VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Internet and Web Programming

(CSE3002)

SCYTHE CHAT

(REAL-TIME GROUP CHAT APPLICATION)

J Component Project

Team Details:

Name	Reg. No
Vibhu Kumar Singh	19BCE0215
Mihir Gupta	19BCE0619

Submitted to – Prof. N. Nalini

ABSTRACT

With the fast development of the Internet, more and more people choose network chatting tools for communication. Traditional real-time chatting software is usually desktop application programmes that run in C/S mode and require particular client programmes to run. The browser-based real-time chatting solution does not require any additional client software, and visual communication may be easily accomplished using the browser. Text communication is accomplished using server data forwarding, while voice and video chat data is transmitted via a point-to-point connection between browsers. Here, in our project we have built a real-time chat application (web version) similar to that of Whatsapp and other chat-applications.

INTRODUCTION

In this busy modern world where people live far from each other due to work and their busy life, communication gap begun to increase. People started to swerve away from each other and started to talk less and less. In their preoccupied life, calling on phone also seemed to be difficult. That’s when chatting comes into picture. With the help of chatting applications like facebook and messenger, people could stay connected with other people much easily. In our project we have also developed an online group chatting application called scythe chat which will help you text your close ones easily and safely. We have developed Scythechat using ReactJS as the frontend framework and Google cloud firestore as the backend database. We also provided end to end encryption to your messaged using cryptoJS which only stores encrypted messages in the database and only decrypts them at the clients machine. ReactJS is an open-source, part based front end library which is dependable just for the view layer of the application. It was at first evolved and kept up with by Facebook and later utilized in its projects like WhatsApp and Instagram. Cloud Firestore is an adaptable just as versatile NoSQL cloud information base. It is utilized to store and adjust information for customer and server-side turn of events. It is utilized for versatile, web, and server improvement from Google Cloud Platform and Firebase. Like the Firebase Real-time Database, it continues to adjust our information by means of constant audience members to the customer application. It gives disconnected help to portable and web so we can make responsive

applications that work paying little mind to arrange dormancy or Internet network.

TECHNOLOGIES

The tech-stack used in our project is as follows:

Front-End:

- **ReactJS**
- **HTML**
- **CSS**
- **Material UI Components**
- **Material UI Icons**
- **JavaScript**

ReactJS is an explanatory, productive, and adaptable JavaScript library for building reusable UI parts. It is an open-source, part based front end library which is dependable just for the view layer of the application. It was at first evolved and kept up with by Facebook and later utilized in its projects like WhatsApp and Instagram.

A ReactJS application is comprised of numerous parts, every part liable for yielding a little, reusable piece of HTML code. The parts are the core of all React applications. These Components can be nested with different parts to permit complex applications to be worked of straightforward structure blocks. ReactJS utilizes virtual DOM based system to fill information in HTML DOM. The virtual DOM works quick as it just changes individual DOM components as opposed to reloading total DOM every reload.

To make React application, we compose React parts that relate to different components. We put together these parts inside more significant level parts which characterize the application structure. For instance, we take a structure that comprises of numerous components like info fields, marks, or fastens. We can compose every component of the frame as React parts, and afterward we join it into a more significant level part, i.e.,

the structure part itself. The structure parts would determine the design of the structure alongside components within it.

The explanation we have picked ReactJS over some other system is a direct result of the accompanying:

- Speed

The React fundamentally permits engineers to use individual pieces of their application on both customer side and the server-side, which at last lifts the speed of the advancement interaction.

In basic terms, various engineers can compose individual parts and all progressions made won't cause the rationale of the application.

- Adaptability

Contrasted with other frontend systems, the React code is simpler to keep up with and is adaptable because of its measured construction. This adaptability, thus, saves gigantic measure of time and cost to organizations.

- Execution

Respond JS was intended to give elite execution at the top of the priority list. The center of the system offers a virtual DOM program and server-side delivering, which makes complex applications run amazingly quick.

- Ease of use

Sending React is genuinely simple to achieve assuming that you have some fundamental information on JavaScript.

Truth be told, a specialist JavaScript designer can without much of a stretch gain proficiency with all intricate details of the React structure in an issue of a little while.

- Mobile Development

Assuming you thought React is for web advancement no one but, you were unable to be all the more off-base! Facebook has as of now

overhauled the structure for creating portable local applications for both Android and iOS stages.

It is apparent that ReactJS ended up being the best frontend system for a task like genuine talk application where speed and execution is critical.

CryptoJS for End-to-End Encryption:

CryptoJS is a developing assortment of standard and secure cryptographic calculations executed in JavaScript utilizing best practices and examples. They are quick, and they have a steady and basic interface.

In the event that you in all likelihood dislike CryptoJS, to examine new elements, or on the other hand to add to the venture, you can visit the CryptoJS conversation bunch

Material UI:

Material-UI is basically a library that permits us to import and utilize various parts to make a UI in our React applications. This saves a lot of time since the designers don't have to compose everything without any preparation.

Material-UI gadgets are intensely propelled by Google's standards on building UIs. It is, in this way, simple for engineers to assemble outwardly engaging applications.

Back-End:

- **Firebase Firestore**
- **Firebase Authentication**
- **Firebase Hosting and Deployment**

We chose Firebase which is a Backend as a Service (BaaS) provided by Google for the following reasons:

- It supports a real-time update
- One developer can write Front-end and Back-end
- It’s cheaper than writing a custom Back-end
- It’s faster than writing a custom Back-end

Firebase is a development platform known initially for its realtime information base that is currently at its center a multi-hub, key-esteem data set advanced for synchronizing information, regularly between client machines or cell phones and incorporated stockpiling in the cloud. It's intended to make life more straightforward for designers by taking care of a large part of the moving back and forth of information. That frees application engineers from the programming loads related with overseeing adaptations or areas. They can compose the new pieces to Firebase and the information will be steady all through the framework.

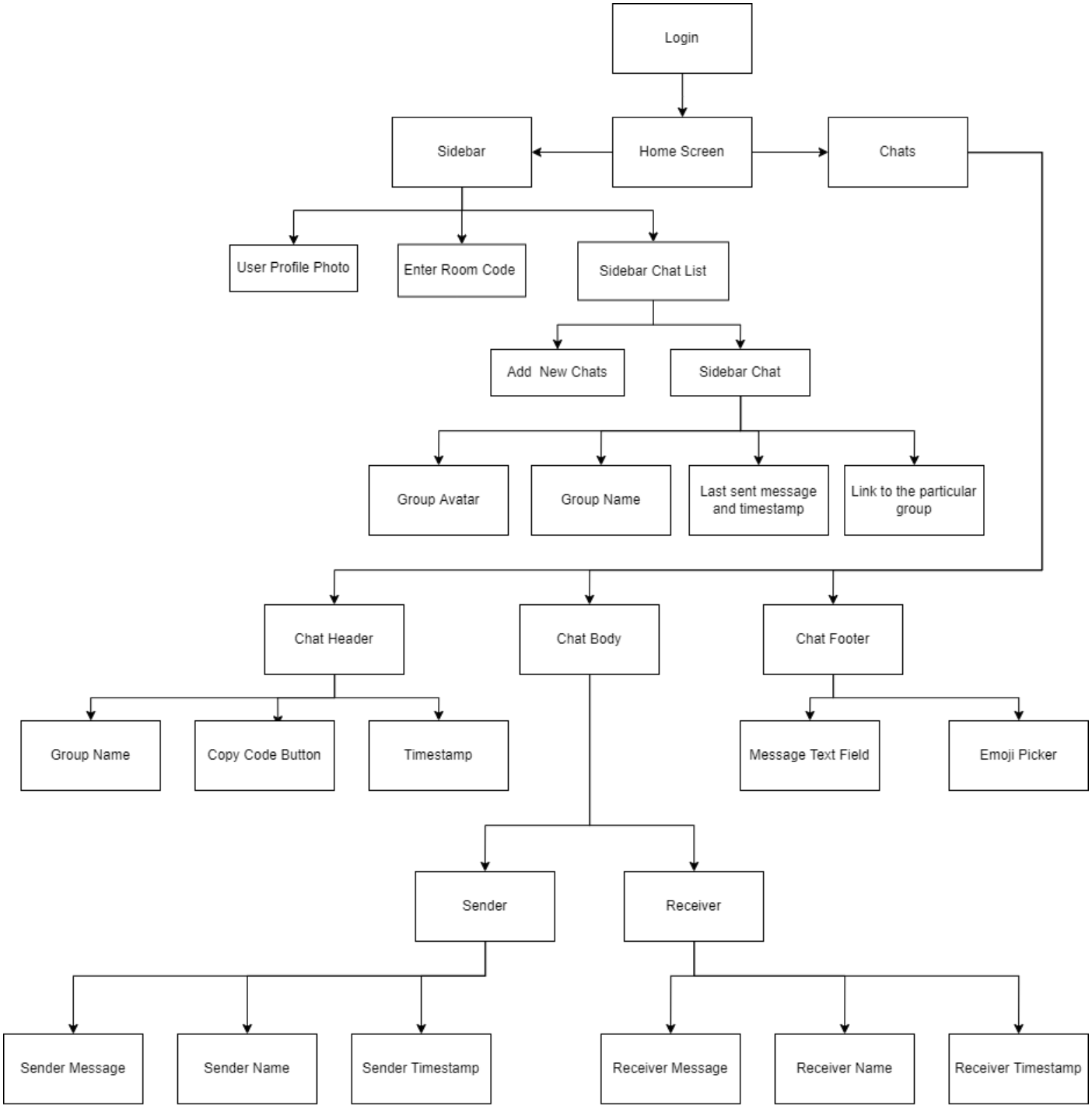
Firebase is esteemed generally in light of the fact that it can continually spread and synchronize changes between neighborhood duplicates of data put away on clients' machines with renditions kept in the cloud. Firebase dispenses with large numbers of the difficulties of blending confirmation, synchronization, and isolation by shuffling different forms and guaranteeing the right pieces are something very similar all through the framework.

Today, Firebase is a focal piece of the Google cloud improvement toolbox. The item, a climax of long periods of development, was situated at the focal point of a versatile backend-as-a-administration presenting from the organization known as Firebase, which Google obtained in 2014. Firebase is accessible through Google. In the meantime, open-source libraries and instruments are accessible that interface with Firebase.

Since real-time updation of data was one of our key requirements, Firebase fits right into the fold. On top of that, it provides an authentication module as well which allows users to log in using various authentication services such as Google, Facebook, Github, LinkedIn, etc.

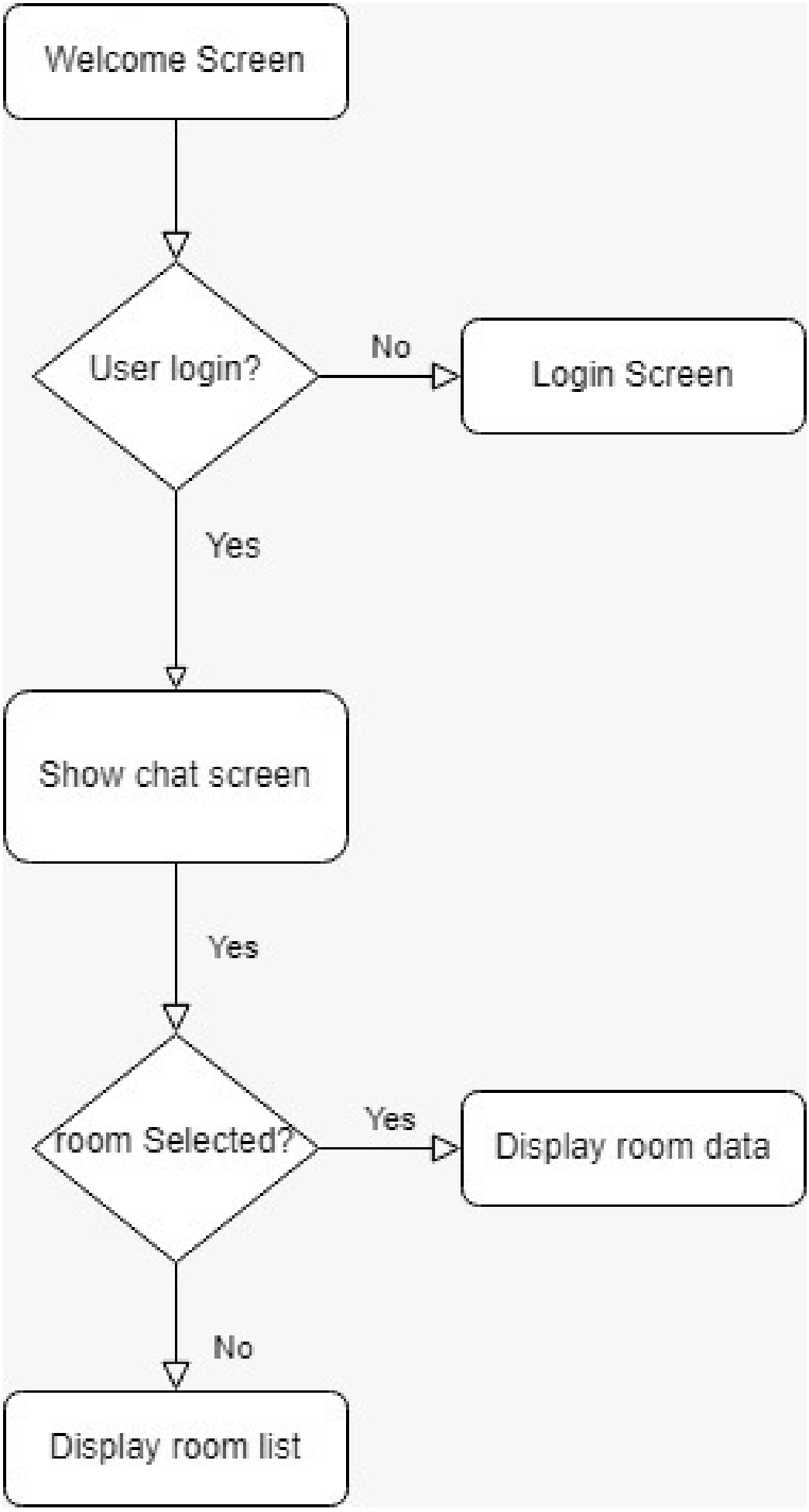
Firebase was also used for the hosting and deployment of this project. It is worth noting that Firebase is a powerhouse of the backend we are using in our project.

ARCHITECTURE



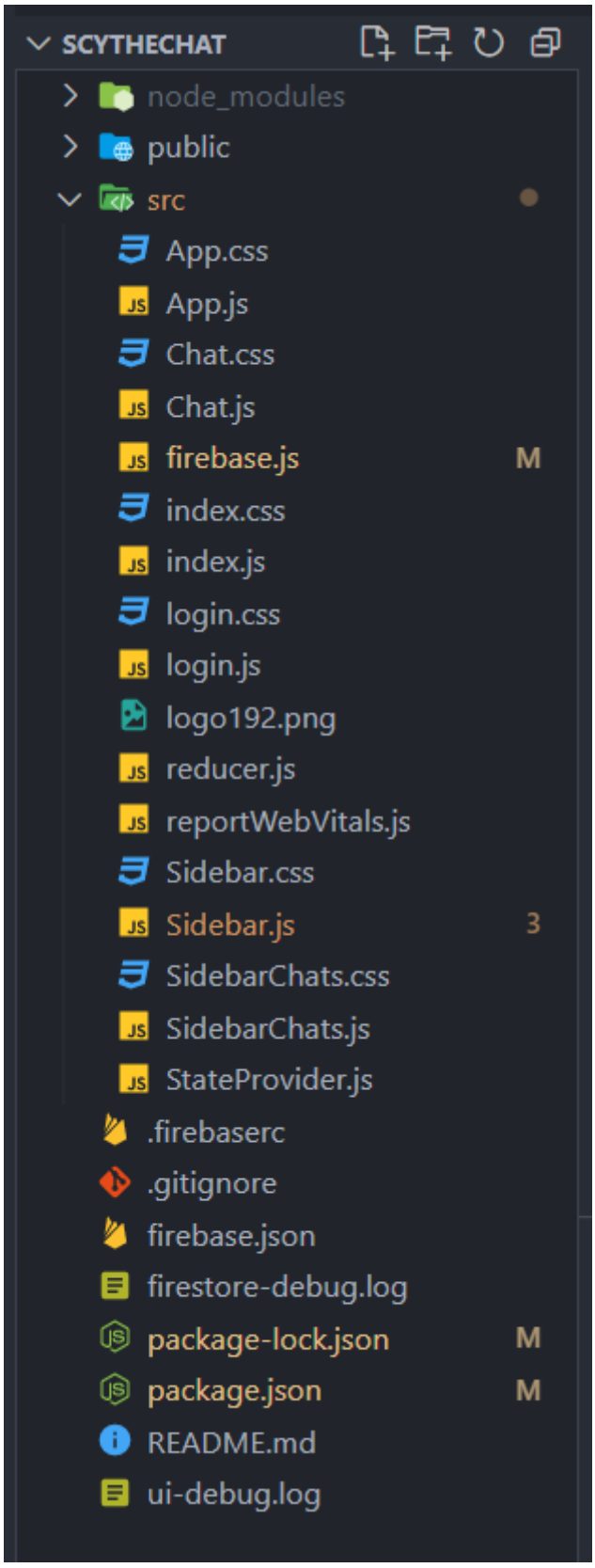
Architecture Diagram

FLOW OF THE PROJECT



IMPLEMENTATION

Folder Structure:



Index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-
scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is
installed on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder
during the build.
      Only files inside the `public` folder can be referenced from
the HTML.

      Unlike "/favicon.ico" or "favicon.ico",
"%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root
public URL.
      Learn how to configure a non-root public URL by running `npm
run build`.
    -->
    <title>ScytheChat</title>
  </head>
  <body>

    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty
page.

      You can add webfonts, meta tags, or analytics to this file.
```

```
      The build step will place the bundled scripts into the <body>
tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn
build`.
    -->
  </body>
</html>
```

Index.js (entry point of the project):

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import reducer,{initialState} from './reducer'
import {StateProvider }from './StateProvider'

ReactDOM.render(
  <React.StrictMode>
    <StateProvider initialState = {initialState} reducer =
{reducer}>
    <App />
    </StateProvider>
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a
function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more:
https://bit.ly/CRA-vitals
reportWebVitals();
```

App.js:

```
import './App.css';
import Sidebar from './Sidebar';
import Chat from './Chat';
import Login from './login'
//import ReactDOM from "react-dom";
import { BrowserRouter as Router, Switch, Route } from "react-
router-dom";
import React,{useState} from 'react';
import { useStateValue } from './StateProvider';

function App() {
  // const [user, setuser] = useState(null);
  const [{user}, dispatch] = useStateValue();
  return (
    <div className="app">

      <div className="app__body">
        {!user ?
          (
            <Login/>
          ) : (
            <Router>
              <Sidebar />
              <Switch>
                <Route path="/rooms/:roomId">
                  <Chat />
                </Route>
              </Switch>
            </Router>
          )}
      </div>
    </div>
  );
}

export default App;
```

index.css:

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI',
'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica
Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier
New',
    monospace;
}
```

App.css:

```
.app
{
  background-color: rgb(0, 0, 0);
  height: 100vh;
  color: white;
  display: grid;
  place-items: center;
}

.app__body
{

  display: flex;
  margin-top: -50px;
  background-color: #131c21;;
  box-shadow: -1px 5px 60px -18px rgb(67, 255, 224);
  height: 90vh;
  width: 90vw;
}
```

Login.js:

```
import React from 'react'
import { auth, provider } from './firebase'
import "./login.css"
import { actionTypes } from './reducer';
import { useStateValue } from './StateProvider';
function Login() {
    const [{},dispatch] = useStateValue();

    const signIn = ()=>{
        auth.signInWithPopup(provider)
        .then((result)=>{
            dispatch({
                type: actionTypes.SET_USER,
                user: result.user,
            })
        })
        .catch((e)=>alert(e.message));
    };

    return (

        <div className = "login">
            <div className="login__image">
                <img
src="https://upload.wikimedia.org/wikipedia/commons/thumb/a/a7/React-icon.svg/1280px-React-icon.svg.png"  alt="img" />
            </div>
            <div className="login__text">
            Welcome to ScytheChat
            </div>
            <div className="login__google">
                <button onClick={signIn}>
                    Login with Google
                </button>
            </div>

        </div>
    )
}

export default Login;
```

Login.css:

```css
.login
{
display: flex;
flex:1;
flex-direction: column;
    /* align-items: center; */
    place-items: center;
    /* place-self: center; */
    margin-top: 10vh;
}
.login__image > img
{
    height: 30vh;
}

.login__text
{
    font-size:  5vw;
    margin-bottom: 5vh;
}
.login__google
{
    border: thin;
    border-style: solid;
    border-color: rgb(67, 255, 255);
    border-radius: 20px;
    width: 20vw;
    display: flex;
    height: 5vh;
    flex-direction: column;
    place-items: center;
    place-content: center;


}

.login__google > button
{
    color:aqua;
    background-color: rgb(0, 0, 0, .0);
    font-size: medium;
    padding: 0px;
    border: none;
    place-content: center;
    width: 100%;
```

```
    height: 100%;
}
.login__google  :hover
{
    background-color: rgba(67, 255, 255, .09);
    border-radius: 30px;
    cursor: pointer;
}
```

Reducer.js:

```
export const initialState = { user:null,};

export const actionTypes = {
    SET_USER: "SET_USER",
};

const reducer = (state,action)=>{
    console.log(action);
    switch(action.type){
        case actionTypes.SET_USER:
            return{
                ...state,
                user:action.user,
            };
        default:
            return state;
    }
};
export default reducer;
```

Sidebar.js:

```
import React, { useEffect, useState } from 'react'
import ChatBubble from '@material-ui/icons/Chat'
import SearchOutLined from '@material-ui/icons/AddRounded'
import ContentCopyIcon from '@material-ui/icons/FileCopyOutlined'
import MoreVert from '@material-ui/icons/MoreVert'
import { Avatar, IconButton } from '@material-ui/core'
import './Sidebar.css'
import SidebarChats from './SidebarChats'
import firebase from 'firebase';
import db from "./firebase.js";
import { useStateValue } from './StateProvider'


function Sidebar() {
    const [rooms, setRooms] = useState([]);
    const [input, setInput] = useState("");
    const [{ user }, dispatch] = useStateValue();
    console.log(user);
    useEffect(() => {
        const unsubscribe =
db.collection('Rooms').onSnapshot(snapshot => (

            setRooms(snapshot.docs.map(doc => ({
                id: doc.id,
                data: doc.data(),


            })))
        ))
        return () => {
            unsubscribe();
        };
    }, [])
    const addMember = (e) => {
        e.preventDefault();
        console.log(input);

        try {
            const dd = db.collection('Rooms').doc(input);
            const dd2 = dd.get()
            .then((something)=>
            {
                if(something.exists)
                {

            dd.update({
```

```
                members:
firebase.firestore.FieldValue.arrayUnion(user.uid)
            }).then(() => alert("Room joined successfully"))
                }
                else
                {
                    alert("Invalid Room Code");
                }


        })
         }
        catch (ss) {
            alert("Invalid Room Code");

        }
    }
        return (
            <div className="sidebar">
                <div className="sidebar__header">
                    <IconButton><Avatar src={user.photoURL}
/></IconButton>
                    <div className="sidebar__headerRight">
                        <IconButton><ChatBubble style={{ color:
'rgb(67, 255, 224)' }} /></IconButton>
                        <IconButton><MoreVert style={{ color:
'rgb(67, 255, 224)' }} /></IconButton>
                    </div>
                </div>
                <div className="sidebar__search">
                    <div className="sidebar__searchContainer">
                        {/* <SearchOutLined /> */}
                        <form>
                            <input placeholder="Enter chat code to
enter an existing room" type="Text" value={input} onChange={(e) =>
setInput(e.target.value)} />
                            <button onClick={addMember}
type="submit"><SearchOutLined /></button>
                        </form>
                    </div>
                </div>
                <div className="sidebar__chats">
                    <SidebarChats addNewChat />
                    {rooms.map(room => (
                        <SidebarChats key={room.id} id={room.id}
name={room.data.name} members={room.data.members} />
```

```
                    ))}

                </div>
            </div>
        )
    }

    export default Sidebar
```

Sidebar.css:

```
.sidebar{
    flex: 0.35;
    display: flex;
    flex-direction: column;
}


.sidebar__header{
    display: flex;
    justify-content: space-between;
    padding: 10px;
    padding-bottom: 0%;

}
.sidebar__headerRight
{
    display: flex;
    align-items: center;
    justify-content: space-between;
    min-width: 10vh;
}
.sidebar__headerRight > .MuiSVGIcon-root
{
    margin-right: 2vh;
    font-size: 24px !important;
}

.sidebar__search
{
    display: flex;
    align-items: center;
    height: 39px;
    padding: 5px;
    padding-top: 2px;
}
```

```css
.sidebar__searchContainer
{
    display: flex;
    align-items: center;
    background-color:rgba(0, 0, 0, 0.0);
    border: thin;
    border-style: solid;
    width: 100%;
    height: 35px;
    border-radius: 20px;

}
.sidebar__searchContainer > form
{
    display: flex;
    flex: 1;
    align-items: center;
    background-color:rgba(0, 0, 0,0);
    border: none;
    width: 90%;
    height: 30px;
    outline: none;
    color: white;
}
.sidebar__searchContainer > form > input
{
    display: flex;
    flex: 1;
    align-items: center;
    background-color:rgba(0, 0, 0,0);
    border: none;
    width: 90%;
    height: 30px;
    outline: none;
    color: white;
    padding-left: 10px;

}
.sidebar__searchContainer > form >button
{

      /* display: none; */
      background-color: rgba(0, 0, 0, 0.0);
      border: none;
      padding: 10px;
    color: gray;
```

```
}
.sidebar__searchContainer > .MuiSvgIcon-root
{
    padding: 10px;
    color: gray;
}
.sidebar__chats{
    overflow: auto;
    background-color: #131c21;;
    flex: 1;
    /* border-top: 1px solid gray; */
}
```

SidebarChat.js:

```
import { Avatar } from "@material-ui/core"
import './SidebarChats.css'
import db from './firebase'
import { Link } from 'react-router-dom'
import { useStateValue } from './StateProvider'
const Cryptr = require('cryptr');
const cryptr = new Cryptr('ScytheChat');
var CryptoJS = require("crypto-js");
function SidebarChats({ id, name, addNewChat, members }) {
    const [seed, setSeed] = useState('')
    const [messages, setMessages] = useState('')
    const [{ user }, dispatch] = useStateValue();
    useEffect(() => {
        if (id) {
            db.collection('Rooms').doc(id).collection('messages').o
rderBy('timestamp', 'desc').onSnapshot(snap =>
setMessages(snap.docs.map(doc => doc.data())))
        }
    }, [id]);
    useEffect(() => {
        setSeed(Math.floor(Math.random() * 5000))
    }, [])

    const createChat = () => {

        const roomName = prompt("Please Enter Room Name");
        if (roomName) {
            db.collection("Rooms").add(
                {
                    name: roomName,
```

```
                    members: [user.uid],
                }
            );
        }
    }

    return !addNewChat ? (
        members.includes(user.uid) ? (
            <Link to={`/rooms/${id}`}>

                <div className='sidebarChats'>
                    <Avatar
src={`https://avatars.dicebear.com/api/bottts/${seed}.svg`} />
                    <div className="sidebarChats__info">
                        <h2>{name}</h2>
                        <p>{(messages[0]?.message)?(CryptoJS.AES.de
crypt(messages[0].message,"ScytheChat").toString(CryptoJS.enc.Utf8)
):" "}</p>
                        {/* <p>{messages[0]?.message}</p> */}
                    </div>

                </div>
            </Link>):
            <></>
    )
        :
        (
            <div onClick={createChat} className="sidebarChats">
                <h2>Add New Chat</h2>
            </div>
        );

}

export default SidebarChats
```

SidebarChats.css:

```css
.sidebarChats{
    display: flex;

    padding: 20px;
    cursor: pointer;
    border-bottom: 1px solid gray;
    color: rgb(67, 255, 255);
    place-items: center;
}


.sidebarChats:hover{
    background-color: rgb(17, 99, 85);
    color: black;
}
.sidebarChats__info > h2
{
    font-size: 18px;


}
.sidebarChats__info{
    margin-left: 18px;
    font-size: 16px;
}
```

Chat.js:

```
import React, { useState, useEffect } from 'react'
import { Avatar, IconButton } from '@material-ui/core'
import SearchOutLined from '@material-ui/icons/SearchOutlined'
import InsertEmoticonIcon from '@material-ui/icons/InsertEmoticon'
import MicIcon from '@material-ui/icons/Mic'
import MoreVert from '@material-ui/icons/MoreVert'
import ContentCopyIcon from '@material-ui/icons/FileCopyOutlined'
import "./Chat.css"
import { useParams } from 'react-router-dom'
import { useStateValue } from './StateProvider'
import db from './firebase'
import firebase from 'firebase';
import InputEmoji from 'react-input-emoji'
// const Cryptr = require('cryptr');
// const cryptr = new Cryptr('ScytheChat');
var CryptoJS = require("crypto-js");

function Chat() {

    //use states
    const [input, setInput] = useState("");
    const { roomId } = useParams();
    const [roomName, setRoomName] = useState("Roommm");
    const [messages, setMessages] = useState([]);
    const [{user}, dispatch] = useStateValue();
    //Retrieve Messages
    useEffect(() => {
        if (roomId) {
            db.collection('Rooms').doc(roomId).onSnapshot(snap => (
                setRoomName(snap.data().name)
            ));
            db.collection('Rooms')
                .doc(roomId)
                .collection('messages')
                .orderBy('timestamp', 'asc')
                .onSnapshot(snap => (
                    setMessages(snap.docs.map(doc => doc.data()))
                ));
        }

    }, [roomId])
    //Copy Code Button
    async function copyCode() {
        await navigator.clipboard.writeText(roomId);
        alert("Team Code Copied")
```

```
      }

    //Encrypting messages and adding them to the database
    //console.log(messages);
    const sendMessage = (e) => {
        // e.preventDefault();
        //console.log(input);
        const encryptedMessage
=   CryptoJS.AES.encrypt(input,"ScytheChat").toString();
        db.collection('Rooms').doc(roomId).collection('messages').a
dd({
            message: encryptedMessage,
            name: user.displayName,
            timestamp:  firebase.firestore.FieldValue.serverTimesta
mp(),
        });
        setInput("");
    }



    return (
        <div className="chat">
            <div className="chat__header">
                <Avatar />
                <div className="chat__headerInfo">
                    <h3>
                        {roomName}
                    </h3>
                    <p>
                        last active{" "}{new
Date(messages[messages.length-
1]?.timestamp?.toDate()).toUTCString()}
                    </p>
                </div>
                <button onClick={copyCode}>
                Copy Code
                <IconButton><ContentCopyIcon style={{ color:
'rgb(67, 255, 224)' }} /></IconButton>
                </button>
                <IconButton><MoreVert style={{ color: 'rgb(67, 255,
224)' }} /></IconButton>

            </div>
    {/*
    Decrypting and displaying messages
```

```
    */}
            <div className="chat__body">
                {messages.map(message=>(
                <p className={`chat__message
${message.name===user.displayName && "chat__reciever"}`}>
                    <span
className="chat__name">{message.name}</span>
                    {CryptoJS.AES.decrypt(message?.message,"ScytheC
hat")?.toString(CryptoJS.enc.Utf8)}
                    <span className="chat__timestamp">
                        {new
Date(message.timestamp?.toDate()).toUTCString()}
                    </span>
                </p>
                ))}
            </div>
            <div className="chat__footer">
                {/* <InsertEmoticonIcon /> */}
                <form>
                    <InputEmoji type="text" placeholder="Type a
message" id="MyInput" value={input} onChange={setInput}
onEnter={sendMessage} />
                    <button onClick={sendMessage}
type="submit">send a message</button>
                </form>
                <MicIcon></MicIcon>
            </div>


        </div>
    )
}

export default Chat;
```

Chat.css:

```
.chat{
    display: flex;
    flex: 0.65;
    flex-direction: column;
}
.chat__header
{
    padding-left: 10px;
    display: flex;
    align-items: center;
    flex-direction: row;
    border-left: 1px solid gray;

}
.chat__header > button
{
    background-color: rgba(0, 0, 0,0);
    color: rgb(67, 255, 224);
    border: none;

}

.chat__headerInfo
{
    padding: 20px;
    align-items: center;
    flex: 1;



}
.chat__headerInfo > h3
{
    display: flex;
    margin-bottom: 0%;

}
.chat__headerInfo > p
{
    display: flex;
    margin-top: 0%;

}

.chat__body
```

```
{
    background-repeat: repeat;
    background-position: top;

    height: 100%;
    padding: 30px;
    overflow: auto;
    background-image:
url("https://wallpaperaccess.com/full/2224368.png");
    background-size: contain;

}
.chat__message
{
    position: relative;

    font-size: 16px;
    background-color:rgba(0, 0, 0, .5);
    border: thin;
    border-style: solid;
    border-color: rgb(67, 255, 255);
    padding: 10px;
    border-radius: 10px;
    width: fit-content;
    margin-bottom: 30px;

}
.chat__reciever
{
    margin-left: auto;
    background-color:rgba(67, 255, 255, .09);

}
.chat__name
{
    position: absolute;
    top: -30px;
    font-size: small;

}
.chat__timestamp
{
    margin-left: 10px;
    font-size: xx-small;
}
```

```css
.chat__footer
{
    display: flex;
    justify-content: space-between;
    align-items: center;
    height: 65px;
    border-top: 1px solid lightgray;
}
.chat__footer > form
{
    flex: 1;
    display: flex;
}
#MyInput
{
    flex: 1;
    align-items: center;
    background-color:rgba(0, 0, 0, 0.0);
    padding-left: 5px;

    border: none;
    height: 30px;
    outline: none;
    color: white;
}
.chat__footer > form > InputEmoji
{
    flex: 1;
    align-items: center;
    background-color:rgba(0, 0, 0, 0.0);
    padding-left: 5px;

    border: none;
    height: 30px;
    outline: none;
    color: white;
}
.chat__footer > form > button
{
    display: none;
}
.chat__footer > .MuiSvgIcon-root
{
    color: lightgray;
}
```

Firebase.js:

```
import firebase from 'firebase'
require('firebase/auth');

const firebaseConfig = {
  apiKey: "AIzaSyBr4_zD9thZ0v-vlIkoZgkOyVH4qfoieYY",
  authDomain: "scythe-chat.firebaseapp.com",
  projectId: "scythe-chat",
  storageBucket: "scythe-chat.appspot.com",
  messagingSenderId: "948298265406",
  appId: "1:948298265406:web:3dfab59c3c4ebd8157326f"
};

  const firebaseApp = firebase.initializeApp(firebaseConfig);
  const db = firebaseApp.firestore();
  const auth = firebase.auth();
  const provider = new firebase.auth.GoogleAuthProvider();

  export {auth,provider};
  export default db;
```

Firebase.json:

```
{
  "hosting": {
    "public": "build",
    "ignore": [
      "firebase.json",
      "**/.*",
      "**/node_modules/**"
    ],
    "rewrites": [
      {
        "source": "**",
        "destination": "/index.html"
      }
    ]
  },
  "emulators": {
    "auth": {
      "port": 9099
    },
    "firestore": {
      "port": 8080
    },
```

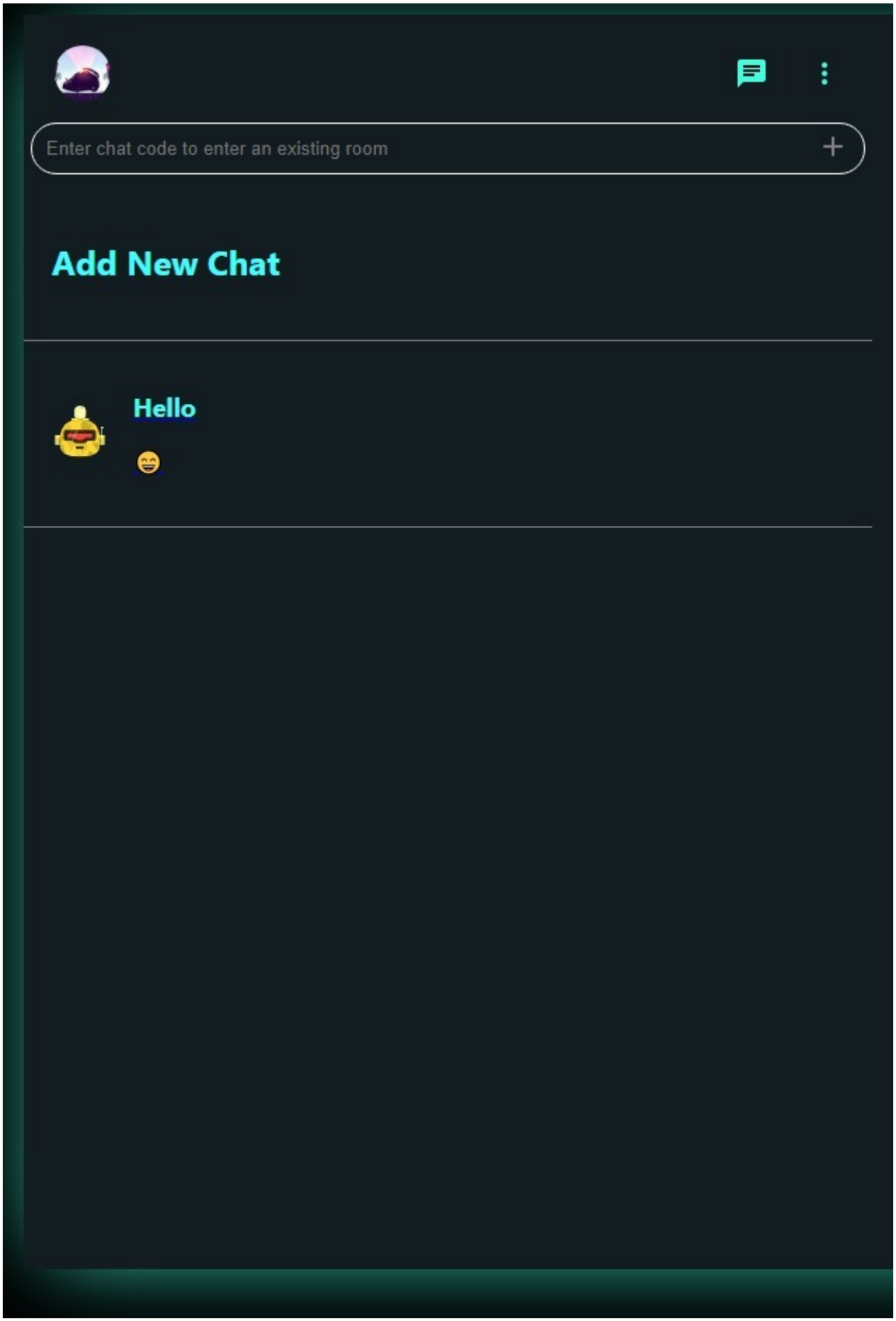
```
    "hosting": {
      "port": 5000
    },
    "ui": {
      "enabled": true
    }
  }
}
```

RESULTS AND SCREENSHOTS

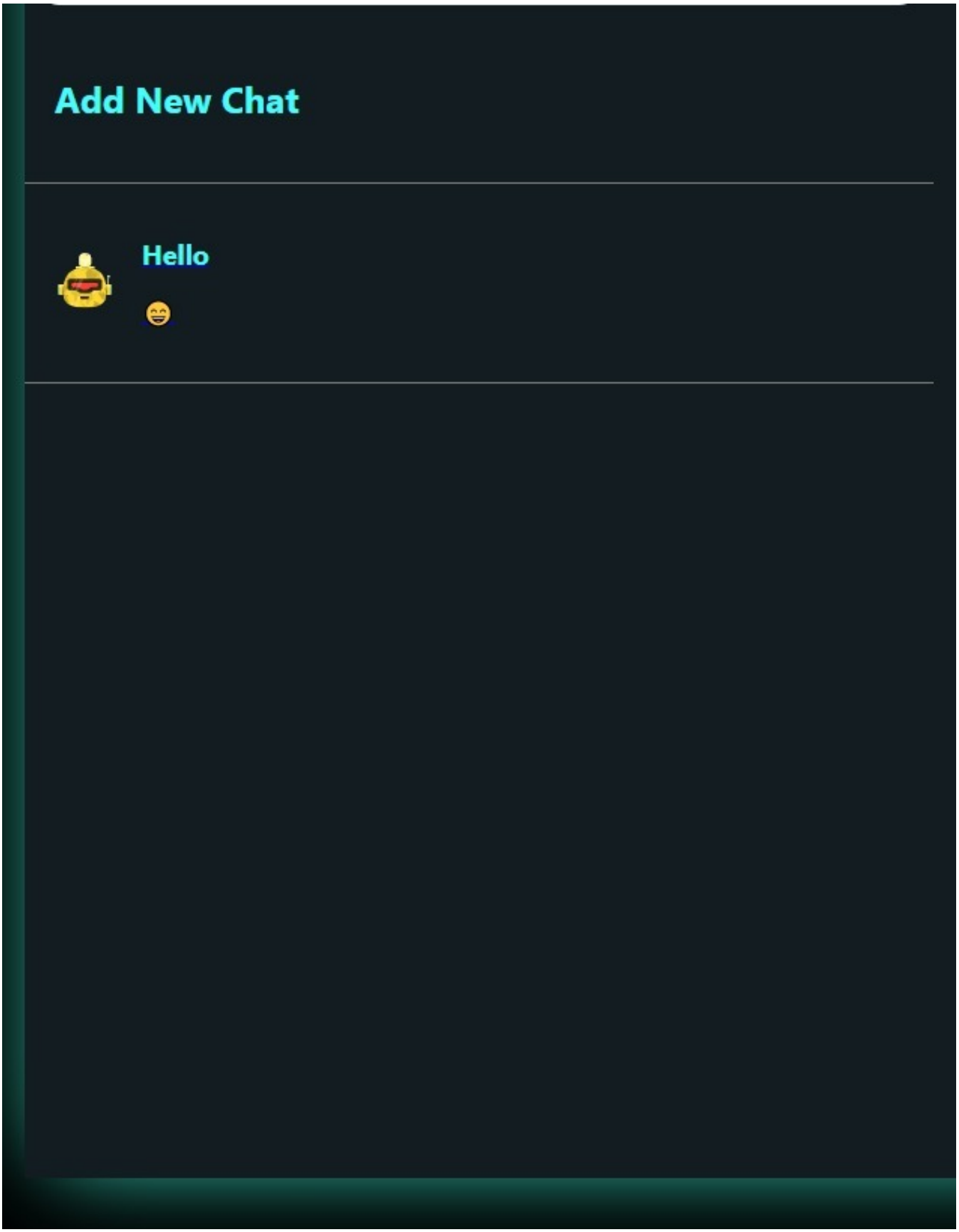
Components:

Login Screen:

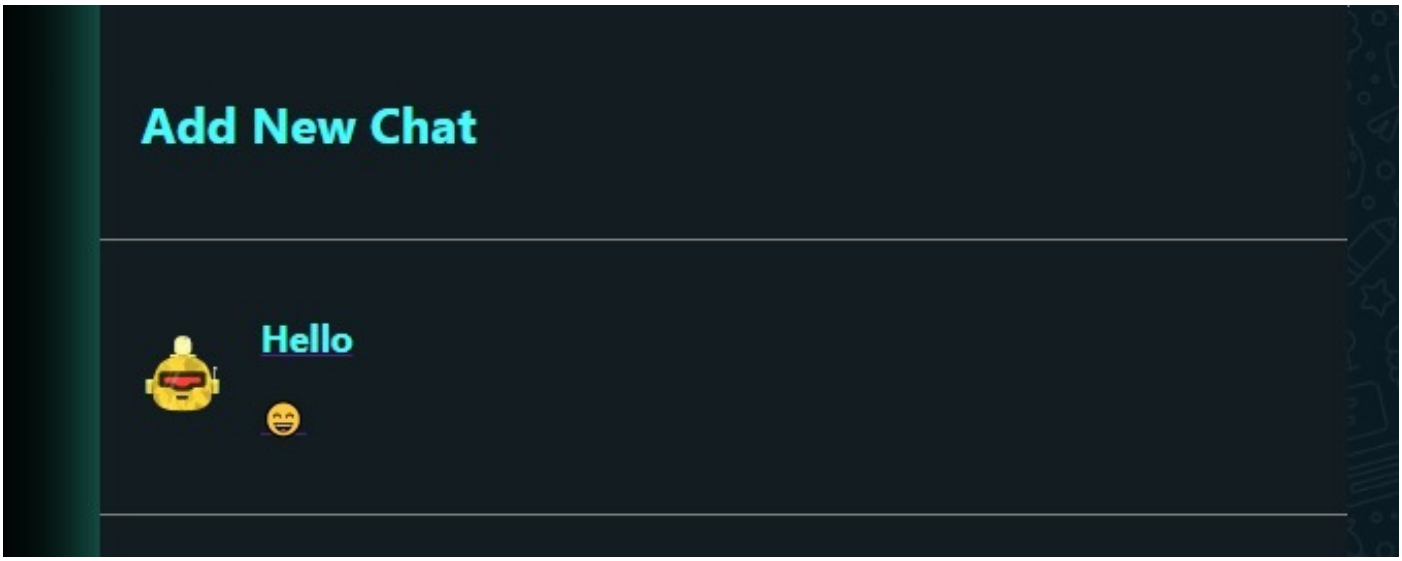
Sidebar:



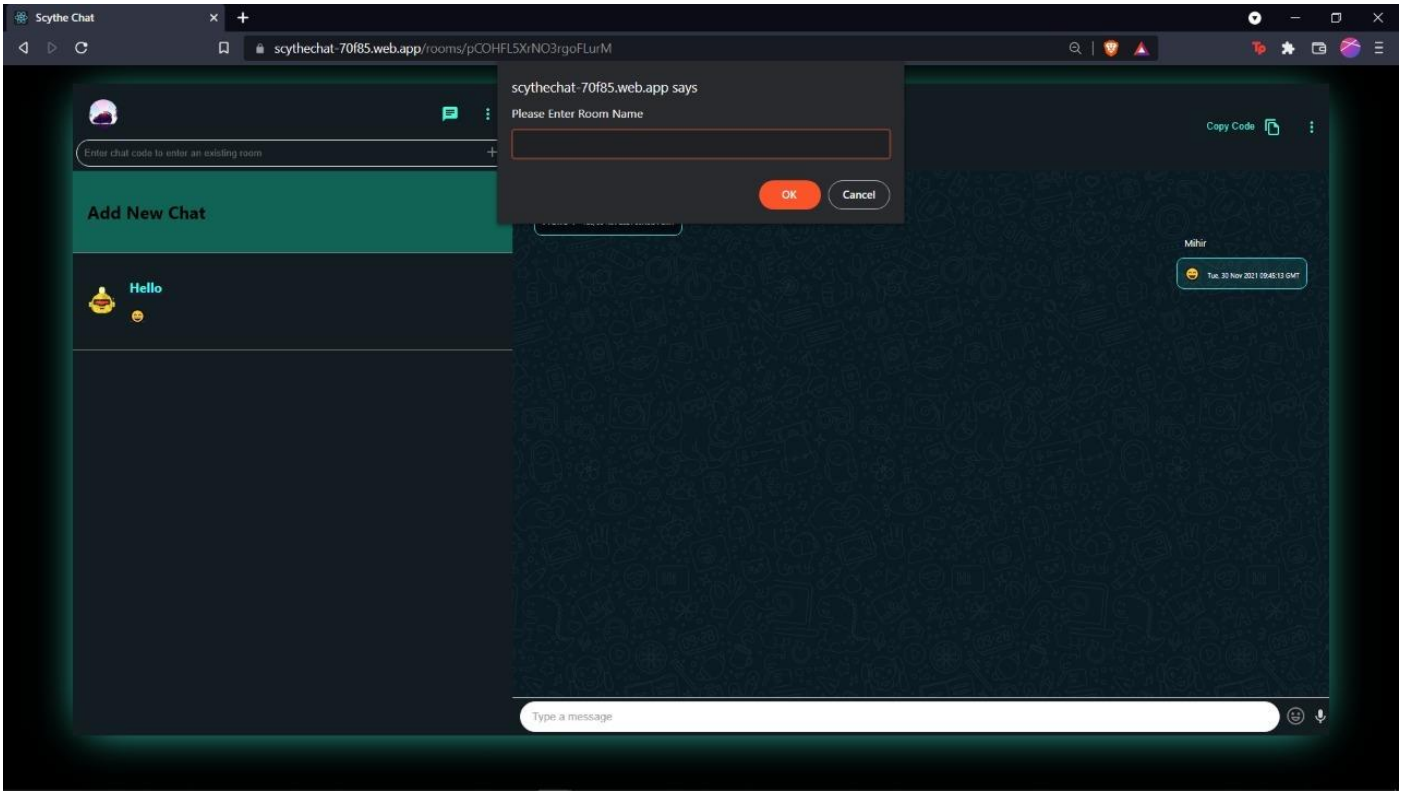
Sidebar Chats:



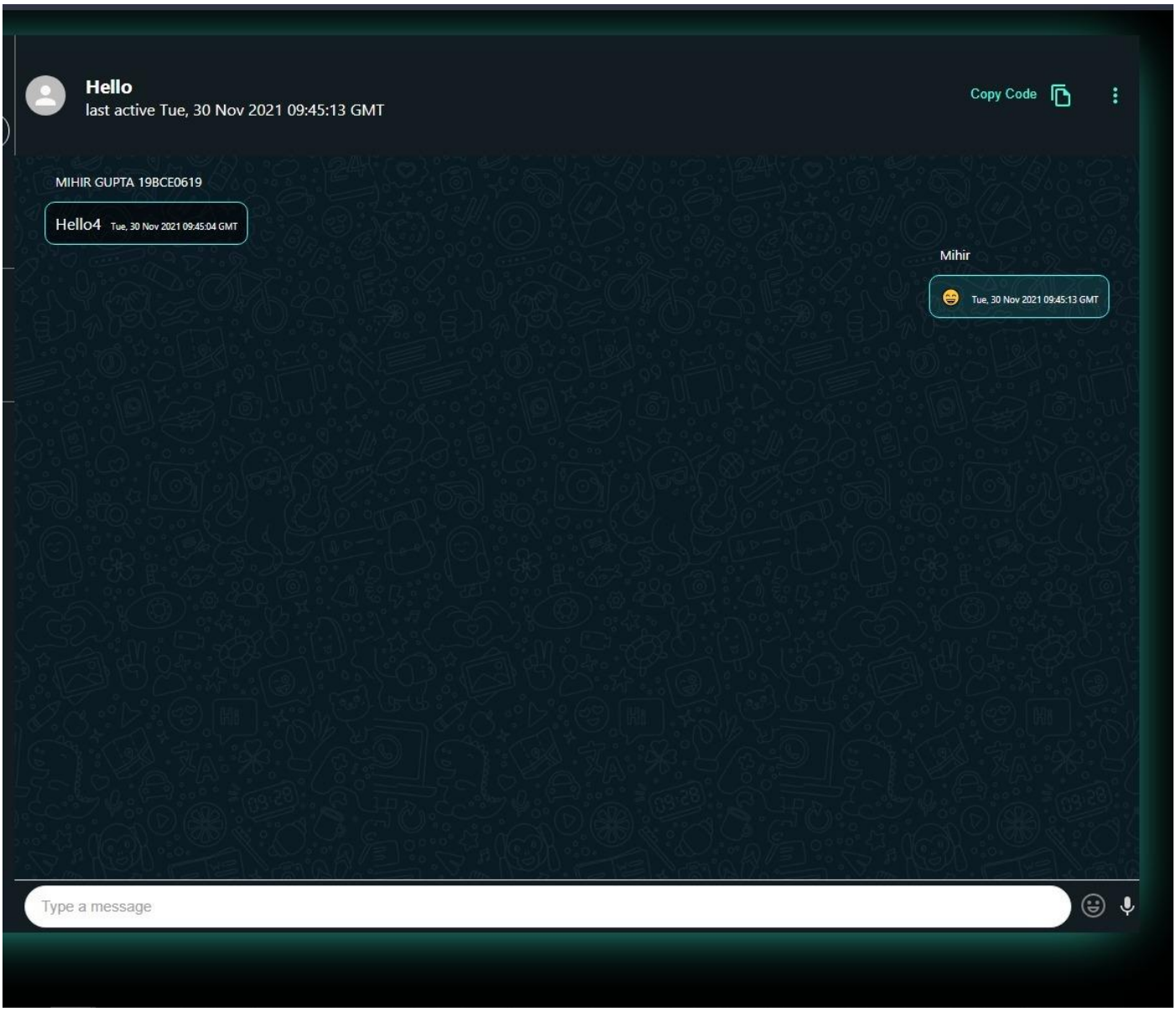
Sidebar Chat:



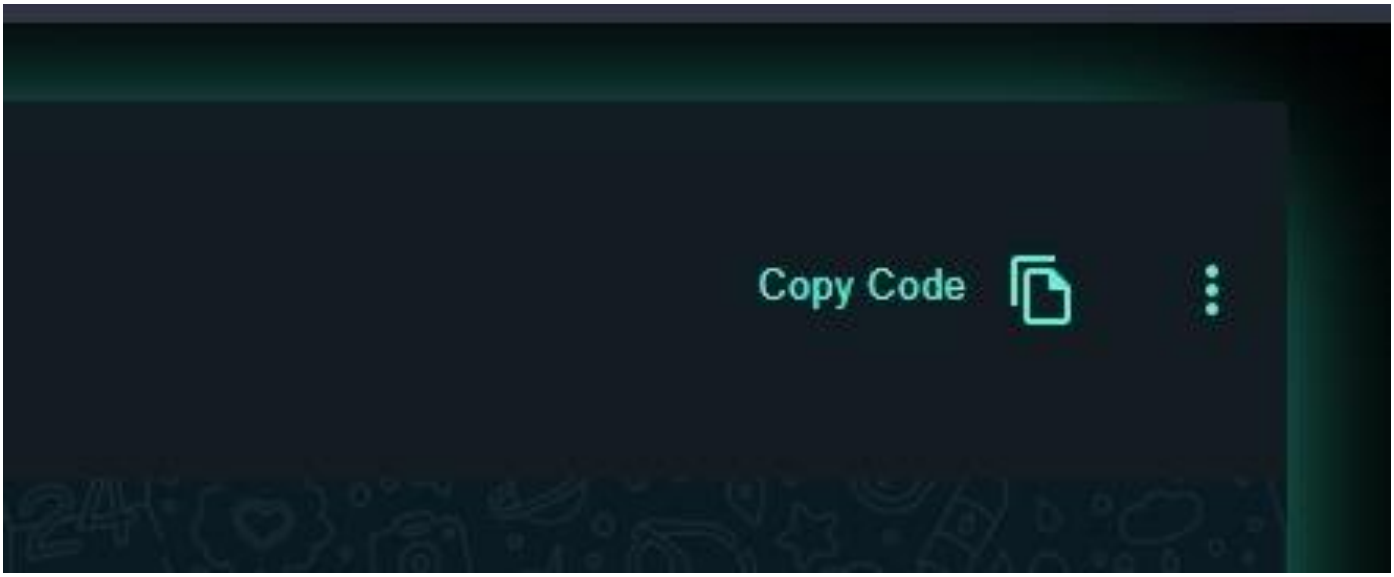
Add New Chat:



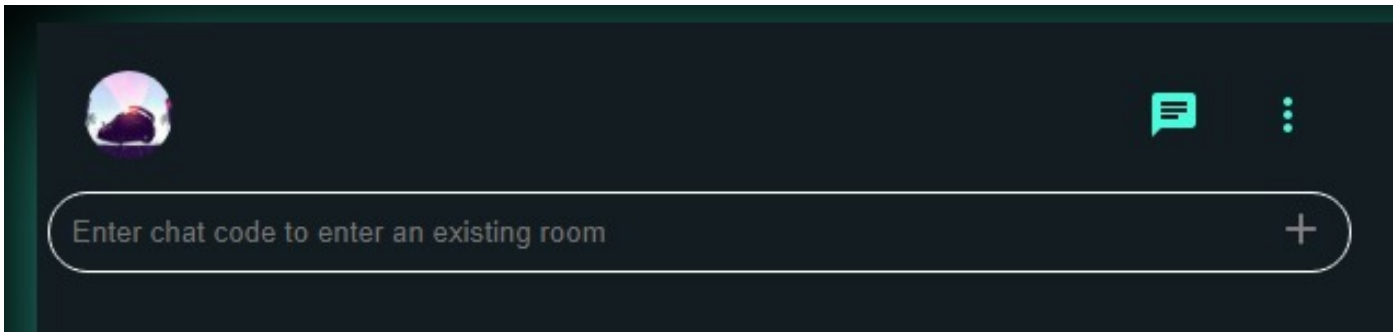
Chat messages page:



Copy room code to add more people:

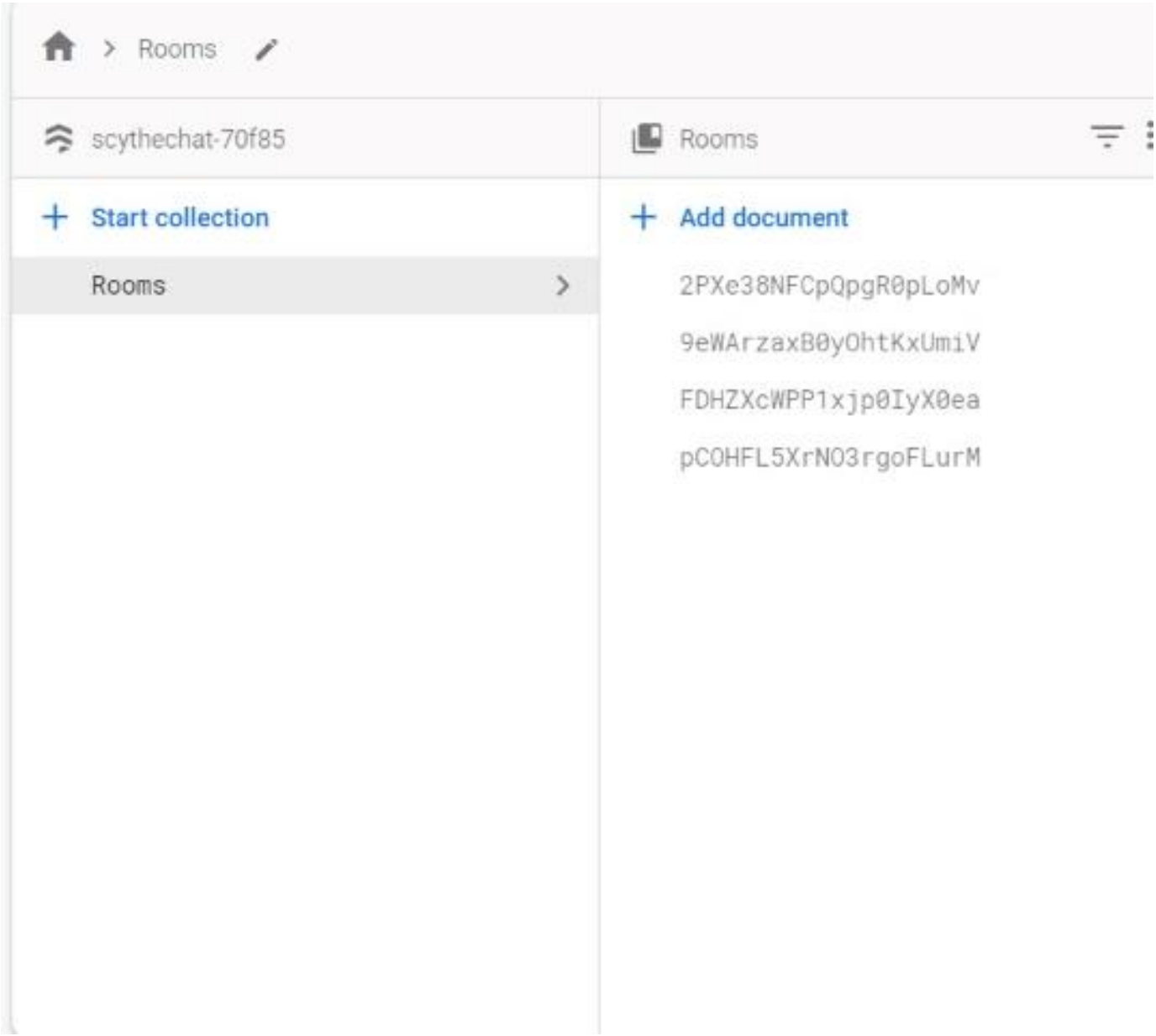


Join existing room:

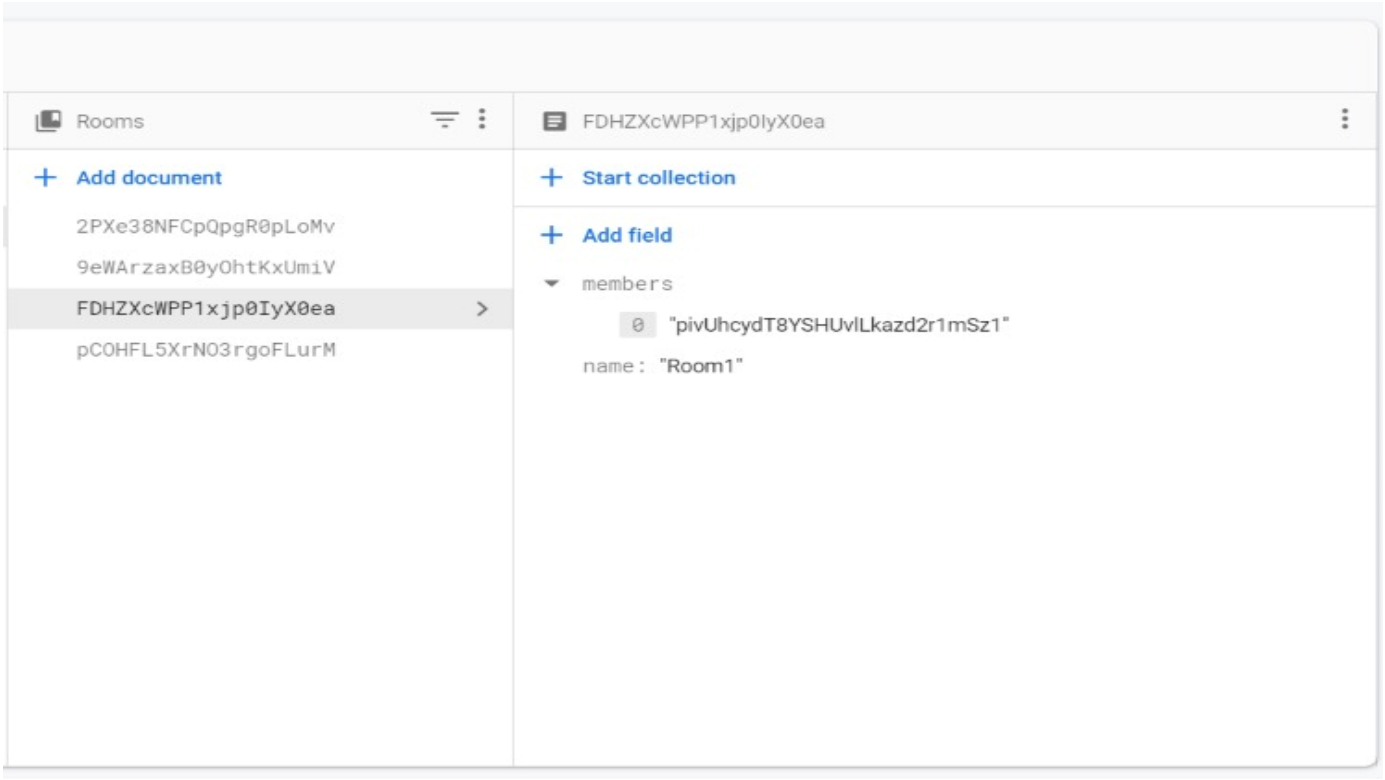


Database:

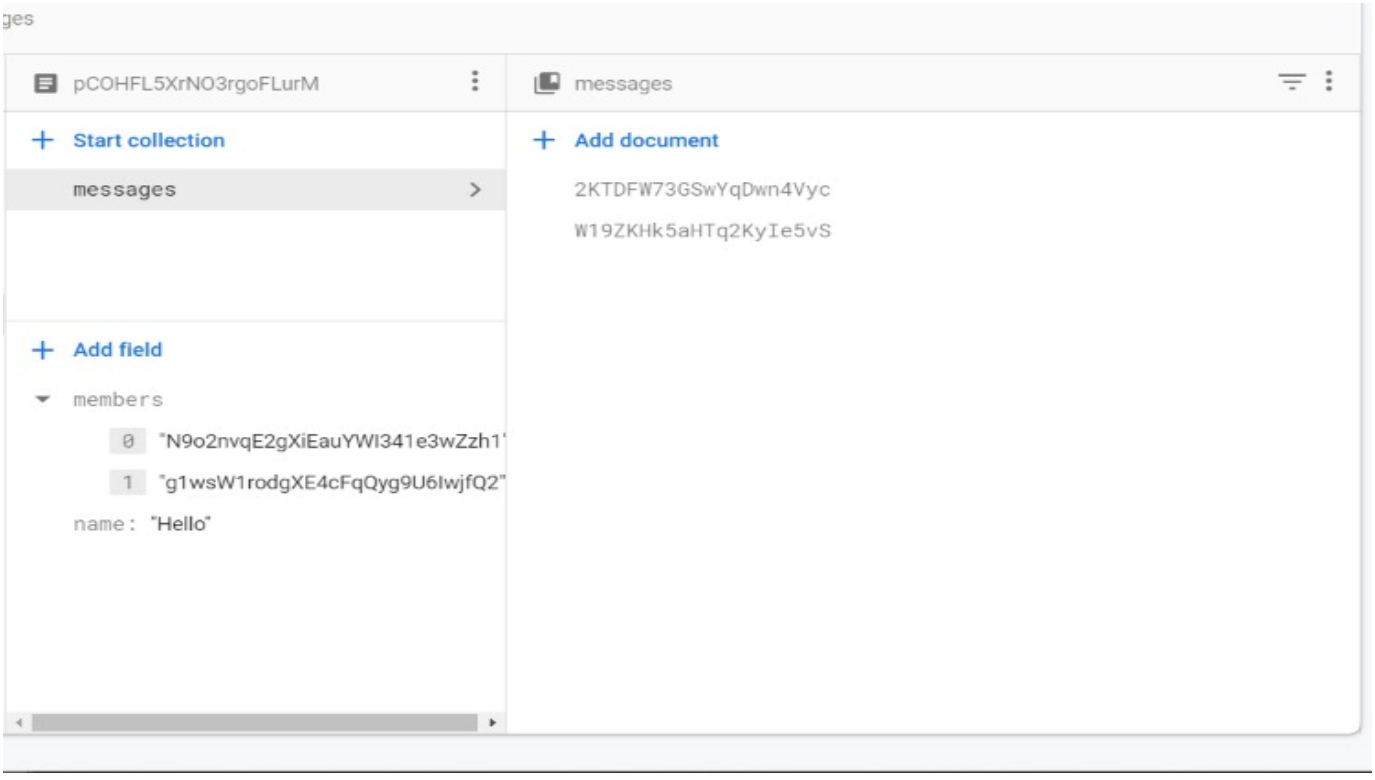
Rooms collection structure:



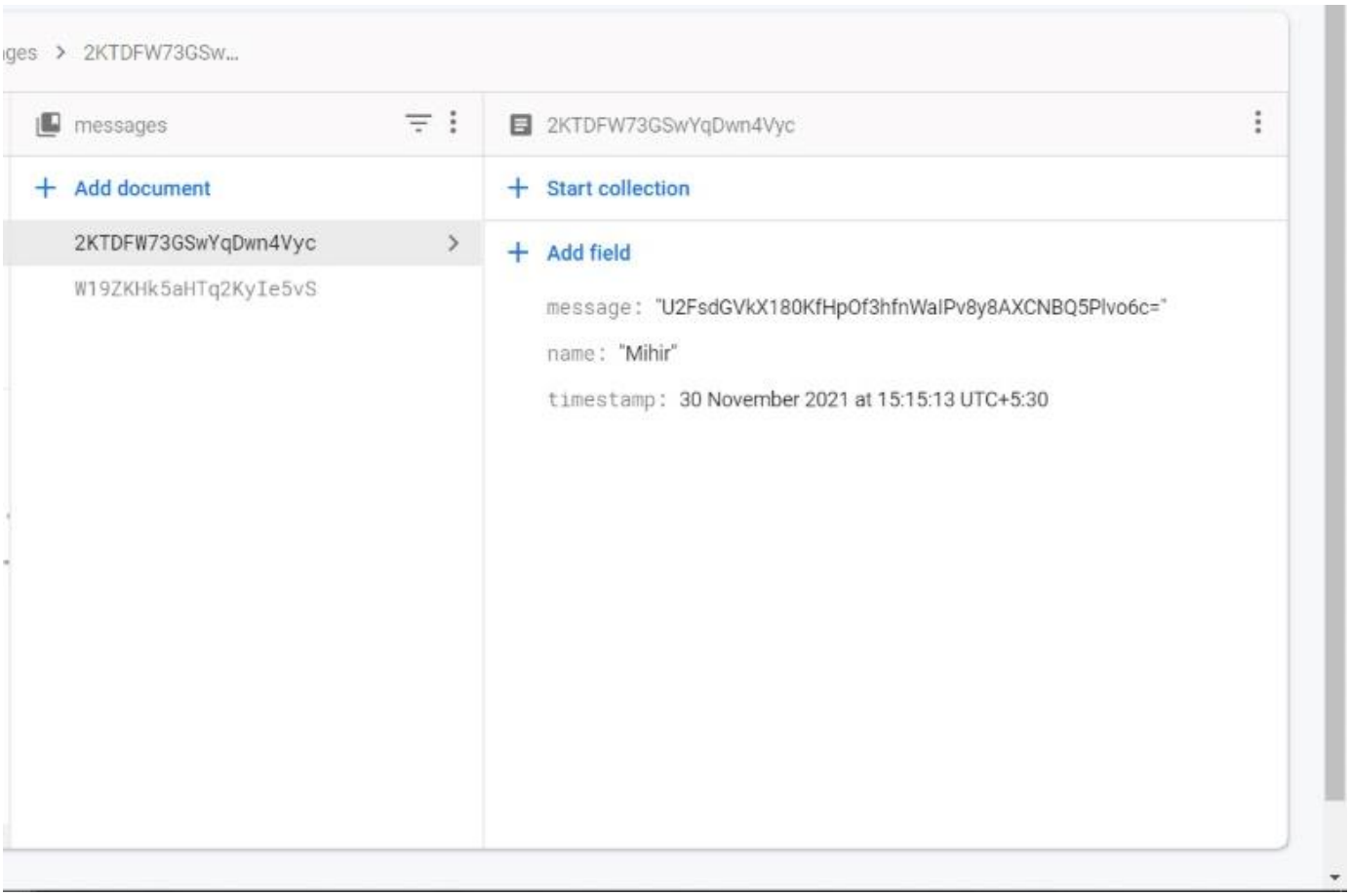
Room document structure:



Messages collection structure:



Room document structure:



CONCLUSION

As the research demonstrated, the plan was to build a chatting application which will allow the users to text in groups with one other easily and securely. This goal was achieved by the end of this project. By the end of this project, we created a group chatting application using ReactJS as the frontend framework and Google cloud firestore as the backend database. We also used multiple APIs like Google Authentication, MaterialUI-core, MaterialUI-icons, dicebear and CryptoJS to smoothen the user experience. We have also implemented the invite only feature to the groups by creating a unique invite code for each group which can be used to enter that group. We have also used redux like technique called reducer to access user info in required components.

We have also achieved real-time database updating using the snapshots of firebase firestore database. This provides user a quick service to send the messages without having heavy load on their device. The snapshots technique is better than actual real-time database as actual real-time

database floods the user network by constant pings of updating, while the snapshots technique only send the updating ping when the user tries to update the database.

Overall, these components combine to form Scythechat.

FUTURE WORK

The future works that can be used to improve our chat application include the following:

- Making the app more user friendly:
  We can add more login options to make the user experience more friendly.
  We can also add multimedia upload option to our project.

- Increase Scalability of the web-application:
  Right now, we are using the free services provided by cloud firestore. In future we can upgrade to better firestore services or any other cloud providers like azure and AWS.

- Improve UI/UX:
  We can smoothen the UI by adding more css animations and bootstrap.

- Add one on one chatting feature

REFERENCES

- https://reactjs.org/docs/create-a-new-react-app.html
- https://create-react-app.dev/
- https://mui.com/
- https://v4.mui.com/
- https://www.npmjs.com/package/crypto-js
- https://firebase.google.com/
- https://console.firebase.google.com/u/0/
- https://www.npmjs.com/package/dicebear