A Project report on

Join Chat

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology
In
Computer Science and Engineering

Submitted by

G. Kiran Deepak	**(19H51A0570)**
K.V.S.S.S.R.H Sri Harsha	**(19H51A0573)**
G. Pranay Kumar	**(19H51A05F8)**

Under the esteemed guidance of

Dr. A. Poongodai
(Associate Professor)

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC & JNTUH , Approved by AICTE,
Permanently Affiliated to JNTUH, Accredited by NBA.)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401

.

2019- 2023

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project-2 report **"Join Chat"** being submitted by **G. Kiran Deepak (19H51A0570), K.V.S.S.S.R.H Sri Harsha(19H51A0573), G.Pranay Kumar (19H51A05F8)**, in partial fulfillment for the award of **Bachelor of Technology** in **Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Dr. A.Poongodai
Associate Professor
Dept. of CSE

Sivaskanda
Professor and HOD
Dept. of CSE

Acknowledgement

With great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to Dr A.Poongodai, Associate Professor , Dept of Computer Science and Engineering for her valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. S.Siva Skandha,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Vijaya Kumar Koppula**, Dean-Academic, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the Teaching & Non- teaching staff of Department of Computer Science and Engineering for their co-operation

Finally we express our sincere thanks to Mr. Ch. Gopal Reddy, Secretary, CMR Group of Institutions, for his continuous care. I sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

G.KIRAN DEEPAK (19H51A0570)
K.V.S.S.S.R.H SRI HARSHA (19H51A0573)
G.PRANAY KUMAR (19H51A05F8)

TABLES OF CONTENTS

S.No.	CONTENTS	Page No
	ABSTRACT	1
1	**INTRODUCTION**	2-3
1.1	Description	2
1.2	Objectives	3
2	**BACKGROUND WORK**	**4-5**
2.1	Existing Solution	4-5
3	**PROPOSED SYSTEM**	**6-10**
3.1	Block diagram	6
3.2	Methodology	6-7
3.3	Technologies used	7-10
3.5	Hardware Requirements	10
3.6	Software Requirements	10
4	**DESIGNING**	**11**
4.1	Algorithmic flowchart	11
5	**RESULT AND DISCUSSION**	**12-28**
5.1	Implementation code	12-23
5.2	Result	24-28
6	**CONCLUSION AND FUTUREWORK**	29
6.1	Conclusion	29
6.2	Future work	29
7	**REFERENCES**	30

IMAGE INDEX

S.no	Fig number	Page number
1	2.1	4
2	2.2	4
3	2.3	5
4	2.4	5
5	3.1	6
6	4.1	11
7	5.1	24
8	5.2	24
9	5.3	25
10	5.4	25
11	5.5	26
12	5.6	26
13	5.7	27
14	5.8	27
15	5.9	28
16	5.10	28

ABSTRACT

Chat refers to the process of communicating, interacting or exchanging messages over the Internet. It involves two or more individuals that communicate through a chat-enabled service or software. Chat may be delivered through text, audio or video communication via the Internet. A chat application has basic two components, server and client. A server is a computer program or a device that provides functionality for other programs or devices. Clients who want to chat with each other connect to the server.The chat application we are going to make will be more like a chat room, rather than a peer to peer chat. So this means that multiple users can connect to the chat server and send their messages. Every message is broadcasted to every connected chat user.

1.INTRODUCTION

1.1DESCRIPTION

Name of our project is **Join Chat**. The main purpose of this project is to build an online real time chatting web application which helps internet users who are online to communicate and share files. Existing Chat applications need a criteria like phone number or mail Id to login but in this there is no need of such data but rather the user is required to enter his user id and room id so, that all his/her friends can join in that room and communicate by messages and also they can share files in that room. This application is secure because the messages in the room are not saved in any database.

Once the users are in the same chat room, they can converse with one another by typing messages into a window where all of the other users in the chat room can see the message. The user can also see all of the messages entered by the other users. Conversations are then carried on by reading the messages entered by the other users in the chat room and responding to them. They can also send files and download them. To develop an instant messaging solution to enable users to seamlessly communicate with each other. The project should be very easy to use enabling even a vice person to use it. This project can play an important role in organizational field where employees can connect through LAN.

Chatting is a method of using technology to bring people and ideas together despite of the geographical barriers. The technology has been available for years but the acceptance was quite recent. Our project is an example of a chat server. It is made up of two applications-the client application, which runs on the user's web browser and server application, runs on any hosting servers on the network. To start chatting client should get connected to server where they can do private and group chat. Security measures were taken during the last one The MERN stack which consists of Mongo DB, Express.js, Node.js, and React.js is a popular stack for building full-stack web-based applications because of its simplicity and ease of use. In recent years, with the explosive popularity and the growing maturity of the JavaScript ecosystem, the MERN stack has been the go to stack for a large number of web applications. This stack is also highly popular among newcomers to the JS field because of how easy it is to get started with this stack. This is a full-stack chat application that can be up and running with just a few steps. Its frontend is built with Material UI running on top of React. The backend is built with Express.js and Node.js. Real-time message broadcasting is developed using Socket.IO.

1.2 OBJECTIVES

The main objective of this project is to allow each and every person to get connected with each other.

- Providing a social platform to users.
- Allows users to share files.
- To allow each person to share their thoughts & views.
- This System can be used as a discussion board. Connecting people together.
- Allowing the user to use anywhere anytime.

2.BACKGROUND WORK

2.1 EXISTING SOLUTION

1.Facebook:

Fig 2.1

Facebook is an online social media and social networking service owned by American company Meta Platforms. Founded in 2004 by Mark Zuckerberg .

Facebook can be accessed from devices with Internet connectivity, such as personal computers, tablets and smartphones. After registering, users can create a profile revealing information about themselves. They can post text, photos and multimedia which are shared with any other users who have agreed to be their "friend" or with different privacy settings, publicly. Users can also communicate directly with each other with Facebook Messenger.

2.Instagram:

Fig 2.2

Instagram is a photo and video sharing social networking service owned by American company Meta Platforms, it was launched in 2010 by Kevin Systrom. The app allows users to upload media that can be edited with filters and organized by hashtags and geographical tagging. Posts can be shared publicly or with pre approved followers.

3.WhatsApp:

Fig 2.3

It is owned by American company Meta Platforms (formerly Facebook). It allows users to send text and voice messages, make voice and video calls, and share images, documents, user locations, and other content. WhatsApp's client application runs on mobile devices, and can be accessed from computers. The service requires a cellular mobile telephone number to sign up.

Fig 2.4

User needs to send a chat request before sending any messages to any user. When users accept their request then only they send the messages otherwise not. User have to login their application with their email id and password.

3.PROPOSED SYSTEM

3.1 Block diagram of the chat Application

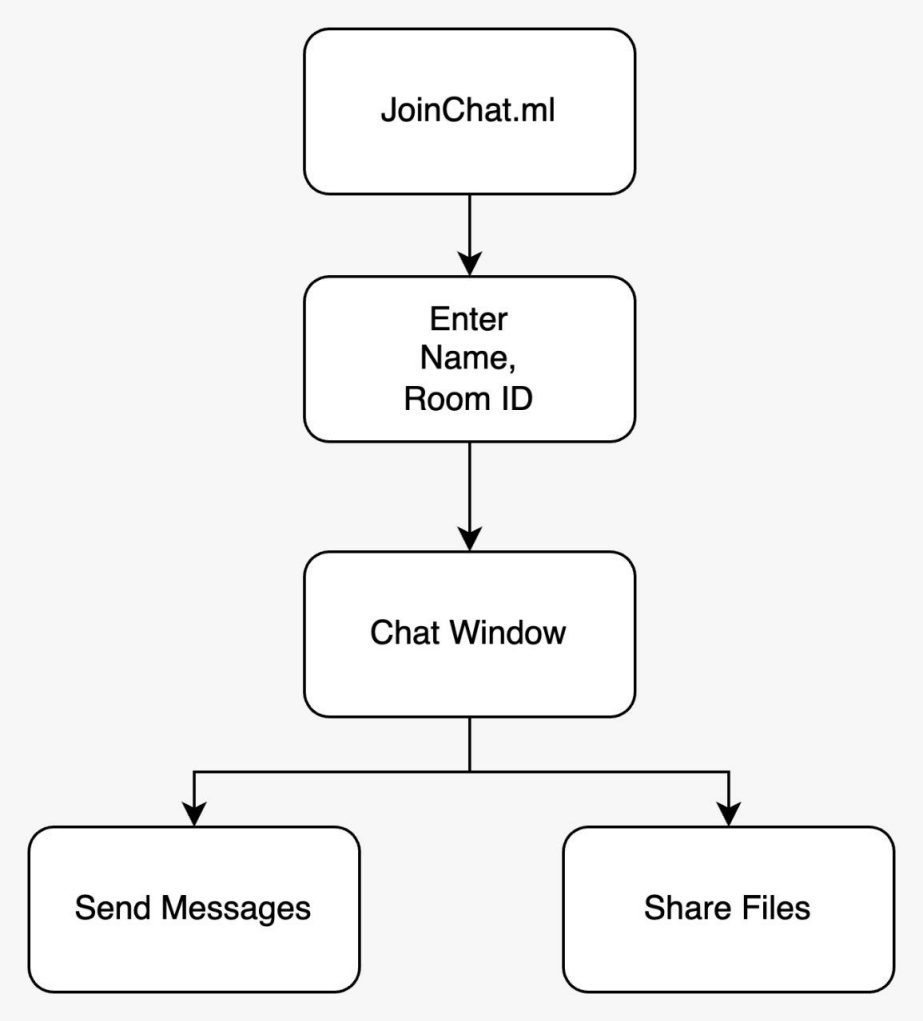

Fig3.1

3.2 METHODOLOGY

This project is to create a chat application and enable the users to chat with each other on the web.

Until now to share any url or attachment from your mobile to our system or friend's system we had to login a social media account like whatsapp or google drive in both our mobile and system which is a bit time consuming and not always we get the need to share files to just our personal systems but to our friends and we don’t want to login our whatsapp in others device which is a hassle but not anymore with [joinchat] enables us to instantly create a room without any sign up and share texts, links, files to our friends or ourselves from our mobile to pc or pc to pc any way how we want....

This project is made up on the idea to create an instant ,messaging and file sharing solution to enable users to seamlessly communicate and share files without signing up and to do it from any device.

Join chat has been developed in order to overcome the difficulties encountered while using the mailing system for communication between the users. Providing user friendly communication channel, live communication facility, no need to sign up, the accessibility to access it from any device whether it is an android or ios or windows or mac or linux , to be able to send files and all of this with ensured privacy as nothing gets stored on the server.

3.3Technologies Used

3.3.1 React

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library responsible only for the view layer of the application. It was created by Jordan Walke, who was a software engineer at Facebook. It was initially developed and maintained by Facebook and was later used in its products like WhatsApp & Instagram. Facebook developed ReactJS in 2011 in its newsfeed section, but it was released to the public in the month of May 2013.

Today, most of the websites are built using MVC (model view controller) architecture. In MVC architecture, React is the 'V' which stands for view, whereas the architecture is provided by the Redux or Flux.

A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

To create React app, we write React components that correspond to various elements. We organize these components inside higher level components which define the application structure. For example, we take a form that consists of many elements like input fields, labels, or buttons. We can write each element of the form as React components, and then we combine it into a higher-level component, i.e., the form component itself. The form components would specify the structure of the form along with elements inside of it.

3.3.2 Mongo DB

MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term ‘NoSQL’ means ‘non-relational’. It means that MongoDB isn’t based on the table-like

relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON (similar to JSON format). A simple MongoDB document Structure:

```
{
title: 'Student',
by: 'Harsha',
url: 'https://studentinfo.org',
type: 'NOSQL'
}
```

SQL databases store data in tabular format. This data is stored in a predefined data model which is not very much flexible for today's real-world highly growing applications. Modern applications are more networked, social and interactive than ever. Applications are storing more and more data and are accessing it at higher rates.

Relational Database Management System(RDBMS) i**s** not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable. If the database runs on a single server, then it will reach a scaling limit. NoSQL databases are more scalable and provide superior performance. MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

3.3.3 Node Js

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project!Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

When Node.js performs an I/O operation, like reading from the network, accessing a database or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back. This allows Node.js to handle thousands of concurrent connections with a single server without introducing the burden of managing thread concurrency, which could be a significant source of bugs.

Node.js has a unique advantage because millions of frontend developers that write JavaScript for the browser are now able to write the server-side code in addition to the client-side code without the need to learn a completely different language. In Node.js the new ECMAScript standards can be used without problems, as you don't have to wait for all your users to update their browsers - you are in charge of deciding which ECMAScript version to use by changing the Node.js version, and you can also enable specific experimental features by running Node.js with flags.

3.3.4 Express Js

Express is the most popular *Node* web framework, and is the underlying library for a number of other popular Node web frameworks. It provides mechanisms to:

- Write handlers for requests with different HTTP verbs at different URL paths (routes).
- Integrate with "view" rendering engines in order to generate responses by inserting data into templates.
- Set common web application settings like the port to use for connecting, and the location of templates that are used for rendering the response.
- Add additional request processing "middleware" at any point within the request handling pipeline.

While *express* itself is fairly minimalist, developers have created compatible middleware packages to address almost any web development problem. There are libraries to work with cookies, sessions, user logins, URL parameters, POST data, security headers, and *many* more. You can find a list of middleware packages maintained by the Express team at Express Middleware (along with a list of some popular 3rd party packages).

3.3.5 Vercel

Vercel is the platform for frontend developers, providing the speed and reliability innovators need to create at the moment of inspiration.

They enable teams to iterate quickly and develop, preview, and ship delightful user experiences. Vercel has zero-configuration support for 35+ frontend frameworks and integrates with your headless content, commerce, or database of choice.

3.3.6WebSocket

The WebSocket API is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.The primary interface for connecting to a WebSocket server and then sending and receiving dataon the connection. The event sent by the WebSocket object when a message is received from the server.
Either a single protocol string or an array of protocol strings. These strings are used to indicate sub-protocols, so that a single server can implement multiple WebSocket sub-protocols (for example, you might want one server to be able to handle different types of interactions depending on the specified protocol). If you don't specify a protocol string, an empty string is assumed.
The constructor will throw a Security Error if the destination doesn't allow access. This may happen if you attempt to use an insecure connection (most user agents now require a secure link for all WebSocket connections unless they're on the same device or possibly on the same network).
WebSockets is an event-driven API; when messages are received, a message event is sent to the WebSocket object. To handle it, add an event listener for the message event, or

use the on message event handler. To begin listening for incoming data, you can do something like this.

3.3.7 Heroku

Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. Our platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market.

Heroku is fully managed, giving developers the freedom to focus on their core product without the distraction of maintaining servers, hardware, or infrastructure. The Heroku experience provides services, tools, workflows, and polyglot support—all designed to enhance developer productivity.

3.5 Hardware Requirements

1.Windows Server with Intel/AMD x86 or x64 processor equivalent (Virtual Machines fully supported)
2.Memmory 2 GB minimum, 4 GB recommended
3.OS: Linux (or) Ubuntu (or) Windows (or) Mobile applications
4.Android phone
5. Internet or LAN connection
Processor with speed 500MHZ

3.6 Software Requirements

1.Any Web Browser
Ex: Google chrome, mozilla firefox

4. DESIGNING

4.1 DATA FLOW DIAGRAM

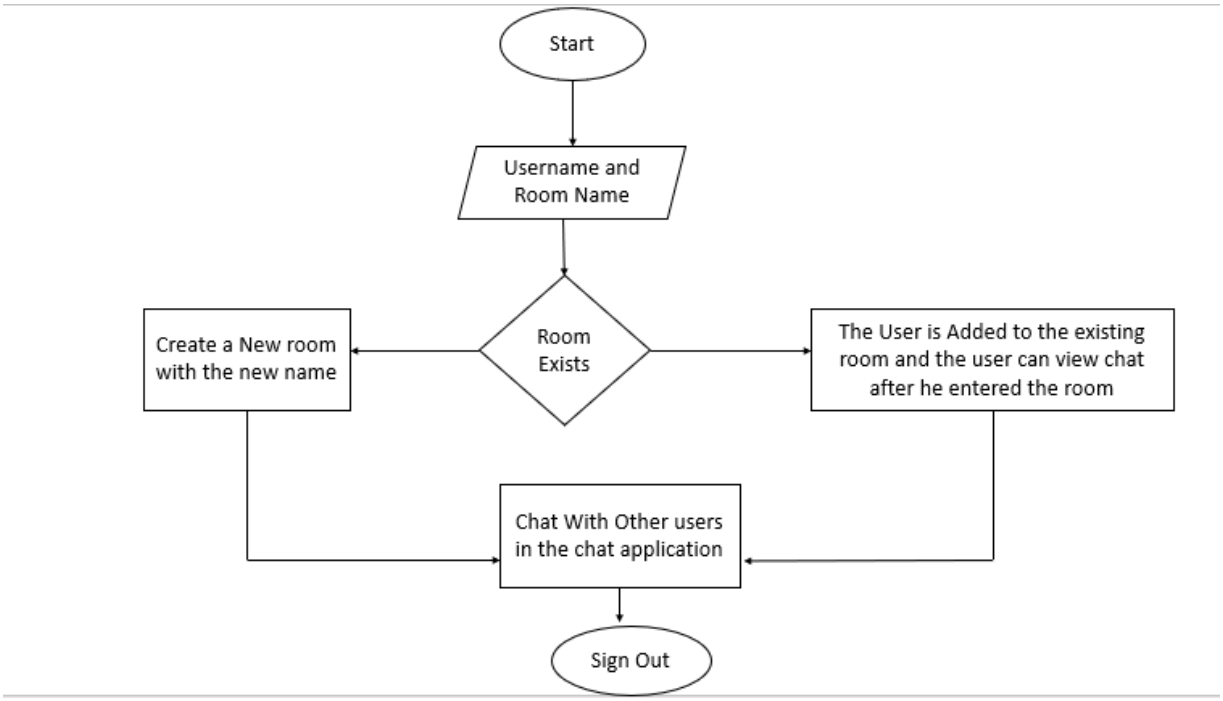

Fig 4.1

5.RESULT AND DISCUSSION

5.1 IMPLEMENTATION

MAIN CODE:
https://github.com/gpranaykumar/FileSharing-Server
https://github.com/gpranaykumar/joinChat_V3_client
https://github.com/gpranaykumar/joinchat_v2_server

main.js

```
import React,{useState} from 'react'
import io from "socket.io-client";
import Chat from './Chat';
const socket = io.connect("https://joinchatv2.herokuapp.com/");

function Main() {
   const [username, setUsername] = useState("")
   const [room, setRoom] = useState("");
   const [showChat, setShowChat] = useState(false);

   const joinRoom = () => {
   if(username !== "" && room !== ""){
     socket.emit("joinRoom", {username, room} );
     setShowChat(true);
   }
  }
   return <>
      {!showChat?(
         <div className='bg-white dark:bg-neutral-800 rounded rounded-lg'>
        <h1 className=' p-5 text-blue-600 dark:text-blue-400 text-center text-xl font-
bold'>Join Chat</h1>
        <div className='mb-4'>
          <label className='relative cursor-pointer'>
            <input type="text" placeholder=" "
            className='h-14 w-64 px-6 mx-4 text-xl text-black dark:text-white
            bg-white dark:bg-neutral-800 border-black dark:border-white border-
opacity-30 hover:border-opacity-100 dark:border-opacity-30 dark:hover:border-
opacity-100 border rounded-lg border-opacity-50
            outline-none focus:border-blue-500 dark:focus:border-blue-500 placeholder-
gray-300 placeholder-opacity-0 transition duration-200'
            onChange={(e) => setUsername(e.target.value)}/>
            <span className='text-xl text-black dark:text-white text-opacity-60 bg-white
dark:bg-neutral-800 absolute left-5 top-0.5 px-1
            transition duration-200 input-text dark:text-opacity-60'>Name</span>
```

```
          </label>
        </div>
        <div className='mb-4'>
          <label className='relative cursor-pointer'>
            <input type="text" placeholder=" "
            className='h-14 w-64 px-6 mx-4 text-xl text-black dark:text-white
            bg-white dark:bg-neutral-800 border-black dark:border-white border-
opacity-30 hover:border-opacity-100 dark:border-opacity-30 dark:hover:border-
opacity-100 border rounded-lg border-opacity-50
            outline-none focus:border-blue-500 dark:focus:border-blue-500 placeholder-
gray-300 placeholder-opacity-0 transition duration-200'
            onChange={(e) => setRoom(e.target.value)}
            onKeyPress={(event) => {
              event.key === "Enter" &&  joinRoom();
          }}  />
            <span className='text-xl text-black dark:text-white text-opacity-60 bg-white
dark:bg-neutral-800 absolute left-5 top-0.5 px-1
            transition duration-200 input-text dark:text-opacity-60'>Room Id</span>
          </label>
        </div>
        <div className='mb-5 flex justify-center'>
          <button type='button'
          className='px-16 py-3 rounded dounded-xl
          text-lg font-semibold text-blue-600
          border border-blue-400
          dark:text-blue-600 dark:bg-neutral-800
          hover:border-blue-600 hover:bg-blue-100 hover:bg-opacity-40
          dark:hover:bg-blue-100 dark:hover:bg-opacity-5' onClick={() => joinRoom()}>
            JOIN ROOM
          </button>
        </div>

      </div>
      ): (
          <Chat socket={socket} username={username} room={room} />
          )}
    </>

}

export default Main
```

Chat.js

```
import React,{ useEffect, useState } from 'react'
import ScrollToBottom from "react-scroll-to-bottom";
import { CopyToClipboard } from "react-copy-to-clipboard";
import { IoDocumentAttachSharp, IoSend } from "react-icons/io5";
import { IoMdAttach } from "react-icons/io";
import axios from 'axios';
```

```
import { HiClipboardCopy, HiOutlineClipboardCopy } from "react-icons/hi";

function Chat({ socket, username, room }) {
   const [currentMessage, setCurrentMessage] = useState("");
   const [messageList, setMessageList] = useState([]);
   const [users, setUsers] = useState([]);
   const [copyBtn,setCopyBtn] = useState(false)
   const [selectedFile, setSelectedFile] = useState(null);
   const [fileLoading, setFileLoading] = useState(false)
   const [fileErr, setFileErr] = useState(false)

   useEffect(() => {
      if(copyBtn){
        setTimeout(() => {
          setCopyBtn(false)
        }, 3000)
      }
      if(fileErr){
         setTimeout(() => {
           setFileErr(false)
         }, 3000)
        }
    },[copyBtn, fileErr])
   const sendMessage = async () => {

      if( selectedFile !== null){
         setFileLoading(true)
         const formData = new FormData();
         formData.append("myfile", selectedFile);
         try {
            const response = await axios({
               method: "post",
               url: "https://file-sharing-gpk.herokuapp.com/api/files",
               data: formData,
               headers: { "Content-Type": "multipart/form-data" },
            });
            console.log("File-upload success");
            console.log(response)
            if(response.data.err){
               console.log("Unable to Upload file")
               console.log(response.data.err)
               setFileErr(true)
            }else{
               const link = response.data.file
               const messageData = {
                  room: room,
                  author: username,
                  message: link,
```

```
                time:
                new Date(Date.now()).getHours() +
                ":" +
                new Date(Date.now()).getMinutes(),
              };
              await socket.emit("chatMessage", messageData);
            }
          }catch(error) {
            console.log("File upload Error: ")
            console.log(error)
            setFileErr(true)
          }
          setFileLoading(false)
          setSelectedFile(null)
        }
        if (currentMessage !== "") {
        const messageData = {
          room: room,
          author: username,
          message: currentMessage,
          time:
          new Date(Date.now()).getHours() +
          ":" +
          new Date(Date.now()).getMinutes(),
        };

        await socket.emit("chatMessage", messageData);
        //setMessageList((list) => [...list, messageData]);
        setCurrentMessage("");
        }
    };

    useEffect(() => {
      socket.on("roomUsers", (res) => {
        // console.log("users-gpk: ");
        // console.log(res)
        setUsers([res.users])
      })
    }, []);
    useEffect(() => {
       socket.on("message", (recData) => {
        // console.log("receive message")
        // console.log(recData)
        setMessageList((list) => [...list, recData]);
        //setUsers([recData.arr])
        //console.log(users)
        });
  }, []);
```

```
  const tempData = () => {
     return ([...Array(20)].map((e, i) => (
       <p className='px-4 py-1 text-black dark:text-white font-semibold'><span
className='px-2 text-green-500'>&#x25CF;</span>user1</p>
    )))
  }
    return (
      <>
        <div className='hidden sm:grid h-full w-full mx-4
        grid grid-cols-1 sm:grid-cols-8 gap-4'>
           <div style={{height:'80vh'}} className=' sm:col-span-3 bg-white
            rounded rounded-xl dark:bg-neutral-800 text-white dark:text-white'>
              <div className='p-1 bg-blue-500 flex justify-between'>
                 <h3 className='font-semibold'>UserName: {username}</h3>
                 <h3 className='bg-blue-500 font-semibold'>Room ID: {room}</h3>
              </div>
                 <h3 className='p-1 text-center text-black  dark:text-white uppercase
font-4xl bg-gray-100 dark:bg-neutral-700'>Active Users</h3>

              <div style={{height: '85%'}} className="users overflow-auto">
                    <div className=' '>
                       {/* { tempData()} */}
                    </div>

                    {
                    users.map((u) => {
                       return u.map((user, idx) => {
                          // console.log("u:")
                          // console.log(user);
                          return (
                             <p className='px-4 py-1 text-black dark:text-white font-
semibold' key={idx}><span className='px-2 text-green-
500'>&#x25CF;</span>{user?.username}</p>
                       )
                       })
                       }
                    )}
              </div>
           </div>
           {/* Chat body  style={{height:'72%'}}*/}
           <div style={{height: '80vh'}}  className=' sm:col-span-5 bg-white rounded
rounded-xl dark:bg-neutral-800'>
                 <div style={{height:'8%'}}  className='flex justify-between items-center
font-semibold text-center text-white bg-blue-500 p-0.5'>
                    <h3 className=''><span className='px-2 text-red-
500'>&#x25CF;</span>
                    Live Chat
```

```
                    </h3>
                    {copyBtn && <h3 className='bg-green-500 px-2
rounded'>copied</h3>}
                    {fileLoading && <h3 className='bg-green-500 px-2 rounded'>File
Uploading...</h3>}
                    {fileErr && <h3 className='bg-red-500 px-2 rounded'>File Upload
Error</h3>}
                    <a href='/' className='mr-1 hover:text-red-400  px-4 hover:bg-white
rounded rounded-xl bg-red-400 text-white'>
                      Leave Room
                    </a>
                  </div>
                <div style={{height:'84%'}} className=''>
                  <ScrollToBottom className="h-full overflow-auto">

                  {/* {tempData()} */}
                  {messageList.map((messageContent, idx) => {
                    if(username === messageContent.author ){
                      //you
                      return (
                        <div className='grid grid-cols-8  ' key={idx}>
                          <div className='col-start-3 col-span-6  '>
                            <div className='flex flex-col items-end'>
                            <div className=' bg-blue-500 dark:bg-neutral-700 m-2 p-2
                          text-white
                          rounded rounded-lg'>
                                <p>{messageContent.message}</p>
                                <div className='flex justify-end
                                text-slate-300 dark:text-gray-400'>
                                  <p id="time "
className=''>{messageContent.time}</p>
                                  <p id="author" className='px-
2'>{messageContent.author}</p>
                                  <CopyToClipboard text={messageContent.message}>
                                  <button onClick={() => {setCopyBtn(true)}}
className='hover:text-white'>
                                    <HiClipboardCopy size={20} />
                                  </button>
                                  </CopyToClipboard>
                                </div>
                              </div>
                              </div>
                          </div>
                        </div>

                      )
                    }else if ( messageContent.author === 'Bot'){
                      return (
```

```
                        <div className='grid grid-cols-8' key={idx}>
                           <div className='col-start-2 col-span-6
                           flex flex-col items-center'>
                              <div className=' bg-neutral-700 dark:bg-blue-500 m-2 p-2
                           text-white
                           rounded rounded-lg'>
                                 <p>{messageContent.message}</p>
                                 {/* <div className='flex justify-end
                                 dark:text-slate-300 text-gray-400'>
                                    <p id="time "
className=''>{messageContent.time}</p>
                                    <p id="author" className='px-
2'>{messageContent.author}</p>
                                 </div> */}
                              </div>
                           </div>
                        </div>
                     )
                  }
                  //others
                  return (
                     <div className='grid grid-cols-8  ' key={idx}>
                        <div className=' col-span-6 '>
                           <div className='flex flex-col items-start'>

                           <div className='bg-indigo-500 dark:bg-neutral-600 m-2 p-2
                           text-white
                           rounded rounded-lg'>
                              <p>{messageContent.message}</p>
                              <div className='flex
                              text-slate-300 dark:text-gray-400'>
                                 <p id="time "
className=''>{messageContent.time}</p>
                                 <p id="author" className='px-
2'>{messageContent.author}</p>
                                 <CopyToClipboard text={messageContent.message}>
                                 <button onClick={() => {setCopyBtn(true)}}
className='hover:text-white'>
                                    <HiClipboardCopy size={20} />
                                 </button>
                                 </CopyToClipboard>
                              </div>
                           </div>
                           </div>
                        </div>
                     </div>
                  )
```

```
                })}
                </ScrollToBottom>
              </div>
              <div className='h-fit flex justify-center items-center'>
              <input type="text" placeholder=" "
                 className=' py-1 w-full px-2 mx-1 text-xl text-black dark:text-white
                 bg-white dark:bg-neutral-800 border-black dark:border-white border-
opacity-30 hover:border-opacity-100 dark:border-opacity-30 dark:hover:border-
opacity-100 border rounded-lg border-opacity-50
                 outline-none focus:border-blue-500 dark:focus:border-blue-500
placeholder-gray-300 placeholder-opacity-0 transition duration-200'
                 value={currentMessage}
                 onChange={(e) => setCurrentMessage(e.target.value)}
                  onKeyPress={(event) => {
                    event.key === "Enter" && sendMessage();
                 }} />
                 <div className='bg-blue-500 rounded rounded-full p-2 text-white'>
                    <input type='file' id='inputFile' className='hidden'
                       onChange={(e) => setSelectedFile(e.target.files[0])}/>
                       <label htmlFor='inputFile'>
                          {selectedFile === null ? <IoMdAttach size={25}
className='cursor-pointer'/>
                               :<IoDocumentAttachSharp size={25} className='cursor-
pointer'/> }
                       </label>
                 </div>
                 <div className='mr-2 text-black dark:text-white hover:text-blue-600
dark:hover:text-blue-600' onClick={sendMessage}>
                    <IoSend size={20} className='-rotate-90'/>
                 </div>
              </div>
            </div>
          </div>




          {/* Mobile */}
          <div className='sm:hidden h-full w-full mx-4
          grid grid-rows-8 sm:grid-cols-8 gap-4'>
             <div style={{height:'25vh'}} className='row-span-2 sm:col-span-3 bg-white
              rounded rounded-xl dark:bg-neutral-800 text-white dark:text-white'>
                <div style={{height: '20%'}} className='p-1 bg-blue-500 flex justify-
between'>
                 <h3 className='font-semibold'>UserName: {username}</h3>
                 <h3 className='bg-blue-500 font-semibold'>Room ID: {room}</h3>
              </div>
                 <h3 style={{height: '20%'}} className='p-1 text-center text-black
```

```
dark:text-white uppercase font-4xl bg-gray-100 dark:bg-neutral-700'>Active
Users</h3>

            <div style={{height: '60%'}} className="users overflow-auto">
                <div className='Grid grid-cols-4'>
                  {/* { tempData()} */}
                </div>
                {
                users.map((u) => {
                  return u.map((user, idx) => {
                    // console.log("u:")
                    // console.log(user);
                    return (
                      <p className='px-4 py-1 text-black dark:text-white font-
semibold' key={idx} ><span className='px-2 text-green-
500'>&#x25CF;</span>{user?.username}</p>
                    )
                    })
                    }
                )}
              </div>
            </div>
            {/* {right} */}
            <div style={{height:'56vh'}}
            className='row-span-5 sm:col-span-5 bg-white rounded rounded-xl
dark:bg-neutral-800'>
                  <div style={{height:'8%'}}  className='flex justify-between items-
center font-semibold text-center text-white bg-blue-500 p-0.5'>
                    <h3 className=''><span className='px-2 text-red-
500'>&#x25CF;</span>
                    Live Chat
                    </h3>
                    {copyBtn && <h3 className='bg-green-500 px-2
rounded'>copied</h3>}
                    {fileLoading && <h3 className='bg-green-500 px-2 rounded'>File
Uploading...</h3>}
                    {fileErr && <h3 className='bg-red-500 px-2 rounded'>File Upload
Error</h3>}
                    <a href='/' className=' hover:text-red-400 mr-1  px-4 hover:bg-
white rounded rounded-xl bg-red-400 text-white'>
                      Leave Room
                    </a>
                  </div>

            <div style={{height:'78%'}} className=''>
              <ScrollToBottom className="h-full overflow-auto">

              {/* {tempData()} */}
```

```
                {messageList.map((messageContent, idx) => {
                  if(username === messageContent.author ){
                    //you
                    return (
                      <div className='grid grid-cols-8  ' key={idx}>
                        <div className='col-start-3 col-span-6
                        '>
                          <div className='flex flex-col items-end'>

                          <div className='bg-blue-500 dark:bg-neutral-700 m-2 p-2
                        text-white
                        rounded rounded-lg'>
                              <p>{messageContent.message}</p>
                              <div className='flex justify-end
                              text-slate-300 dark:text-gray-400'>
                                <p id="time "
className=''>{messageContent.time}</p>
                                <p id="author" className='px-
2'>{messageContent.author}</p>
                                <CopyToClipboard text={messageContent.message}>
                                <button onClick={() => {setCopyBtn(true)}}
className='hover:text-white'>
                                  <HiClipboardCopy size={20} />
                                </button>
                                </CopyToClipboard>
                              </div>
                            </div>
                            </div>
                        </div>
                      </div>

                    )
                  }else if ( messageContent.author === 'Bot'){
                    return (
                      <div className='grid grid-cols-8' key={idx}>
                        <div className='col-start-2 col-span-6
                        flex flex-col items-center'>
                          <div className=' bg-neutral-700 dark:bg-blue-500 m-2 p-2
                        text-white
                        rounded rounded-lg'>
                              <p>{messageContent.message}</p>
                              {/* <div className='flex justify-end
                              dark:text-slate-300 text-gray-400'>
                                <p id="time "
className=''>{messageContent.time}</p>
                                <p id="author" className='px-
2'>{messageContent.author}</p>
                              </div> */}
```

```
                            </div>
                        </div>
                    </div>
                )
            }
            //others
            return (
                <div className='grid grid-cols-8  ' key={idx}>
                    <div className=' col-span-6 '>
                        <div className='flex flex-col items-start'>
                            <div className='bg-indigo-500 dark:bg-neutral-600 m-2
p-2
                        text-white
                        rounded rounded-lg'>
                                <p>{messageContent.message}</p>
                                <div className='flex
                                text-slate-300 dark:text-gray-400'>
                                <p id="time "
className=''>{messageContent.time}</p>
                                <p id="author" className='px-
2'>{messageContent.author}</p>
                                <CopyToClipboard text={messageContent.message}>
                                <button onClick={() => {setCopyBtn(true)}}
className='hover:text-white'>
                                    <HiClipboardCopy size={20} />
                                </button>
                                </CopyToClipboard>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            )

        })}
        </ScrollToBottom>
    </div>
    <div style={{height:'16%'}} className='pb-2 flex justify-center items-
center'>
    <input type="text" placeholder=" "
        className=' py-1 w-full px-2 mx-1 text-xl text-black dark:text-white
        bg-white dark:bg-neutral-800 border-black dark:border-white border-
opacity-30 hover:border-opacity-100 dark:border-opacity-30 dark:hover:border-
opacity-100 border rounded-lg border-opacity-50
        outline-none focus:border-blue-500 dark:focus:border-blue-500
placeholder-gray-300 placeholder-opacity-0 transition duration-200'
        value={currentMessage}
        onChange={(e) => setCurrentMessage(e.target.value)}
```

```
                    onKeyPress={(event) => {
                      event.key === "Enter" && sendMessage();
                    }} />
                    <div className='bg-blue-500 rounded rounded-full p-2 text-white'>
                      <input type='file' id='inputFile' className='hidden'
                        onChange={(e) => setSelectedFile(e.target.files[0])}/>
                        <label htmlFor='inputFile'>
                          {selectedFile === null ? <IoMdAttach size={25}
classNameName='cursor-pointer'/>
                            :<IoDocumentAttachSharp size={25} className='cursor-
pointer'/> }
                        </label>
                    </div>
                    <div className='mr-2 text-black dark:text-white hover:text-blue-600
dark:hover:text-blue-600' onClick={sendMessage}>
                      <IoSend size={20} className='-rotate-90'/>
                    </div>
                  </div>

              </div>
            </div>
          </>
    )
}

export default Chat
```

5.2Results

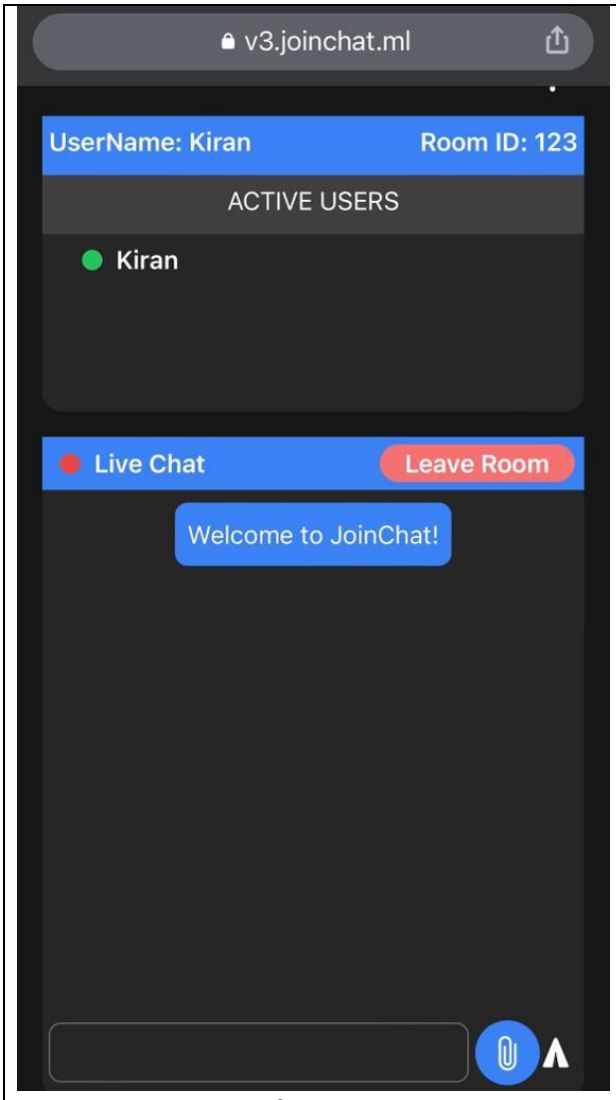

fig 5.1

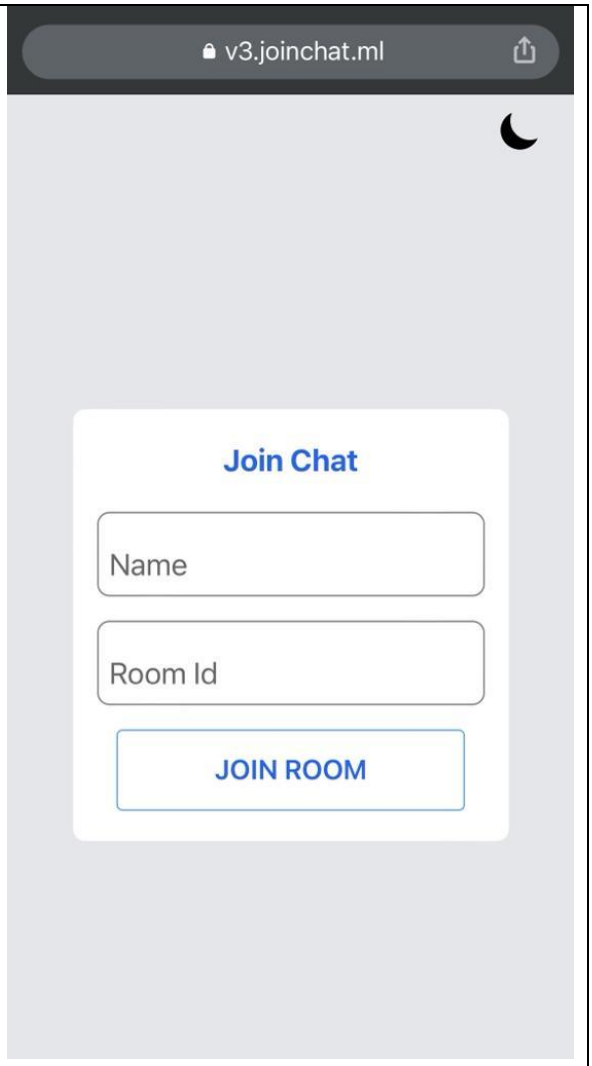

fig 5.2

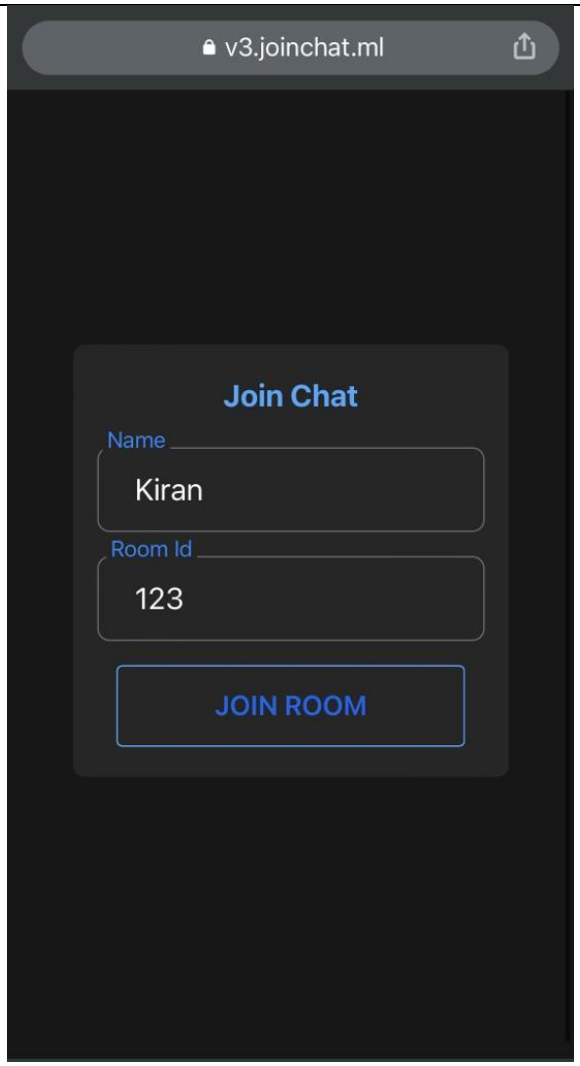
v3.joinchat.ml
Join Chat
Name
Kiran
Room Id
123
JOIN ROOM

fig 5.3

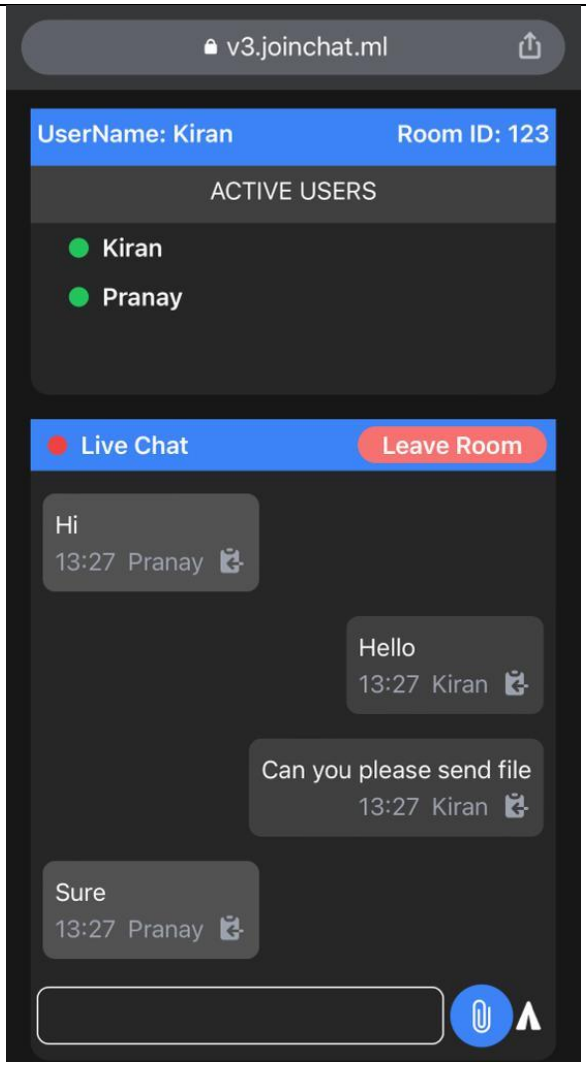
v3.joinchat.ml
UserName: Kiran
Room ID: 123
ACTIVE USERS
Kiran
Pranay
Live Chat
Leave Room
Hi
13:27 Pranay
Hello
13:27 Kiran
Can you please send file
13:27 Kiran
Sure
13:27 Pranay

fig 5.4

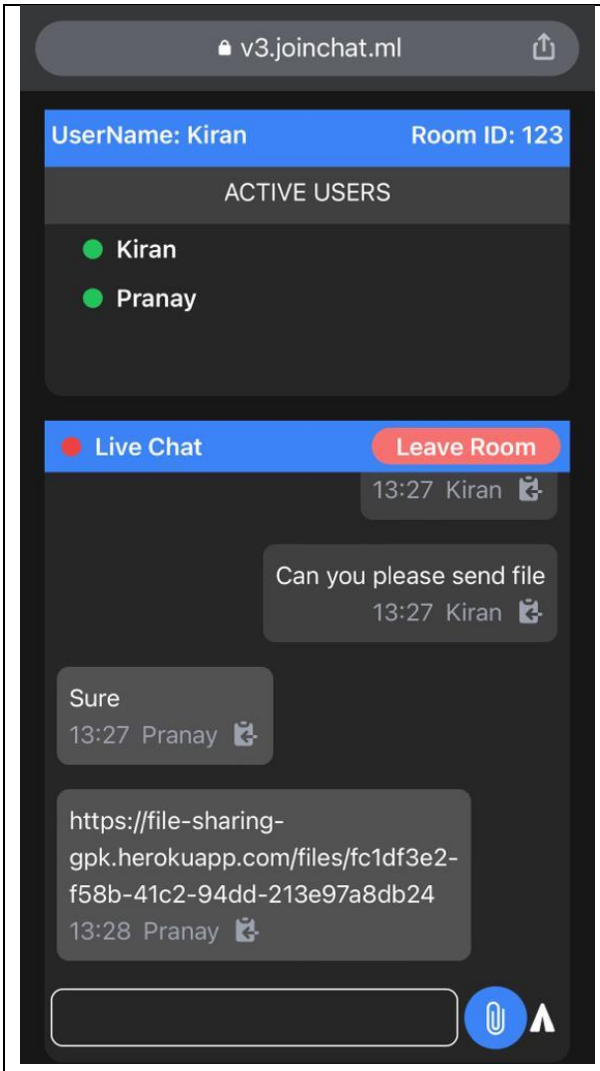
v3.joinchat.ml
UserName: Kiran
Room ID: 123
ACTIVE USERS
Kiran
Pranay
Live Chat
Leave Room
13:27 Kiran
Can you please send file
13:27 Kiran
Sure
13:27 Pranay
https://file-sharing-gpk.herokuapp.com/files/fc1df3e2-f58b-41c2-94dd-213e97a8db24
13:28 Pranay

fig 5.5

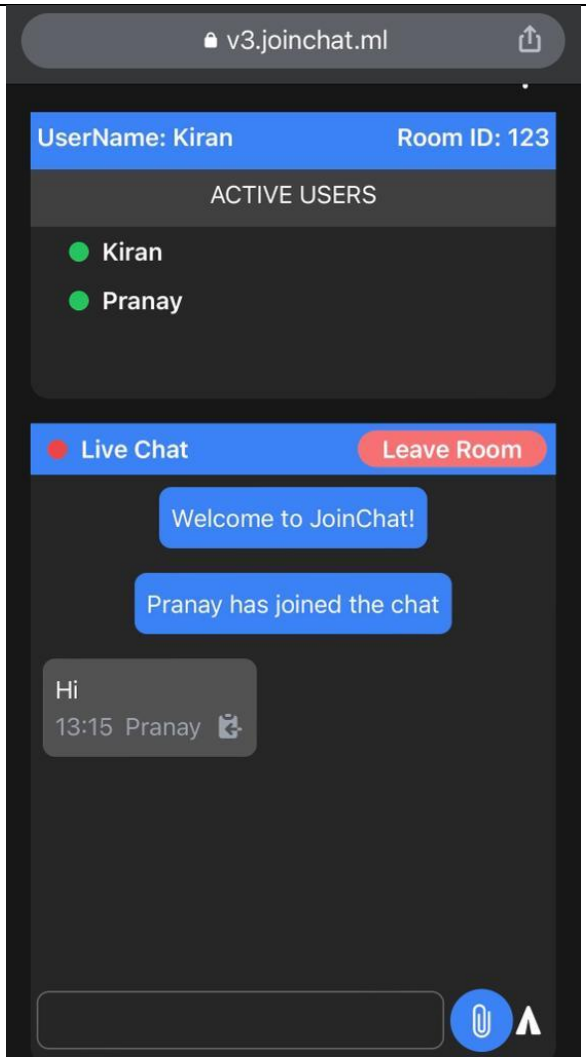
v3.joinchat.ml
UserName: Kiran
Room ID: 123
ACTIVE USERS
Kiran
Pranay
Live Chat
Leave Room
Welcome to JoinChat!
Pranay has joined the chat
Hi
13:15 Pranay

fig 5.6

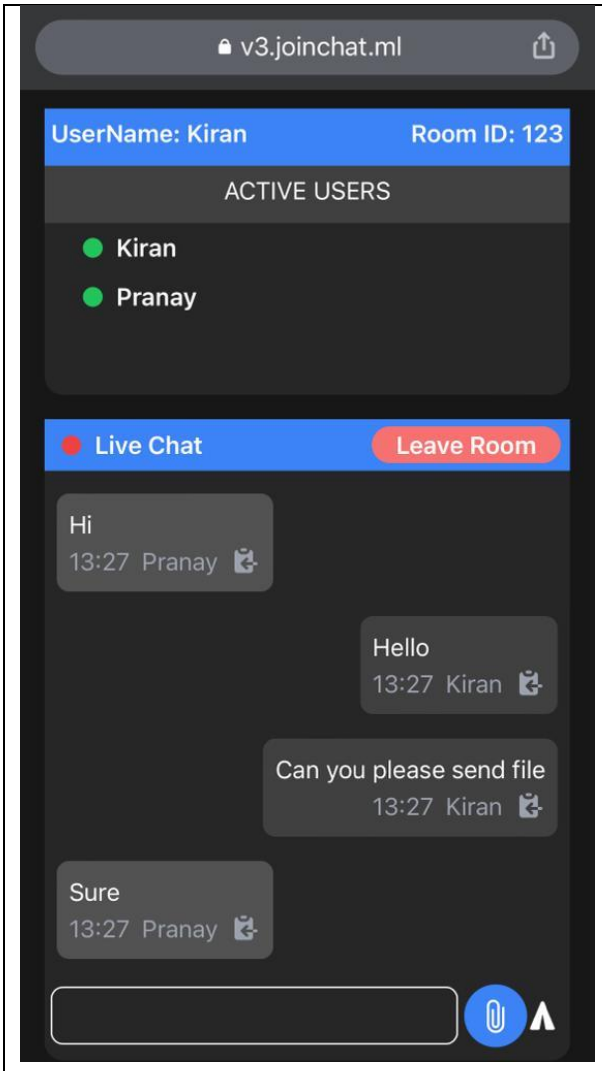
v3.joinchat.ml
UserName: Kiran
Room ID: 123
ACTIVE USERS
Kiran
Pranay
Live Chat
Leave Room
Hi
13:27 Pranay
Hello
13:27 Kiran
Can you please send file
13:27 Kiran
Sure
13:27 Pranay

fig 5.7

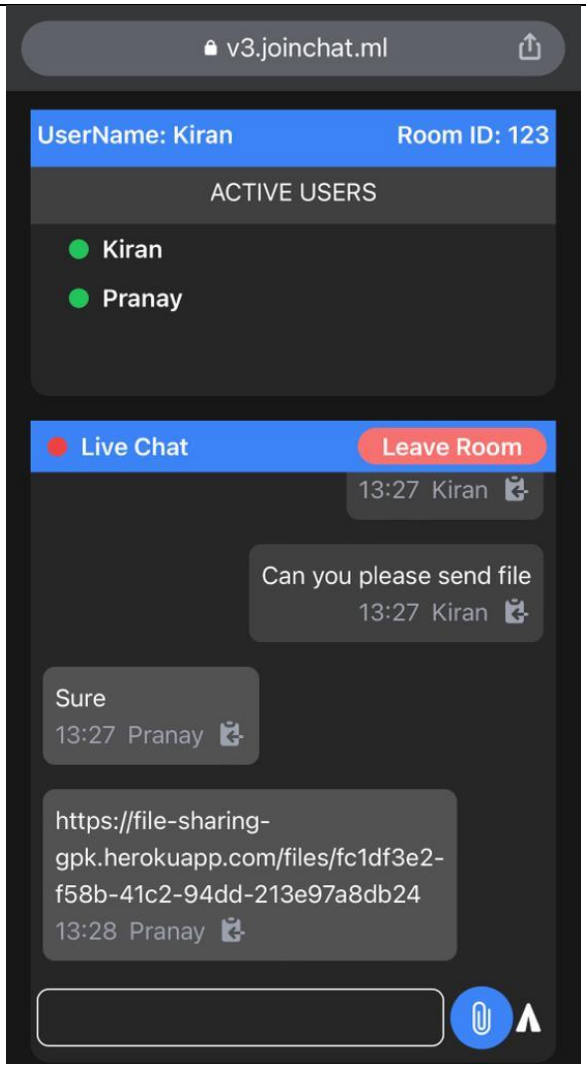
v3.joinchat.ml
UserName: Kiran
Room ID: 123
ACTIVE USERS
Kiran
Pranay
Live Chat
Leave Room
13:27 Kiran
Can you please send file
13:27 Kiran
Sure
13:27 Pranay
https://file-sharing-gpk.herokuapp.com/files/fc1df3e2-f58b-41c2-94dd-213e97a8db24
13:28 Pranay

fig 5.8

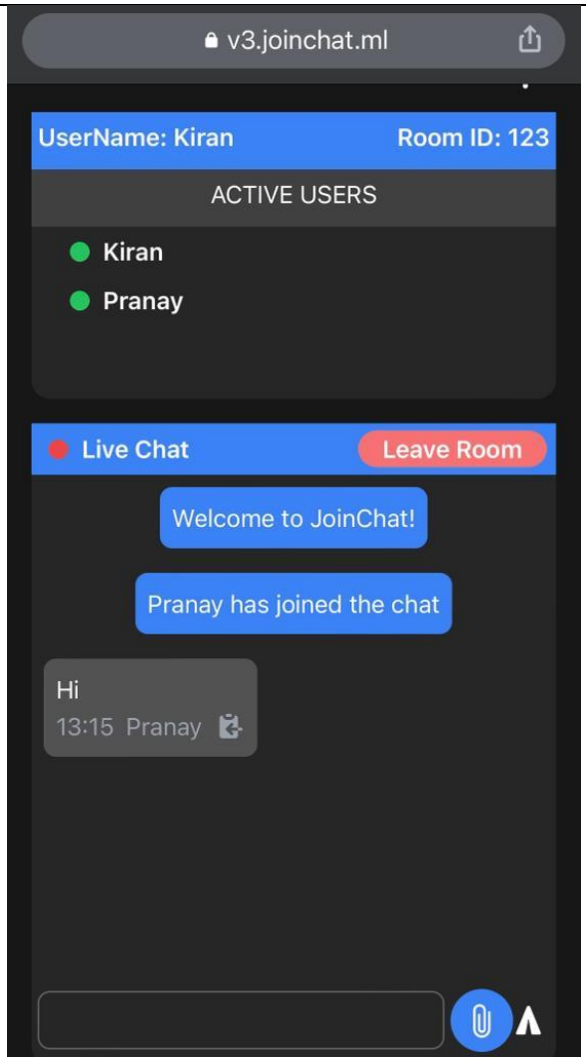

fig 5.9

Fig 5.10

6.CONCLUSION AND FUTURE WORK

- Our project solves the problem to always login into different social media apps and sites to communicate and share files over different devices with different people but by this all in one web based chat application where there is no need to sign in and anyone can use it to chat with people and share files from any device to any device.

- Nothing is stored on the server ,that means all the messages, files, users logs, nothing gets recorded and the chat system is encrypted.

- It is fast, reliable and saves time.

- This project interface is simple and any person will be able to use it at first glance.

6.1 FUTURE WORK

There is always room for improvements in any app. Right now, we are just dealing with text communication in public rooms with the feature to send any type of attachments . There are several chat apps which serve similar purposes as this project, but these apps were rather difficult to use and needed to create an account in it and add all our details in it and provide confusing interfaces. In future we may be extended to include features such as:

- Private rooms with key
- Voice Message
- Video Message
- Audio Call
- Video Call
- Group Call

7. REFERENCES

- https://socket.io/docs/v4/
- https://tailwindcss.com/docs/installation
- https://www.npmjs.com/package/multer
- https://www.investopedia.com/articles/investing/102615/story-instagram-rise-1-photo0sharing-app.asp
- https://www.docsity.com/en/online-chatting-system/4614473/
- https://www.slideshare.net/adkpcte/report-on-online-chatting
- http://indusedu.org/pdfs/IJREISS/IJREISS_3661_55346.pdf