

Project Report

On

D.APP: Decentralized Chat App

*Submitted in partial fulfillment of the
requirement for the award of the degree of*

Bachelor Of Technology



**Under The supervision Of
Mr. Gaurav Rawat
Assistant Professor**

Submitted By

**Shashank Bora – 19SCSE1010670
Devansh Singh Tomar– 19SCSE1010083**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
May, 2023**



**SCHOOL OF COMPUTING SCIENCE AND
ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA**

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the entitled “**D.APP: Decentralized Chat App**” in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of **February,2023 to May, 2023**, under the supervision of **Mr. Gaurav Rawat, Assistant Professor, Department of Computer Science and Engineering**, of School of Computing Science and Engineering , Galgotias University, Greater Noida.

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Shashank Bora – 19SCSE1010670

Devansh Singh Tomar – 19SCSE1010083

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. Gaurav Rawat
Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **Shashank Bora – 19SCSE1010670, Devansh Singh Tomar – 19SCSE1010083** has been held on _____ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Project Coordinator

Signature of Dean

Date:

Place:

Abstract

The growing concern for personal privacy is on the verge to reach the threshold, thus is the number of companies in the coarse market of social media which can have an eruptive response if the concern for personal privacy is precluded. In resent there have been many advancements the field of encrypted or secure private messaging like Telegram, Signal and WhatsApp but as these social media giants move mainstream, they are debased on their primary goal i.e. personal privacy.

With the collected data we are developing a complete decentralized messaging app which will not be controlled by any mainstream tech companies like amazon AWS or Google clouds.

Our main objective while developing this whole project was to prioritize personal privacy and secure user data. The data from the user should not be used for any purpose whatsoever thus even for registering to our web app no personal information is required from the user which provides complete anonymity for the user.

It works on “GUN” i.e. an ecosystem of modular tools. Graphs take you beyond just immutable hashes, they let you do real time multiplayer updates on any type of data. The decentralized database for developers.

Our goal is to build complete decentralized encrypted messaging and for other social media purposes.

By not using (connecting) any database for our project and using Gun modules which helps in connecting peers which itself proves its decentralized framework.

We have developed a web app which is easily compatible with **ios/android/windows** and any other operating system with a web browser with the help of web3.0 thus our web app is accessible with just a link and does not require any system installation.

Table of Contents

Title	Page No.
Candidates Declaration	II
Abstract	IV
Contents	
List of Table	VI
List of Figures	VII
Acronyms	VIII
Chapter 1	
Introduction	9
1.1 Disadvantages of current system	10
1.2 Merits of proposed system	11-12
Chapter 2	
Literature Survey	13-15
Chapter 3	
Functionality/Project design	16-28
Chapter 4	
Results and Discussion	29-31
Chapter 5	
Conclusion and Future Scope	
5.1 Conclusion	32
5.2 Future Scope	32
Reference	33

List of Table

S.No.	Caption	Page No.
1	Introduction	9
2	Disadvantages of current system	10
3	Merits of proposed system	11-12
4	Literature Survey	13-15
5	Understanding previous models	16
6	Whatsapp Architecture	17
7	Block chain messaging	18-19
8	Project design	20-28
10	Result and Interface	29-31
10	Conclusion	32
11	References	33

List of Figures

S.No.	Title	Page No.
2.1	Cellular text messaging	14
2.2	Whatsapp architecture	16
2.3	Whatsapp architecture	17
2.4	Block chain messaging	19
3.1	Source code 1.0	21
3.4	Source code 2.0	24
3.5	Source code 3.0	25
4.1	Web app interface 1.0	29
4.2	Web app Interface 2.0	30

Acronyms

B.Tech.	Bachelor of Technology
M.Tech.	Master of Technology
BCA	Bachelor of Computer Applications
MCA	Master of Computer Applications
B.Sc. (CS)	Bachelor of Science in Computer Science
M.Sc. (CS)	Master of Science in Computer Science
SCSE	School of Computing Science and Engineering

CHAPTER - 1

Introduction

The most convenient thing internet has provided is “Instant messaging” and is used so often that it has become a norm in most Asian countries messaging is done via internet as the cellular phone services are not cost effective for daily text messaging but the internet messaging or instant messaging is quite on the cheaper side in countries like India and most of the South Asian Subcontinent and countries like Canada and some part of European hence making it an absolute flourishing market. In other countries the situation is not very appalling as musing at it seems in countries like USA cellular services are quite cheap thus they are custom to text via text messaging.

But things take turn when “Social Media” was introduced which without a doubt broke the all the barriers and made the world its community and which clearly shows from the data that 53.6% of the world’s population use social media which is integrated with instant messaging and other features.

When things come to privacy while sending or receiving text messages whether it’s on social media or on any messaging app the companies have kept the user in dark on most of the times or called out in public due to various data/chat leaks and selling data to various buyers unethically for various reasons.

Not only in messaging big companies like AWS and Google cloud sell their services to other companies thus only screened companies hold proprietorship for the data.

We are making an app which will not be a faction of the tech giants and will be solely independent.

Decentralized messaging make your messages far more secure than any other messaging system ever created. Adding encryption is only viable option for third parties like hackers not for the company that is providing it.

Here comes D.A.P.P (Decentralized Chat App) In Our app a subset of each users data will be stored with each user so making all the peers in the network the entire database for the app thus no one will be in actual control just like block chain but faster and without the need of miner thus saving a lot of time and money.

Disadvantages of Current System

Social media data is widely used in conservation science to study human interactions with the environment. User-generated content, such as images, video, text, and audio, and related metadata can be used to test such interactions. Social media numbers provide free access to user-generated social media platforms. However, as with any research involving people, social media-based scientific research requires compliance with the highest standards of data privacy and data protection, even if the data is publicly available. In the event that social media data is misused, the risks to privacy and well-being of individual users could be significant. We have investigated the legal basis for the use of social media data while validating the rights of data heads through research based on the European Union General Data Protection Regulation. Risks associated with the use of social media data in research include incorrect and intentional identification of error and may cause psychological or physical harm to the target person. Collecting, storing, protecting, sharing, and managing social media data in a way that prevents potential risks to the affected users, one must reduce data, make data anonymous, and follow a strict data management process. Risk-based methods, such as monitoring the impact of data privacy, can be used to identify and reduce privacy risks for social media users, to demonstrate accountability and compliance with data protection legislation. We recommend that conservation scientists carefully consider our recommendations in establishing their research objectives in order to facilitate responsible use of social media data in conservation science research.

Social media users agree to a set of terms and conditions of a platform, it may be argued that their publicly available data should be considered equally publicly available for use in scientific research. However, not all social media users may be aware they agreed to terms and conditions of social media platforms, and not all platforms allow the use of their data for scientific research without separate agreements (Zimmer **2010**). Obtaining informed consent from social media users whose data are the subject of a research investigation is a way to address this issue. However, gaining informed consent is virtually impossible when data from hundreds of thousands of social media users are used (Ess **2019**). Therefore, it is of foremost importance that all identifiable content of social media users be removed to protect user privacy. Protecting privacy becomes especially crucial when researching sensitive topics that may pose risks (e.g., embarrassment, reputational damage, or prosecution) to the social media users. A key aspect of social media data analytics is, therefore, how to consider legal aspects and avoid risks to and ensure privacy of social media users during data preprocessing, analysis, publication, and sharing.

MERITS OF PROPOSED SYSTEM

Decentralized social networks in our case D.APP operates on independently run servers, rather than on a centralized server owned by a business. Mastodon is one example of a decentralized social network. It is based on open-source software and functions a lot like Twitter. Another example is Steem, which runs on a social block chain. Block chain technology allows data entries to be stored in servers anywhere in the world. It fosters transparency, as the data can be viewed in near real time by anyone on a network.

Decentralized social networks give users more control and autonomy. An individual can set up their social network and determine how it operates and what users can say. Instead of having content monitored by a corporation, the founder of a federated social network can establish the terms of acceptable behavior for the site.

Social media promotes connectivity, community building, and knowledge sharing. People can use social media to drive social and political change, bring awareness to important issues, raise funds for those in need, and promote their businesses. However, social media's ugly side can include cyberbullying, political misinformation, and even criminal activity. Because decentralized social networks are largely immoderate, both the positive and negative outcomes become more extreme.

User control

Corporate entities control major social media sites, and a small group of people within these companies sets the rules of engagement. This has raised concerns about free speech and censorship among users. Last year, Facebook enacted high-profile bans on individuals from all sides of the political spectrum, from Louis Farrakhan to Alex Jones. Banning violent, hateful, and dangerous messaging helps protect social media users from malicious online activity, but some believe the bans run contrary to ideals of free speech.

Free Speech

A decentralized social network allows users more control. Unlike centralized social networking platforms, federated networks foster independence without a central authority. Benefits include censorship resistance, ownership over personal data, and improved control over user-generated content. In other words, users do not accept censorship and insist on having the final say on their content. This means no one else, whether a corporation or site administrator, can make modifications to content created by users. No one can remove content generated by users, either.

In a federated network, no single group can dictate other groups' rules. For example, anyone on Mastodon can run their own social media site without a central authority, meaning them (and other users) can post anything they want without worrying about having their post taken down. A downside of this structure is that hate groups also have the freedom to launch their own social media sites. While individuals can block these groups, they cannot prevent them from engaging on the network.

Personal Data and Privacy

User concerns about control of their personal data have led to the establishment of the General Data Protection Regulation (GDPR) in Europe. The legislation considers users “data controllers.” Social media companies are known as “data processors.” The GDPR definition of data controller means that users own their own data. By law, companies must hand over more control of personal data to users, at least those based in Europe. Companies are penalized for not following GDPR regulations.

Decentralized social networks have provided another answer to data privacy and security. On federated social networks, users can create accounts without having to link to real-world identities, like email addresses or phone numbers. Furthermore, these networks often rely on public-key cryptography for account security, rather than relying on a single organization to protect user data.

While this can create advantages from a data security perspective, it also presents challenges. For example, bootstrapped federated social networks may shut down because of a lack of funds, causing users to lose their data and connections. In this instance, users have no simple way to reconnect with others on the network because federated networks do not keep records of personal data on servers. In terms of privacy, these platforms do not necessarily encrypt data, which means that private messages may be visible to administrators.

CHAPTER-2

Literature Survey

D.A.P.P which is Decentralized Chat App is a decentralized messaging web app made on web3.0. Most messaging apps have an uncanny resemblance due to fixed features as it aims to be ease of use otherwise it can be a social media which is not our goal.

The most used and trending means for instant messaging is obviously text messaging via cellular network, every cell tower presides over an area of land, where it receives and transmits radio waves. When a text message is written, it is transmitted as binary code using a particular frequency of radio waves specific to that user. The signal is received by a nearby cell tower, which then directs the information to be transmitted by another tower near the intended recipient. Cell towers are connected by a network of computers that constantly monitor the locations of cell phones so that each phone can communicate with the tower closest to it.

Then coming to the new era of internet messaging using web on social media or any other medium.

Data travels across the internet in packets. Each packet can carry a maximum of 1,500 bytes. Around these packets is a wrapper with a header and a footer. The information contained in the wrapper tells computers what kind of data is in the packet, how it fits together with other data, where the data came from and the data's final destination.

Gun allows you to group multiple records and add them to a set. A Gun's set, is a mathematical set with unique unordered items. Let's say we have two nodes, and we want to create a group for them. First, we create the group node (a set) and then we use the set method to add other nodes or plain objects to it. Note that the plain objects, just like update operations, will be converted to Gun nodes automatically.

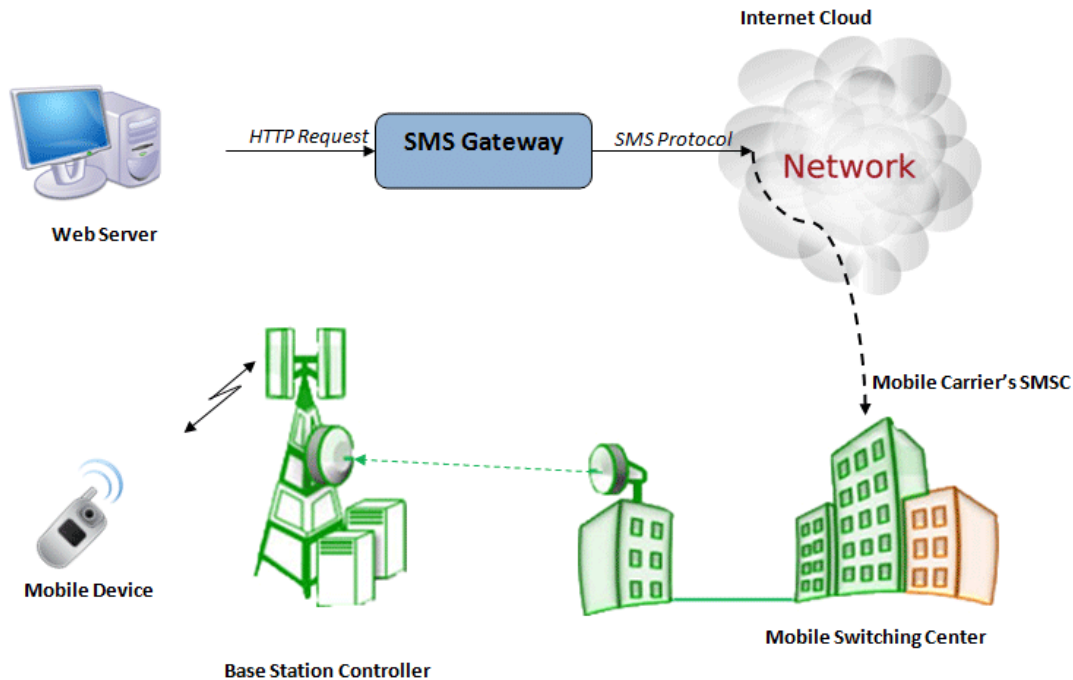


Figure 2.1

When you send an e-mail to someone, the message breaks up into packets that travel across the network. Different packets from the same message don't have to follow the same path. That's part of what makes the Internet so robust and fast. Packets will travel from one machine to another until they reach their destination. As the packets arrive, the computer receiving the data assembles the packets like a puzzle, recreating the message.

All data transfers across the Internet work on this principle. It helps networks manage traffic -- if one pathway becomes clogged with traffic, packets can go through a different route. This is different from the traditional phone system, which creates a dedicated circuit through a series of switches. All information through the old analog phone system would pass back and forth between a dedicated connections. If something happened to that connection, the call would end.

That's not the case with traffic across IP networks. If one connection should fail, data can travel across an alternate route. This works for individual networks and the Internet as a whole. For instance, even if a packet doesn't make it to the destination, the machine receiving the data can determine which packet is missing by referencing the other packets. It can send a message to the machine sending the data to send it again, creating redundancy. This all happens in the span of just a few milliseconds.

When everyone had smartphone near the end of 2014-16 due to boom of affordable internet in India people preferred instant messaging via apps such as Whatsapp which was first on the list then came telegram and it was followed by big tech companies like Facebook and twitter which integrated messaging on their platforms.

A chat application has the following components: Messaging application, server and a persistent connection. The messaging application is the part that you see. This is the part of the system that resides on your phone, laptop or personal computer as a small app. There is a text box where you type messages and another text box where all the previous messages in the conversation are shown along with the timelines. There is a button to send the messages and on the desktop / laptop, pressing the Enter / Return key will do the job. There is a trigger to open a set of emoticons that you can use. You can achieve the same using a combination of symbols that look like the same emoticon when looked sideways. Finally, for more sophisticated apps, there will be voice / video calls.

Before being ready for use, the messaging app connects to a central server. It is only because of this connection that you are able to send messages to others who are connected to the same server and others are able to see you online and send messages to you. To establish a connection, the user must first authenticate with his/her credentials.

On the server side of the equation, there is a server-side software that listens for connections from the instant messaging client. The server maintains a map of which connections belongs to which user, so that messages can be reliably relayed to the correct recipient and marked as being sent by a certain sender.

Finally, the third component is a persistent connection. This simply means that the instant messaging app must remain connected to the server ALL the time. A user is seen online and is able to send and receive messages ONLY as long as the connection is held stably. Internet failures, unreliable Internet connection, company firewall rules and Internet provider restrictions will often cause the connection to fail and the instant messaging app either does not work or suffers dropped messages.

Chapter - 3

Functionality & Project design

An important point to note is that an instant messenger can only connect with the server of the same company. You cannot usually facilitate a chat between instant messengers from different companies. A Google Hangout can only chat with another Hangout. A Whatsapp user cannot chat with a Facebook Messenger user. This is because the way Hangout encodes and encrypts its messages on the Internet cannot be understood by Whatsapp or Facebook Messenger. This is true for all the company-specific chat protocols.

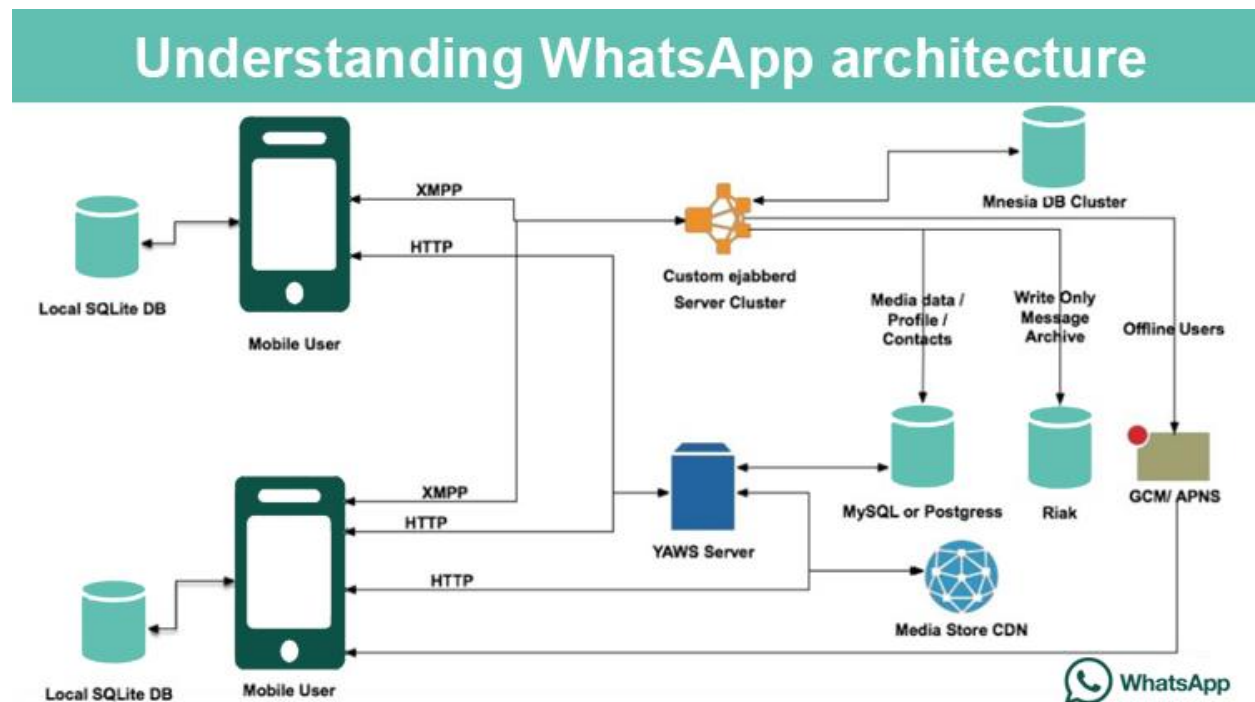


Figure 2.2

Whatsapp

Data collected by app: Phone Number, Email Address, Contacts, Coarse Location, Device ID, User ID, Advertising Data, Purchase History, Product Interaction, Payment Info, Crash Data, Performance Data, Other Diagnostic Data, Customer Support, Product Interaction, Other User, Content, Metadata.

Architecture of WhatsApp Messaging App

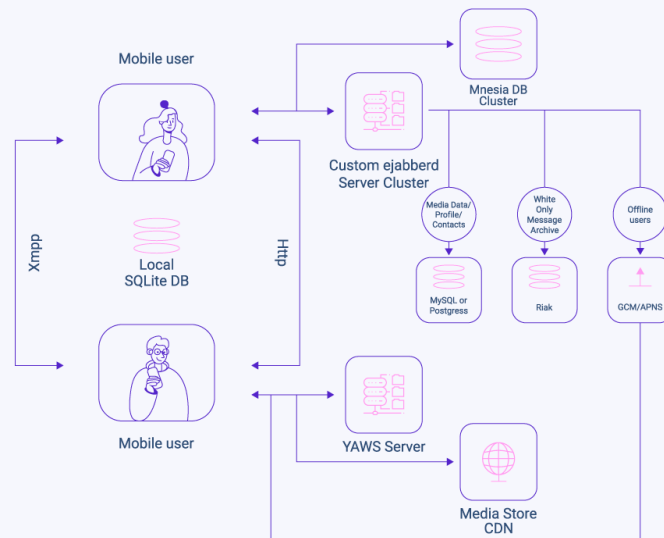


Figure 2.3

Facebook Messenger

Data collected by app: Precise Location, Coarse Location, Physical Address, Email Address, Name, Phone Number, Other User Contact Info, Contacts, Photos or Videos, Gameplay Content, Other User Content, Search History, Browsing History, User ID, Device ID, Third-Party Advertising, Purchase History, Financial Info, Product Interaction, Advertising Data, Other Usage Data, Crash Data, Performance Data, Other Diagnostic Data, Other Data Types, Developer's, Advertising or Marketing, Health, Fitness, Payment Info, Sensitive Info, Product Personalization, Credit Info, Other Financial Info, Emails or Text Messages.

Signal

Data collected by app: None.

You need to sign up on Signal using your mobile phone number for registration, but the app does not link your phone number to your identity. If you check the app on the App Store it states that there is no data linked to you.

Telegram

Data collected by app: Name, Phone Number, Contacts, User ID.

D.A.P.P

Data collected by app: **NONE**

BLOCK CHAIN MESSAGING

Block chain is the other name of decentralization and privacy with regards to personal privacy. Therefore, such revelations have gathered special attention towards initiatives like e-Chat. E-Chat is a block chain-based decentralized messenger and “the fastest growing social network”. It uses peer-to-peer (p2p) protocols based on the block chain, for instant text messaging. For transferring binary files, like photos and documents, e-Chat will employ the Interplanetary File Transfer protocol, which is the full form for IPFS. Even IPFS itself is a block chain technology, so e-Chat seems to exploit the benefits of block chain pretty well. Since this is decentralized messaging, e-Chat can even utilize other benefits of redundancy, like backups. If the chat history on a given device (or node, to be a little more technical) is deleted, it can be reconstructed using the neighboring peers’ chat histories involving that node. However, it should be noted that this benefit isn’t exclusive to decentralized messaging services - even Whatsapp, that is based on more of a client-server architecture (i.e., centralized), can use a similar technique to offer chat reconstruction to its users. But e-Chat has more to offer than just replicate the functionality of popular message services, with a different implementation. Most people don’t care as much about the implementation as they do about the interface.

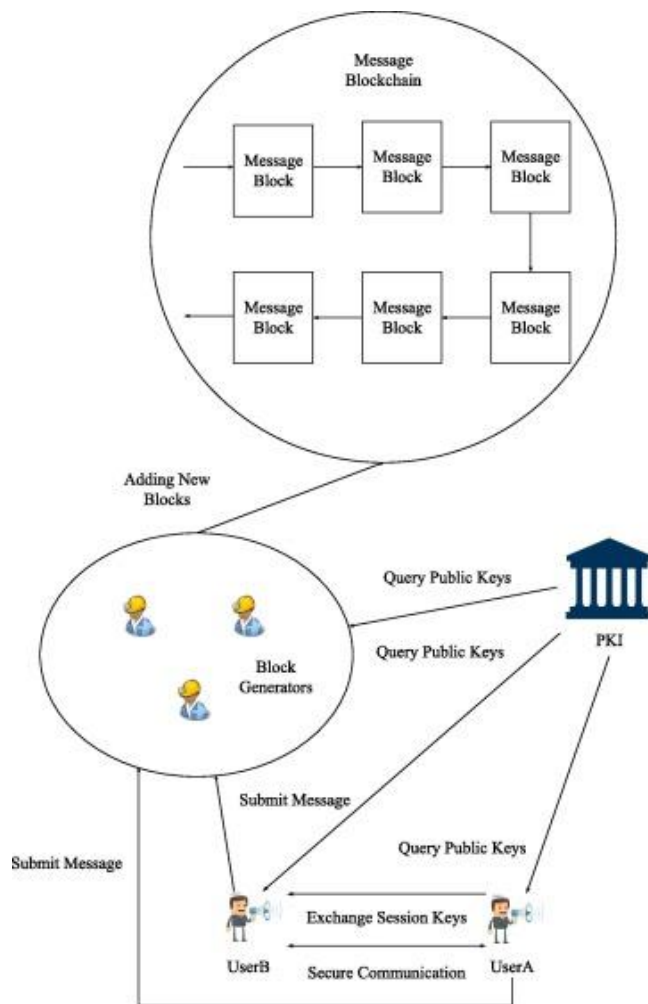


Figure 2.3

Cons –

Energy use: Block chain uses a lot of energy. As one of the largest networks in the world, Block chain uses as much electricity as Nigeria. Though people are working on solving this issue, it is having massive repercussions both economically and environmentally.

Compliance issues. Since block chain relies on dispersing information across a global network, international regulations could affect the information shared across borders. So, for instance, though AWS centralizes its offering, the decentralized nature of block chain could lead to some compliance issues concerning data residency as it is hosted in other countries.

Scalability issues. Decentralization can create an environment where if usage drastically increases suddenly in on application, another's processes could be

impeded. Users must understand the nuances and select the best method to work with their needs and capabilities.

In our project we have not used a database rather used gun.js module which will be discussed further.

Languages used –

Java script

Svelte

CSS

HTML

Other Technical requirements-

Nodejs

GitBash

Git

Gun

Nprok(For Clients side hosting only)

VScode

Gun

We have installed gun from the following command prerequisite (nodejs)

-npm install gun.

We have imported gun for security, encryption and authorization the gun module used is “**gun/sea**”

This gun module will enable user authentication and this uses End-to-End encryption by generating keys for both username and password for every new user.

Next gun module used-

“gun/axe”

This module helps to connect peers together which is the basic requirement for any social media or messaging app.

Then we initiated the data base by giving variable.

We enable “**sessionStorage**” so that user remains logged in during sessions.

```
JS user.js  X
gun-chat > src > JS user.js > ...
1  import GUN from 'gun';
2  import 'gun/sea';
3  import 'gun/axe';
4  import { writable } from 'svelte/store';
5
6  // Database
7  export const db = GUN();
8
9  // Gun User
10 export const user = db.user().recall({sessionStorage: true});
11
12 // Current User's username
13 export const username = writable('');
14
15 user.get('alias').on(v => username.set(v))
16
17 db.on('auth', async(event) => {
18   const alias = await user.get('alias'); // username string
19   username.set(alias);
20
21   console.log(`signed in as ${alias}`);
22 });
```

Figure – 3.1

Svelte

Svelte is a radical new approach to building user interfaces. Whereas traditional frameworks like React and Vue do the bulk of their work in the browser, Svelte shifts that work into a compile step that happens when you build your app.

Instead of using techniques like virtual DOM diffing, Svelte writes code that surgically updates the DOM when the state of your app changes.

We have used Svelte for our web app interface as it is more efficient and saves a lot of time.

With Svelte we have initially created function such as-

-Sign Out

-Setting Username

For the header module of the app

```
<script>
  import { username, user } from './user';

  function signout() {
    user.leave();
    username.set('');
  }
</script>

<header>
<h1>🗨️</h1>
  {#if $username}
    <div class="user-bio">
      <span>Hello <strong>{$username}</strong></span>
      <img src={`https://avatars.dicebear.com/api/initials/${$username}.svg`} alt="avatar" />
    </div>

    <button class="signout-button" on:click={signout}>Sign Out</button>

  {:else}
    <h3>D.I.M</h3>
  {/if}
</header>
```

Figure – 3.2

```
App.svelte X
gun-chat > src > App.svelte > script
1 <script>
2   import Chat from './Chat.svelte';
3   import Header from './Header.svelte';
4 </script>
5
6 <div class="app">
7   <Header />
8   <Chat />
9 </div>
```

Figure – 3.3

In the Login page – we added

-Username

-Password

Setting Username max length to 16 and min to 3

Setting password text type to password for privacy.

And added Login and Signup function with binding it with other two function with if statement.

```
Login.svelte X
gun-chat > src > Login.svelte > script
1 <script>
2   import { user } from './user';
3
4   let username;
5   let password;
6
7   function login() {
8     user.auth(username, password, ({ err }) => err && alert(err));
9   }
10
11  function signup() {
12    user.create(username, password, ({ err }) => {
13      if (err) {
14        alert(err);
15      } else {
16        login();
17      }
18    });
19  }
20 </script>
21
22 <label for="username">Username</label>
23 <input name="username" bind:value={username} minlength="3" maxlength="16" />
24
25 <label for="password">Password</label>
26 <input name="password" bind:value={password} type="password" />
```

Figure – 3.4

Key Uniqueness out of scope due to low traffic and early stage of implementation
Thus this feature is still not required but would be added in the future.

Chat Interface Module

Using the previously initiated database module for storing chat data
- Just created a single giant chat room for experiment and added the features below.

Used

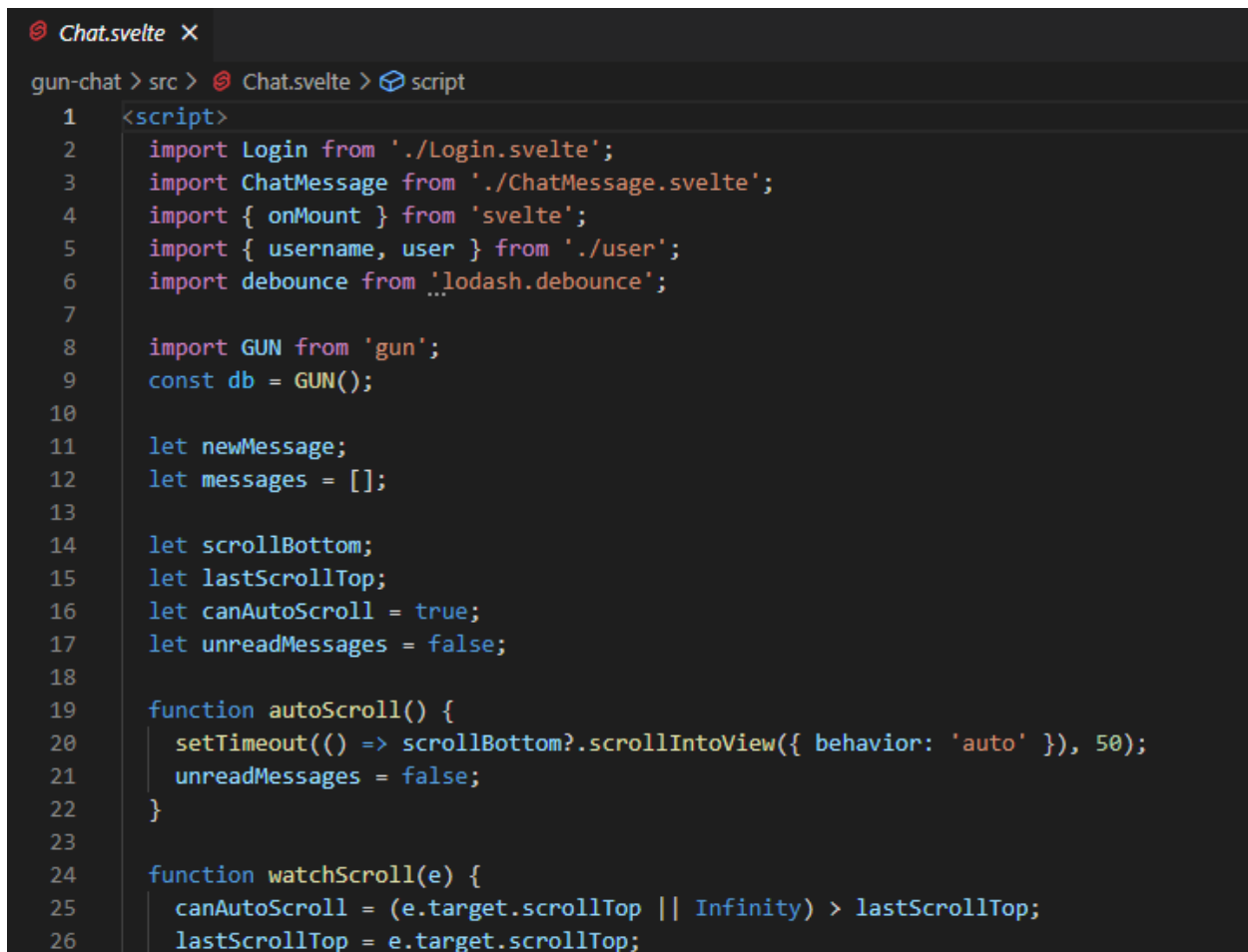
Who- For the alias that sent the message

What –For the message sent

When- For message timestamp

When all the three conditions are met in a single message the message is stored in an array and used “Each” in Svelte to loop over the array this will render the chat messages after every message sent and used “.when” key in Svelte to sort all the messages efficiently.

Added submit for the send message function.



```
Chat.svelte X
gun-chat > src > Chat.svelte > script
1 <script>
2   import Login from './Login.svelte';
3   import ChatMessage from './ChatMessage.svelte';
4   import { onMount } from 'svelte';
5   import { username, user } from './user';
6   import debounce from 'lodash.debounce';
7
8   import GUN from 'gun';
9   const db = GUN();
10
11   let newMessage;
12   let messages = [];
13
14   let scrollTop;
15   let lastScrollTop;
16   let canAutoScroll = true;
17   let unreadMessages = false;
18
19   function autoScroll() {
20     setTimeout(() => scrollTop?.scrollIntoView({ behavior: 'auto' }), 50);
21     unreadMessages = false;
22   }
23
24   function watchScroll(e) {
25     canAutoScroll = (e.target.scrollTop || Infinity) > lastScrollTop;
26     lastScrollTop = e.target.scrollTop;
```

Figure – 3.5

```
Chat.svelte x
gun-chat > src > Chat.svelte > script
29   $: debouncedWatchScroll = debounce(watchScroll, 1000);
30
31   onMount(() => {
32     var match = {
33       // lexical queries are kind of like a limited RegEx or Glob.
34       '.': {
35         // property selector
36         '>': new Date(+new Date() - 1 * 1000 * 60 * 60 * 3).toISOString(), // find any indexed property larger ~3 hours ago
37       },
38       '-': 1, // filter in reverse
39     };
40
41     // Get Messages
42     db.get('chat')
43       .map(match)
44       .once(async (data, id) => {
45         if (data) {
46           // Key for end-to-end encryption
47           const key = '#foo';
48
49           var message = {
50             // transform the data
51             who: await db.user(data).get('alias'), // a user might lie who they are! So let the user system detect whose data it is.
52             what: (await SEA.decrypt(data.what, key)) + '', // force decrypt as text.
53             when: GUN.state.is(data, 'what'), // get the internal timestamp for the what property.
54           };

```

Figure – 3.6

Used Html and CSS for page structure and design-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset='utf-8'>
  <meta name='viewport' content='width=device-width,initial-scale=1'>

  <title>Gun Chat</title>

  <link rel='icon' type='image/png' href='/favicon.png'>
  <link rel='stylesheet' href='/global.css'>
  <link rel='stylesheet' href='/build/bundle.css'>

  <script defer src='/build/bundle.js'></script>
</head>

<body>
</body>
</html>
```

Figure – 3.7


```
# global.css X
gun-chat > public > # global.css > html
1  html, body {
2      position: relative;
3      width: 100%;
4      height: 100%;
5  }
6
7  body {
8      color: #333;
9      margin: 0;
10     box-sizing: border-box;
11     font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen-Sans, Ubuntu, Cantar
12     overflow: hidden;
13 }
14
15 a {
16     color: rgb(0,100,200);
17     text-decoration: none;
18 }
19
20 a:hover {
21     text-decoration: underline;
22 }
23
24 a:visited {
25     color: rgb(0,80,160);
26 }
```

Figure – 3.8

```
# global.css X
gun-chat > public > # global.css > html
28 label {
29     display: block;
30 }
31
32 input, button, select, textarea {
33     font-family: inherit;
34     font-size: inherit;
35     -webkit-padding: 0.4em 0;
36     padding: 0.4em;
37     margin: 0 0 0.5em 0;
38     box-sizing: border-box;
39     border: 1px solid #ccc;
40     border-radius: 2px;
41 }
42
43 input:disabled {
44     color: #ccc;
45 }
46
47
48 body {
49     background-color: #282c34;
50 }
51
52 .app {
53     text-align: center;
```

Figure – 3.9

Other Modules

The other packages which are inbuilt while creating the file are already created

```
launch.json U x
gun-chat > .vscode > {} launch.json > Launch Targets > {} Launch Chrome against localhost
1  {
2    // Use IntelliSense to learn about possible attributes.
3    // Hover to view descriptions of existing attributes.
4    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5    "version": "0.2.0",
6    "configurations": [
7      {
8        "type": "pwa-chrome",
9        "request": "launch",
10       "name": "Launch Chrome against localhost",
11       "url": "http://localhost:5000",
12       "webRoot": "${workspaceFolder}"
13     }
14   ]
15 }
```

Figure – 3.10

Installed ngrok – for hosting which is required when used for client side or testing.

```
C:\Users\SYSTEM H521\Downloads\ngrok-stable-windows-amd64\ngrok.exe - ngrok http 5000 -host-header="localhost:5000"
ngrok by @inconshreveable
Session Status      online
Account             kunal.panchal271@gmail.com (Plan: Free)
Version             2.3.40
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://5625-117-97-231-113.ngrok.io -> http://localhost:5000
Forwarding           https://5625-117-97-231-113.ngrok.io -> http://localhost:5000
Connections
  ttl    opn    rt1    rt5    p50    p90
   0     0     0.00  0.00  0.00  0.00
```

Figure – 3.11

CHAPTER-4

Result and Discussion

The Output and Working of our program

Login Interface

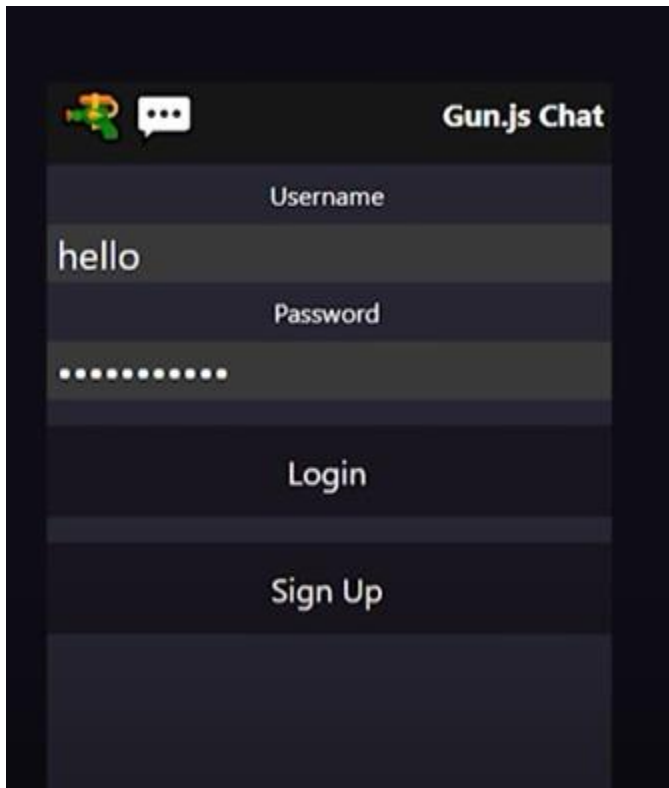


Figure – 4.1

Chat Interface

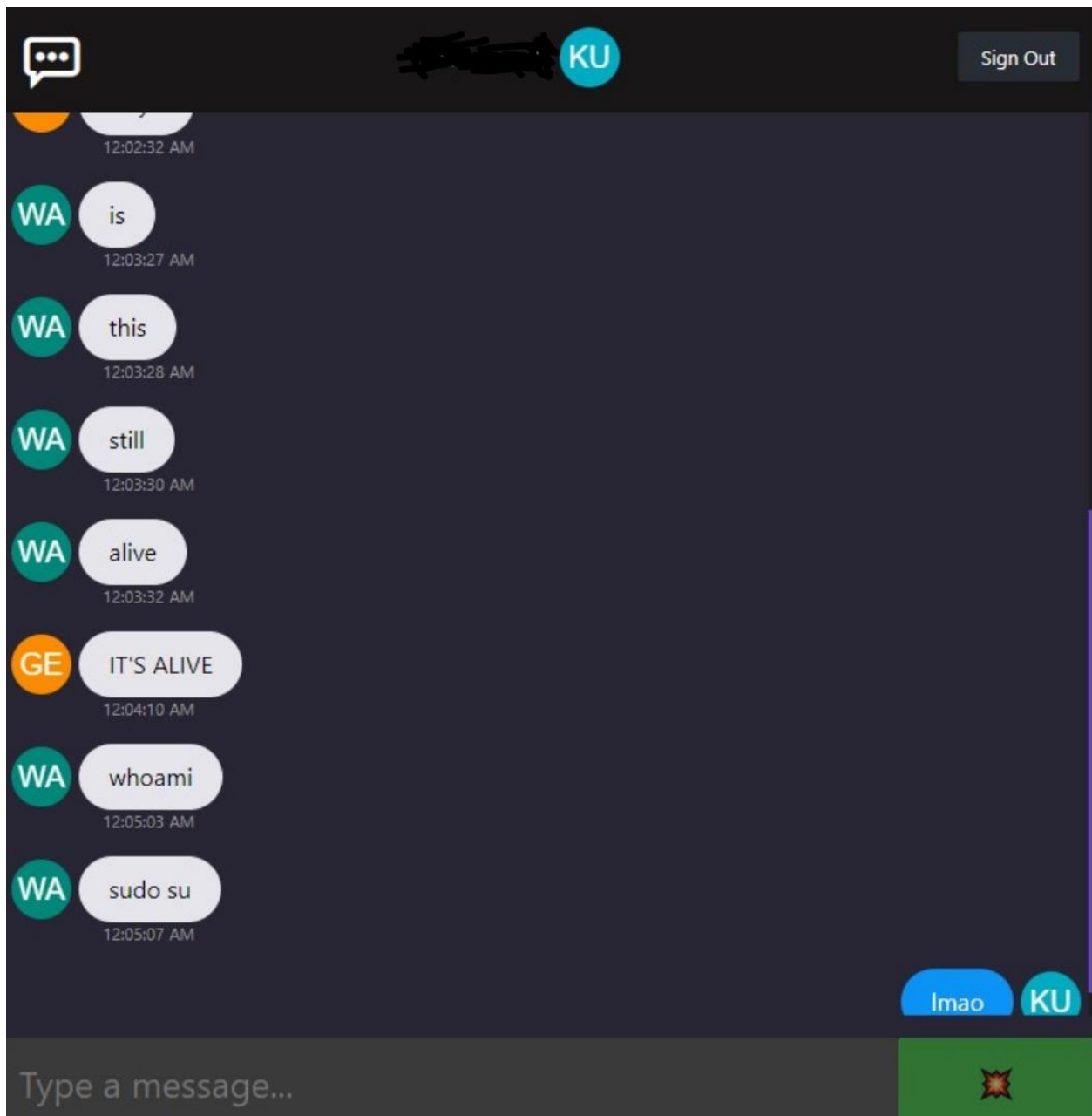


Figure – 4.2

With Designated
Sign out Button
Send Message Button

Username Depicted on top
Sending message function –

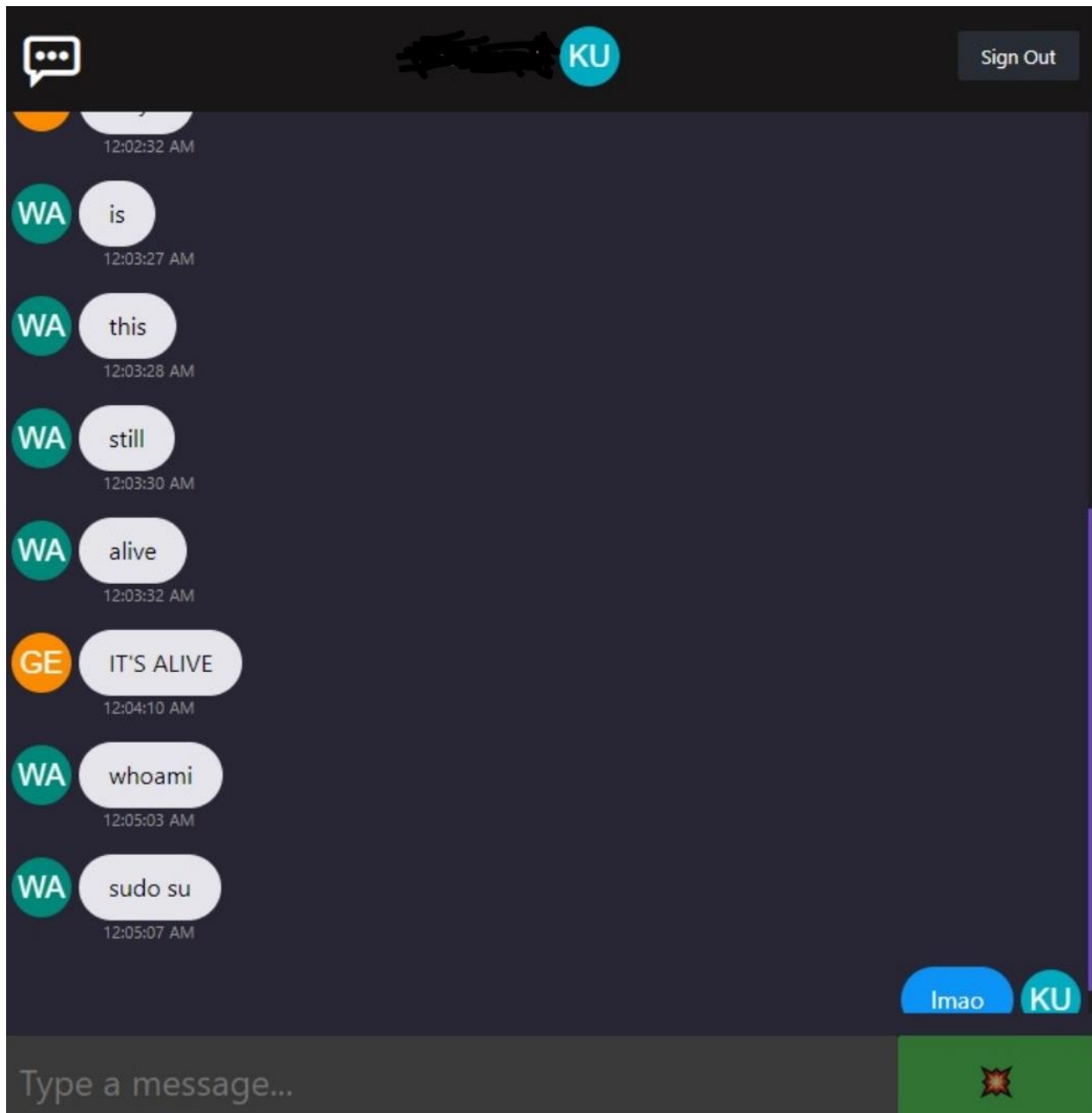


Figure – 4.3

With Sorted Messages
Scroll Option to view unread messages.

CHAPTER - 5

Conclusion & Future Scope

We have successfully implemented our project with even local hosting and hosting to network and our web app is working perfectly with all the operating systems that is IOS/ANDROID/WINDOWS with any web browser.

Our future scope for this project is as follows-

Features to be added-

-Personal messaging

-Sharing of media

-Web app to app for increasing cache memory for more than 5mb

-Faster messaging.

Other Long term Goals for the Project-

Creating Social media platform from messaging and adding features like any generic social media platform with key focus to data privacy and decentralization.

Solving existing problems like limited cache storage which can easily be solved but would not be considered decentralized. Adding unique username and password for security and adding better encryption than end to end encryption as our app does not have a database but rather node connected so if any developer finds the node the data for the user is easily accessible thus to prevent that better encryption which is not time costly and is efficient enough to sustain the standards of instant messaging is required.

CHAPTER – 6

References

Gunjs docs

<https://gun.eco/docs/API>

Stackoverflow

<https://stackoverflow.com/>

Literature Research

<https://conbio.onlinelibrary.wiley.com/doi/full/10.1111/cobi.13708>

<https://sopa.tulane.edu/blog/decentralized-social-networks>

<https://www.floridatechonline.com/blog/information-technology/blockchain-as-a-service-pros-and-cons/>

svelte dev

<https://svelte.dev/>

Youtube

https://www.youtube.com/watch?v=J5x3OMXjgMc&t=34s&ab_channel=Fireship