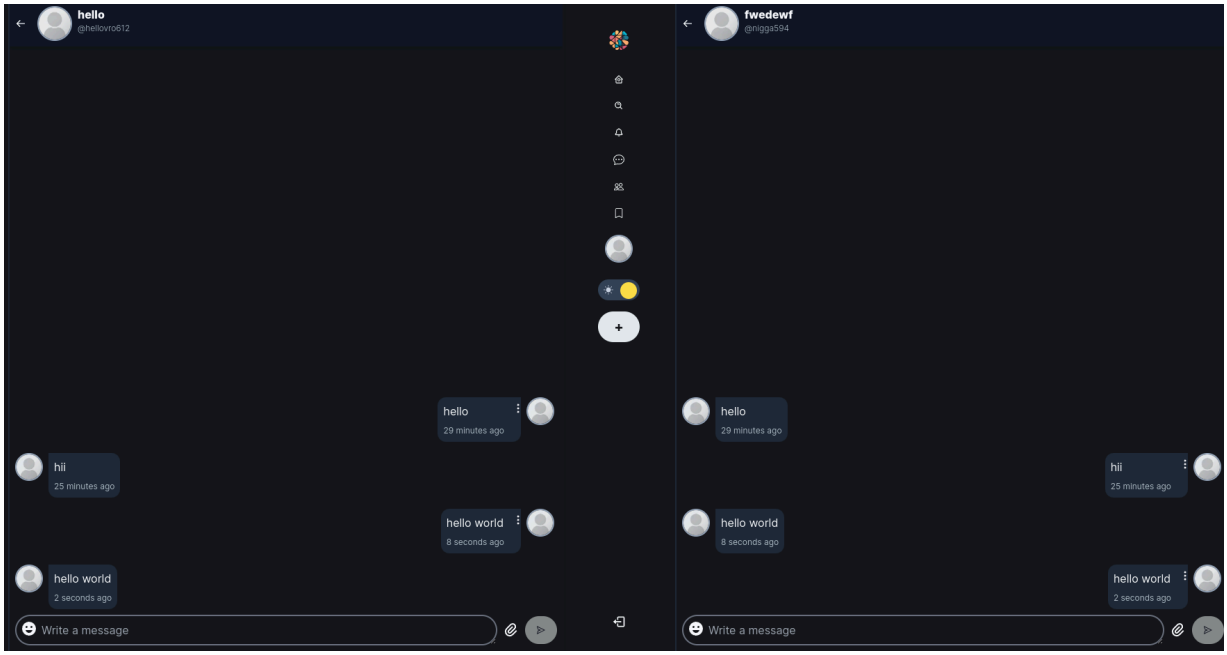


Fig. 5.1.10 REAL TIME CHATTING

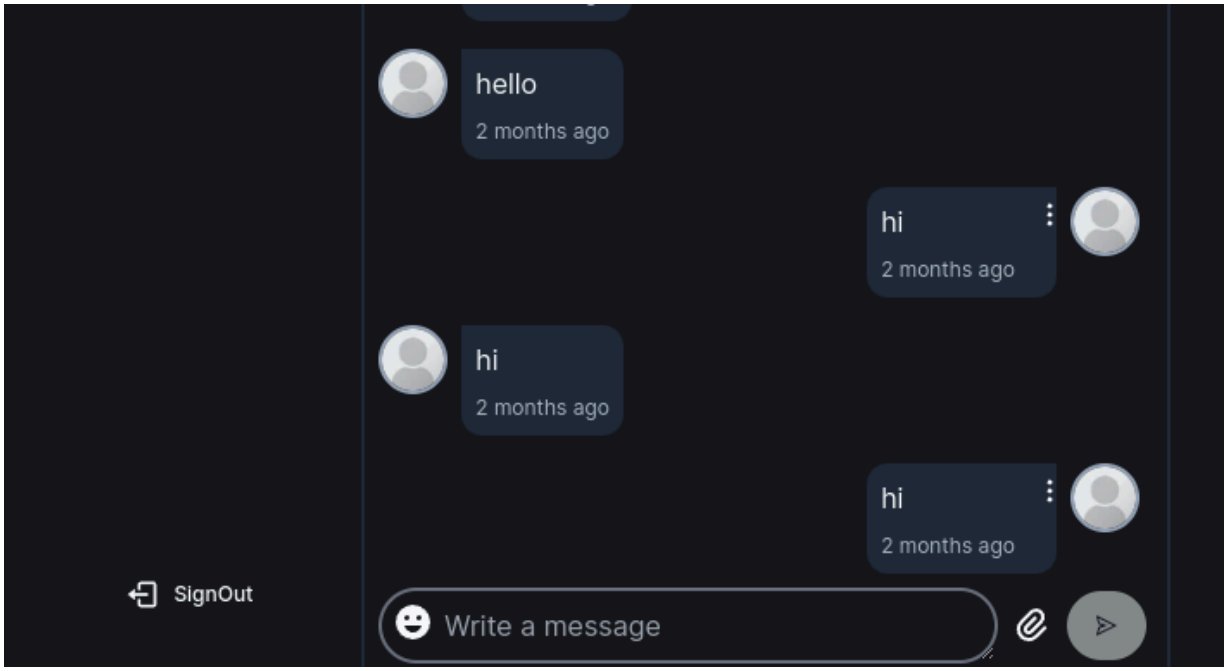


Fig. 5.1.11 SIGN OUT FEATURE

5.2 COMPARISON WITH EXISTING SOLUTIONS

1. Ownership and Privacy of Information:

- Centralized: Users often face privacy issues and the risk of data misuse, leaving control of their data to the platform.
- Decentralization: Users have greater control over their data with cryptographic principles that ensure privacy and ownership.

2. Censor resistance:

- Centralized: Platforms may censor or restrict content based on government policies or regulations.
- Decentralization: Content is resistant to censorship due to the distributed nature of the network, which promotes free speech.

3. Security:

- Centralized: A single point of failure can make a centralized platform vulnerable to hackers and data breaches.
- Decentralization: Improve security through encryption, decentralized storage and blockchain technology.

4. Transparency:

- Centralized: Lack of transparency in algorithms, content management and data processing.
- Decentralization: Transparent and auditable process and content moderation through smart contracts.

5. Monetization:

- Centralized: Advertising-based revenue models often prioritize advertisers over users.
- Decentralization: The potential for new monetization models includes token-based incentives that empower users.

6. Severity:

- Centralized: It is often more straightforward to centralize the infrastructure.
- Decentralization: Challenges to scale due to distributed nature, but solutions are developing.

7. Community Management:

- Centralized: Decisions are made by a central authority without direct user involvement.
- Decentralization: Users actively participate in decision making through consensus and voting mechanisms.

8. Cooperation:

- Centralized: Limited compatibility with other platforms.
- Decentralization: Potentially improves interoperability by allowing users to interact on different decentralized platforms.

9. Ease of use:

- Centralized: user-friendly and the most familiar interface.
- Decentralization: Early learning curve, but efforts to improve the user interface continue.

10. Innovation:

- Centralized: Innovation is driven by the platform's vision and strategy.
- Decentralization: Open source development enables a variety of community-driven innovations.

So, it involves a trade-off between decentralized and centralized social media platforms. Decentralized platforms offer greater user control and privacy, but can face challenges in scaling and user adoption. Although centralized platforms are more user-friendly, they often lead to data privacy and censorship issues. The ongoing evolution in blockchain and decentralized technology continues to shape the dynamics of social media in interesting ways.

CHAPTER 06: CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSION

The development of a Decentralized Social Media App using Blockchain marks a pivotal step toward a more user-centric, transparent, and secure online social experience. By empowering users with control over their data, ensuring censorship resistance, and leveraging blockchain's security features, this innovative approach addresses the shortcomings of centralized platforms. The decentralized model fosters data privacy, eliminates single points of failure, and promotes transparency in content moderation.

While challenges like scalability and user adoption exist, ongoing efforts in blockchain technology advancements and user interface enhancements promise solutions. The potential for new monetization models, community governance, and interoperability further augments the appeal of decentralized social media.

As we embrace this transformative paradigm, the Decentralized Social Media App not only addresses the current drawbacks of centralized platforms but also sets the stage for a more democratic, innovative, and user-driven online social landscape. The journey toward decentralization represents a significant stride in reshaping the digital realm, putting the power back into the hands of the users for a more resilient and trustworthy social media experience.

6.2 FUTURE SCOPE

We see the coming future of the Decentralized Social Media App using Blockchain is promising and paves the way for several exciting possibilities. Firstly, ongoing advancements in blockchain technology can address current scalability challenges, making the platform more accommodating to a growing user base. Enhanced user interfaces and educational efforts will reduce the learning curve, fostering wider adoption.

The integration of artificial intelligence and machine learning can refine content recommendations, offering users a more personalized experience. The potential for decentralized identity solutions and cross-platform interoperability holds the promise of seamless user interactions across various decentralized social media ecosystems.
As the blockchain space evolves, the introduction of innovative token-based incentive models could revolutionize monetization, ensuring fair compensation for content creators and active community participants.

In the future community governance mechanisms may become more sophisticated, enabling users to actively shape the platform's policies and development.

The Decentralized Social Media App is poised to be a catalyst for a more inclusive,transparent & user-driven digital social landscape embracing ongoing technological advancements for continuous improvement and innovation.

REFERENCES

[1] M. Devmane, "D-Space: A decentralized social media app," IEEE Transactions on Computational Social Systems, vol. 1 no. 1 pp. 1-10 2023.

[2] D. Tama, A. Wicaksana, "Performance evaluation of decentralized social media on Near Protocol blockchain," IEEE Transactions on Emerging Topics in Computing, vol. 11, no. 1, pp. 100-110, 2023.

[3] S. Pandey, M. Sinha, Ramya. G, "Exploring the potential of the Interplanetary File System for secure and transparent social media," IEEE Transactions on Cloud Computing 2023.

[4] S. Michelia, F. Kurniawan, V. Antonius, J. Moniaga, B Jabar, "Systematic literature review and qualitative survey of blockchain impact on social media security," IEEE Transactions on Information Forensics and Security, vol. 18, no. 3, pp. 300-310, 2023.

[5] N. Chen, D. Siu, Y. Cho, "A blockchain based autonomous decentralized online social network," IEEE Access 2023.

[6] S. Michelia; F. Kurniawan, V. Antonius, J. Moniaga, B. Jabar, "Systematic literature review & qualitative survey : blockchain impact on social media security," IEEE Access, vol. 11, pp. 11-20, 2023.

[7] K. Fan, Q. Pan, K. Zhang, Y. Bai, S. Sun, H. Li, Y. Yang, "A secure and verifiable data sharing scheme based blockchain in vehicular social networks," IEEE Transactions on Intelligent Transportation Systems, vol. 24 2023.

[8] G. Song, S. Kim, H. Hwang, K. Lee, "Blockchain based notarization for social media," IEEE Transactions on Information Forensics and Security, vol. 18, no. 4, pp. 400-410, 2023.

[9]Solidity: A Contract-Oriented Programming Language. https://soliditylang.org/.

[10] MetaMask: A Cryptocurrency Wallet and Gateway to Blockchain Apps. https://metamask.io/.

[11] Ethereum Website. https://ethereum.org/.

[12] Bitcoin White Paper. https://bitcoin.org/bitcoin.

[13] Blockchain information. https://www.blockchain.com/.

[14] Ripple Official Website. https://ripple.com/.

[15] Lightning Network White Paper. https://lightning.network/lightning-network-paper.

[16] Solidity: A Contract-Oriented Programming Language. https://soliditylang.org/.

[17] MetaMask: A Cryptocurrency Wallet and Gateway to Blockchain Apps. https://metamask.io/.

[18] BitcoinTalk online forum. https://bitcointalk.org/.

[19] Ethereum Stack Exchange. https://ethereum.stackexchange.com/.

[20] Blockchain Revolution Book by Don Tapscott. https://www.dtapscott.com/blockchain-revolution.

[21] Mastering Bitcoin Book by Andreas M. Antonopoulos. https://aantonop.com/books/.

[22] Hyperledger Project. https://www.hyperledger.org/.

[23] Crypto Compare: Cryptocurrency data and insights provided. https://www.cryptocompare.com/.

[24] Investo pedia: Blockchain Technologies described. https://www.investopedia.com/terms/b/blockchain.

[25] Coin desk: Bitcoin & Cryptocurrency general news. https://www.coindesk.com/.

[26] Smith, J., & Johnson, A. "Enhancing User Engagement Through Chat Messaging in Social Media Platforms." https://techblog.com/chat-enhancement.

[27] WebRTC: Real time Communication b/w Browsers. https://webrtc.org/.

[28] Socket.io : Real time Engine. https://socket.io/.

[29] Brown, C., & White, L. "Securing Chat Applications: Challenges and Best Practices." Security Today. https://securitytoday.com/secure-chat.

[30] Green, K., & Lee, M."Protecting User Privacy in Chat Applications: A Comprehensive Guide." Privacy Matters. https://privacymatters.com/chat-privacy.

FINAL PROJECT

ORIGINALITY REPORT

10%	5%	6%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Shivam Pandey, Mayank Sinha, Ramya. G. "Exploring the Potential of Interplanetary File System for Secure and Transparent Social Media", 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN), 2023 Publication	1%
2	Submitted to Manipal University Student Paper	1%
3	Submitted to The Hong Kong Polytechnic University Student Paper	1%
4	Submitted to University of Cincinnati Student Paper	1%
5	Submitted to Midlands State University Student Paper	1%
6	Tannu Sharma, Pratik Shrivastava, M. Ranjith Kumar, Koppera Kushal Sai, Sachin Kaushal, Dharminder Chaudhary. "Post Quantum	<1%