



# **ECHO-MESSAGER**



## **MINI PROJECT REPORT**

*Submitted by*

**S. KALEESWARAN (REG.NO: 95192101041)**

**V. KIRTHICK SAKTHI (REG.NO: 95192101046)**

**A. JOSHIN SINGH (REG.NO: 95192101501)**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**P.S.R. ENGINEERING COLLEGE, SIVAKASI -626140**

(An Autonomous Institution, Affiliated to Anna University, Chennai)

**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY 2024**

**BONAFIDE CERTIFICATE**

Certified that this project report titled “**ECHO MESSENGER**” is the bonafide work of **S.KALEESWARAN (REG.NO: 95192101041)** **V.KIRTHICK SAKTHI (REG.NO: 95192101046)** **A. JOSHIN SINGH (REG.NO: 95192101501)** who carried out the project work under my supervision.

**SIGNATURE**

**Mrs.R.Anitha M.E.,**  
**SUPERVISOR,**  
Assistance Professor  
Department of Computer Science and  
Engineering,  
P.S.R. Engineering College,  
Sivakasi - 626140.

**SIGNATURE**

**Dr. A. Ramathilagam M.E., Ph.D.,**  
**HEAD OF THE DEPARTMENT**  
Professor & Head,  
Department of Computer Science and  
Engineering,  
P.S.R. Engineering College,  
Sivakasi - 626140.

**Submitted for the MINI PROJECT Viva-Voce Examination to be held on**

.....

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We take this opportunity to all those who helped towards successful completion of this project. At the very outset we thank the almighty for his profuse blessings showered on us.

We thank our beloved parents whose encouragement and support help us to complete our project successfully.

We unreservedly express our indebtedness to our Managing Trustee and Correspondent **Thiru.R. SOLAISAMY** and Director **Er. S. VIGNESWARI ARUNKUMAR** for providing the needed facilities.

It is our privilege to convey our sincere thanks to our respected honorable Principa **Dr.J.S. SENTHIL KUMAAR M.E., Ph.D.**, for giving us permission to do this project in our institution.

We wish to extend and express our sincere thanks to our adored Head of the Department **Dr.A. RAMATHILAGAM, M.E., Ph.D., Professor** for her excellent guidance and motivation to complete our project proficiently.

We also wish to express our hearty thanks to our Project Guide **Mrs.R.ANITHA M.E.**, for his / her excellent guidance and constant encouragement during this project work.

We also wish to express our sincere thanks to our Project Coordinator **Dr. S. SINGARAVELAN, M.E., Ph.D., Professor,** and **Mr. P. RAMA SUBRAMANIAN, M.E., Assistant Professor** for their continuous guidance and suggestions, without whom this project would not have become a reality.

We also bound to thank to all the Teaching and Non-Teaching staff members of our Computer Science and Engineering Department whose supported us with their wishes and prayers that bring success and completion of our project.

## **ABSTRACT**

Echo Messenger is a state-of-the-art chat application prioritizing user privacy, seamless communication, and optimized performance. With advanced encryption, conversations are securely protected from unauthorized access, ensuring peace of mind. Its intuitive interface simplifies messaging, media sharing, and group chats for effortless communication. Optimized algorithms minimize data usage while maintaining fast performance, guaranteeing uninterrupted chats. Compatible across devices, users can stay connected seamlessly on smartphones, tablets, or computers. Offering customizable settings and scheduling options, Echo Messenger caters to personal and professional communication needs. Join millions of satisfied users and experience the simplicity, security, and reliability of Echo Messenger today. "Echo Messenger redefines secure communication with its innovative blend of end-to-end encryption and user-friendly interface. By prioritizing privacy without sacrificing usability, it offers peace of mind to users concerned about data security. Its advanced encryption algorithms ensure that messages remain confidential, shielding them from unauthorized access. With features like customizable notifications and Echo Messenger delivers a seamless communication experience across devices. Whether on a smartphone, tablet, or desktop, users can trust Echo Messenger to protect their sensitive information while maintaining convenience. Join the revolution in secure messaging with Echo Messenger – where privacy meets simplicity.

# TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	4
	LIST OF FIGURES	6
	LIST OF ABBREVIATIONS	7
1	INTRODUCTION	8
2	LITERATURE SURVEY	10
3	SYSTEM ARCHITECTURE	13
3.1	EXISTING SYSTEM	13
3.2	PROPOSED SYSTEM	14
4	SYSTEM SPECIFICATION	18
4.1	HARDWARE REQUIREMENTS	18
4.2	SOFTWARE REQUIREMENTS	18
5	SYSTEM IMPLEMENTATION	19
5.1	MODULUS DESCRIPTION	19
5.1	USER AUTHENTICATION	19
5.2	REGISTRATION	19
5.3	LOGIN INTERFACE	20
6	RESULTS	21
7	CONCLUSION AND FUTURE ENHANCEMENT	25
8	APPENDIX	26
9	REFERENCE	49

## LIST OF FIGURES

<b>S.NO</b>	<b>FIGURE NO</b>	<b>DESCRIPTION OF FIGURE</b>	<b>PAGE NO</b>
1.	3.3.1	System Architecture	15
2.	3.3.2	Use Case Diagram	16
3.	3.3.3	Class Diagram	16
4.	3.3.4	Flow Diagram	17
5.	3.3.5	Sequence Diagram	17

## LIST OF ABBREVIATIONS

<b>SYMBOLS</b>	<b>ABBREVIATIONS</b>
MERN	MongoDB, Express.js, React.js, Node.js
API	Application Programming Interface
JSON	JSON Web Token (for authentication)
REST	Representational State Transfer
CSS	Cascading Style Sheets
CRUD	Create, Read, Update, Delete
HTML	Hyper Text Markup Language
DOM	Document Object Model
DB	Database
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment

# **1.INTRODUCTION**

The chatting application has huge impact on day to day life. There are numerous chatting application available in this world. Each application has different additional features varying from other applications. These application organizations compete with each other and add some competing features during each release. They have reached people much and have an impact on people's life. People find a better application from an available internet application which they feel much reliable and secure. The above mentioned applications have billion users all over the world. Those companies are one of the top companies in the world. They have higher revenue per year and have many employees for their organizations developing additional features to compete with other organizations during their each release. These applications have different features and follows different ways to ensure security of their user data. Today a data theft is the major crime and most people are involved in it. There are many cases being filed these days about personal data loss. So the organizations have to ensure the security from data loss by the third party data crisis. The basic chatting system should involve both sending and receiving processes simultaneously. In this application both sending and receiving messages simultaneously happens through MERN concept.

Introducing Echo Messenger, a cutting-edge chat application designed to revolutionize your communication experience. Echo Messenger goes beyond traditional messaging platforms, offering a seamless blend of connectivity, expression, and security. Stay connected with friends, family, and colleagues effortlessly, regardless of distance, thanks to Echo Messenger's robust network. Express yourself in vibrant ways with a plethora of options, from text to emojis, stickers. Your privacy and security are paramount, with end-to-end encryption and advanced security features ensuring your conversations remain confidential. Echo Messenger synchronizes seamlessly across all



your devices, allowing you to access your messages anytime, anywhere. Customize your chat experience to your liking with personalized themes and notification settings. Join the Echo Messenger community today and discover a new era of communication, where connections are vibrant, expressions are limitless, and conversations echo with life.

Echo Messenger is not just a chat application, it's a dynamic ecosystem where conversations thrive and connections deepen. With its intuitive interface and user-friendly features, Echo Messenger makes staying in touch a breeze. Whether you're catching up with old friends, collaborating with colleagues, or sharing moments with loved ones, Echo Messenger offers a versatile platform for every interaction. One of Echo Messenger's standout features is its extensive array of expressive tools. Echo Messenger provides the perfect tools to express yourself authentically.

Security is paramount in today's digital landscape, and Echo Messenger takes it seriously. With end-to-end encryption safeguarding your messages and stringent security measures protecting your data, you can chat with confidence, knowing your privacy is prioritized. Echo Messenger seamlessly integrates into your daily life, with synchronized conversations across all your devices. Whether you're on your smartphone, tablet, or desktop computer, your messages are always accessible, ensuring you never miss an important moment or conversation.

Personalization is key, and Echo Messenger empowers you to tailor your chat experience to your preferences. Choose from a variety of themes, customize notification settings, and organize your chats to suit your style. With Echo Messenger, you're in control of your communication journey. Join the Echo Messenger community today and experience a new standard of connectivity, expression, and security. Elevate your conversations, deepen your connections, and let your voice echo loud and clear with Echo Messenger.

## **1. LITERATURE SURVEY**

**1. In 2020, "Designing and deploying a real-time web-based chat server"** by Diotra Henriyan, Devie Pratama Subiyanti, Rizki Fauzian and published by the International Conference on Engineering and Technology (ICSET).

### **Methodology:**

In their paper, Diotra Henriyan, Devie Pratama Subiyanti, and Rizki Fauzian outline a methodology for designing and deploying a real-time web-based chat server. They begin by defining the project requirements and objectives, followed by researching existing chat server architectures and technologies. The authors then propose their design solution, outlining the system's architecture, components, and functionalities. Next, they implement the proposed design using appropriate programming languages and frameworks, ensuring scalability, reliability, and security. Testing is conducted to validate the system's performance and functionality, with any identified issues addressed through iterative improvements. Finally, the deployed chat server is evaluated in real-world scenarios to assess its usability, performance, and user satisfaction, providing valuable insights for future enhancements.

**2. In 2022, Jhalak Mittal, "Arushi Garg, Shivani Sharma" and "Online Chat Request"** was published by the International Journal of Research in Engineering, IT and Social Sciences, ISSN 2250-0588.

### **Methodology:**

The methodology employed in developing the online chat request system involved a systematic approach encompassing several key stages. Firstly, an extensive review of existing literature and chat systems was conducted to gain insights into relevant technologies and methodologies. Subsequently, thorough requirements analysis was undertaken through stakeholder consultations and surveys to ascertain both functional and

non-functional requirements. Based on these requirements, a robust system architecture was meticulously designed, considering factors such as scalability, reliability, and user experience. The implementation phase involved the translation of the designed architecture into a tangible system using appropriate programming languages and frameworks. Throughout the development process, rigorous testing procedures were executed to validate the system's functionality, performance, and reliability. Upon successful testing, the system was deployed to a production environment, with careful planning to ensure a smooth transition. Post-deployment, the system underwent evaluation to assess its effectiveness and usability in real-world scenarios, with feedback informing potential improvements. Comprehensive documentation was prepared to aid in system maintenance and future development efforts. Finally, mechanisms for continuous improvement were established to support ongoing enhancements and updates based on evolving user needs and technological advancements.

**3. In 2023, Jhalak Mittal, Arushi Garg, and Shivani Sharma titled "Enhancing User Engagement in Online Chat Systems" published in the International Journal of Advanced Computer Science.**

### **Methodology:**

The study employed a systematic approach starting with a literature review on user engagement in online chat systems. Stakeholder consultations and user surveys were conducted to gather requirements and preferences. Based on findings, engagement strategies were conceptualized and integrated into a chat system prototype. Rigorous testing assessed strategy effectiveness through user testing and analysis. Feedback was iteratively used to refine strategies for optimization. Final strategies underwent validation to ensure applicability across demographics. Comprehensive documentation summarized methodology, implementation, and results. Overall, a structured and systematic approach was followed to enhance user engagement in online chat systems.

**User Interface and Experience:** A good chat application should have an intuitive user interface that is easy to use and navigate. It should also provide a seamless user experience across different devices and platforms (Kumar & Pal, 2021).

**Real-time Messaging:** Real-time messaging is a critical feature of chat applications. The application should support instant messaging with minimal latency. Moreover, it should handle message delivery reliably, even under poor network conditions (Huang et al., 2020).

**Security and Privacy:** Chat applications should ensure the security and privacy of user data. They should use end-to-end encryption to protect messages from eavesdropping. Moreover, they should comply with data protection regulations and provide transparency on how user data is used and stored (Rahman et al., 2021).

**Group Chats and Collaboration:** Group chats and collaboration features are essential in chat applications, especially in a professional setting. The application should support creating and managing groups, sharing files and documents, and conducting audio and video conferences (Jensen et al., 2020).

**Performance Measurement:** The performance of chat applications can be measured using various parameters such as response time, throughput, and availability. It is essential to monitor these parameters continuously to ensure the application performs well under different loads and conditions (Kumar & Pal, 2021).

### **3. SYSTEM ARCHITECTURE**

#### **3.1 EXISTING SYSTEM**

The existing system of a chat application comprises several interconnected components, each serving a vital role in delivering a seamless user experience. When the existing system was studied, it was found having some problems, existing system was very time consuming and was not very efficient. The drawback of the existing system has resulted in to the development of new system, which is very user friendly and effective. Existing system was also very low in performance. While developing the new system all requirements of the end user was taken into consideration. These have been maximum efforts towards overcoming the drawbacks of the existing system, while the new system was designed & developed.

A chat application typically consists of several interconnected components that work together to enable seamless communication between users. At its core, the system revolves around messaging functionality, allowing users to send and receive text-based messages in real-time or near-real-time.

Underlying infrastructure, including servers and databases, handle the storage and transmission of messages, user data, and other information critical to the functioning of the application. This infrastructure must be robust and scalable to accommodate growing user bases and fluctuating demands.

Security features, such as end-to-end encryption, help safeguard user privacy and protect sensitive information from unauthorized access or interception. Encryption ensures that messages remain confidential and secure, even if they're intercepted during transmission

or stored on servers.

In addition to basic messaging functionality, many chat applications offer additional features like file sharing, emojis, and integrations with other services or platforms. These features enhance the user experience and make the application more versatile and appealing to a broader audience.

## **DISADVANTAGES:**

1. Privacy vulnerabilities: Risks of data breaches and unauthorized access to conversations.
2. Addiction potential: Excessive use of chat apps may lead to dependency and reduced productivity.
3. Information overload: Constant notifications and messages can overwhelm users, leading to stress and cognitive fatigue.

## **3.2 PROPOSED SYSTEM**

The chat application will be developed using the MERN stack, comprising MongoDB for database management, Express.js for handling backend APIs, React.js for building the frontend user interface, and Node.js for server-side runtime. Authentication is a crucial component of any chat application, ensuring that only authorized users can access the platform and engage in conversations. This involves login processes, account creation, password management, and sometimes additional security measures like two-factor authentication. User interface elements, such as contact lists, chat windows, and message threads, form the visual backbone of the application, providing users with intuitive ways to navigate and interact with their conversations. Additionally, features like chat groups or channels allow users to communicate with multiple people simultaneously, fostering collaboration and community-building. Security features, such as end-to-end encryption, help safeguard user privacy and protect sensitive information from unauthorized access or

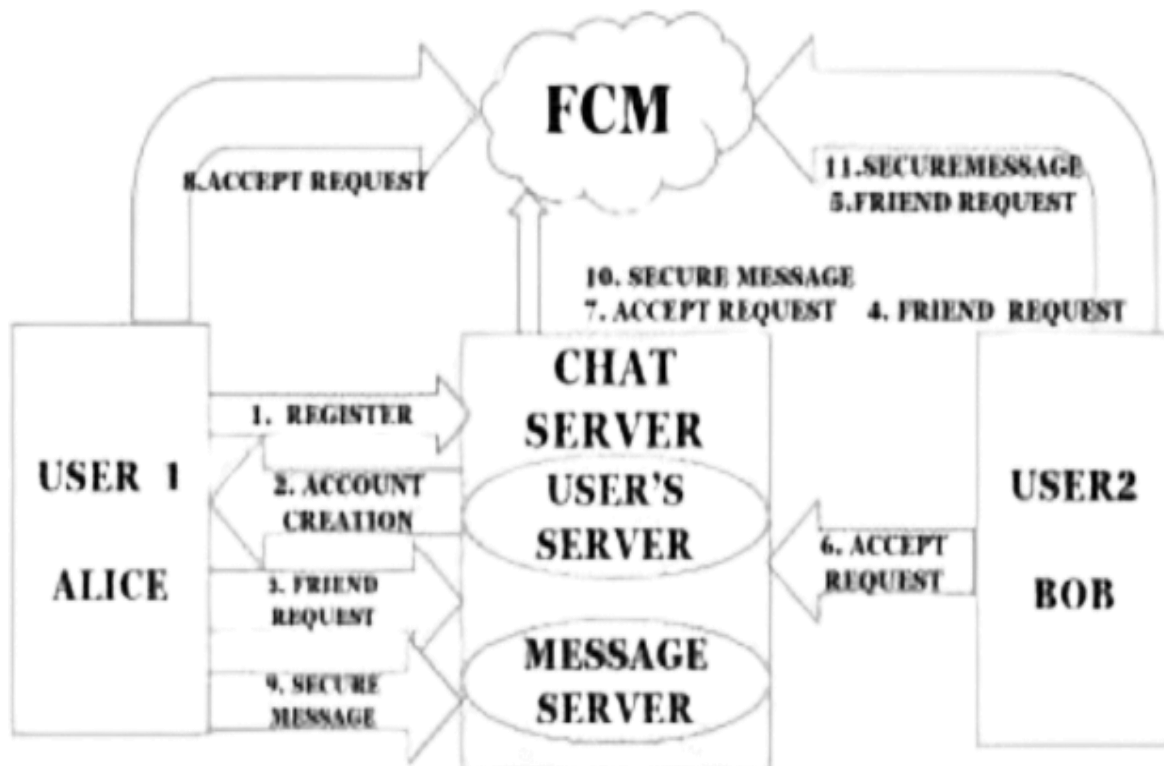
interception. Encryption ensures that messages remain confidential and secure, even if they're intercepted during transmission or stored on servers.

### Advantages:

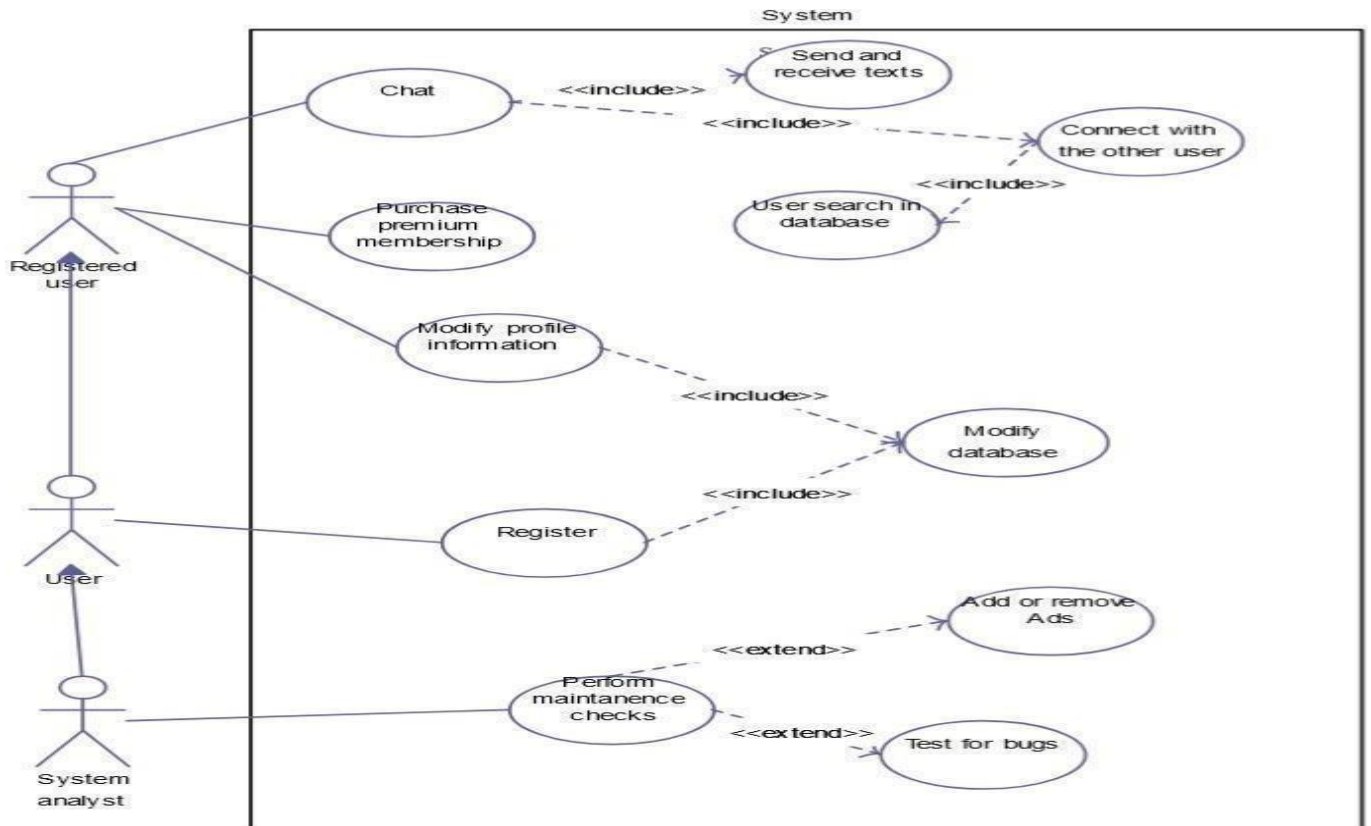
- User can register and login.
- You can search for others who are using this web app.
- User can chat and send emojis to others.
- User can create groups.
- You can add or remove members from the group.
- All your data will be stored on the database so that they can be restored once you login again.

## 3.3 SYSTEM DIAGRAMS

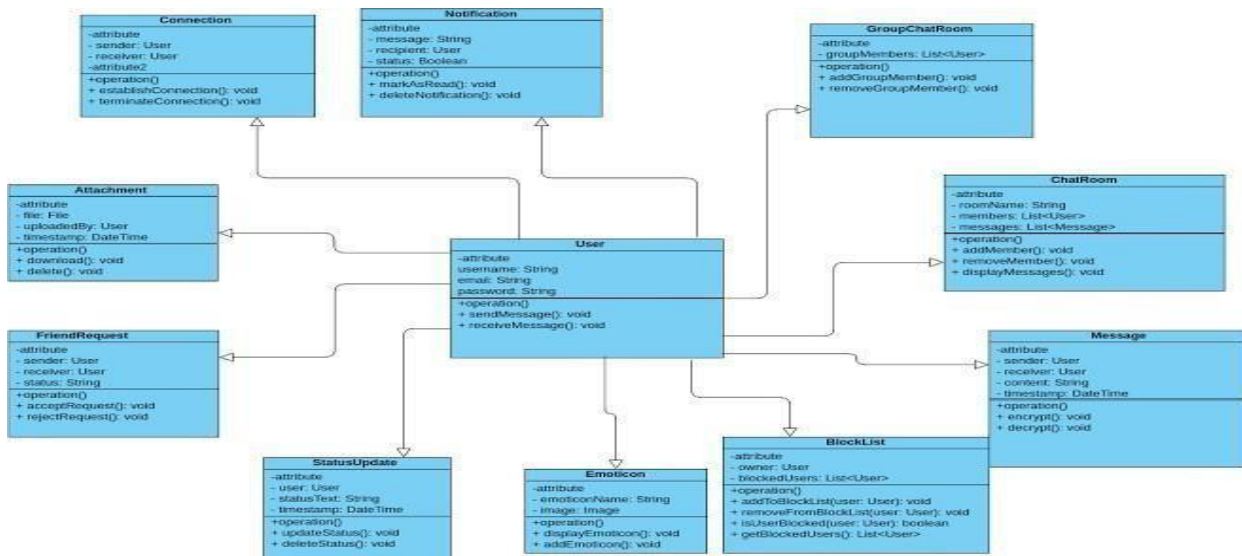
### 3.3.1 SYSTEM ARCHITECTURE



### 3.3.2 USE CASE DIAGRAM

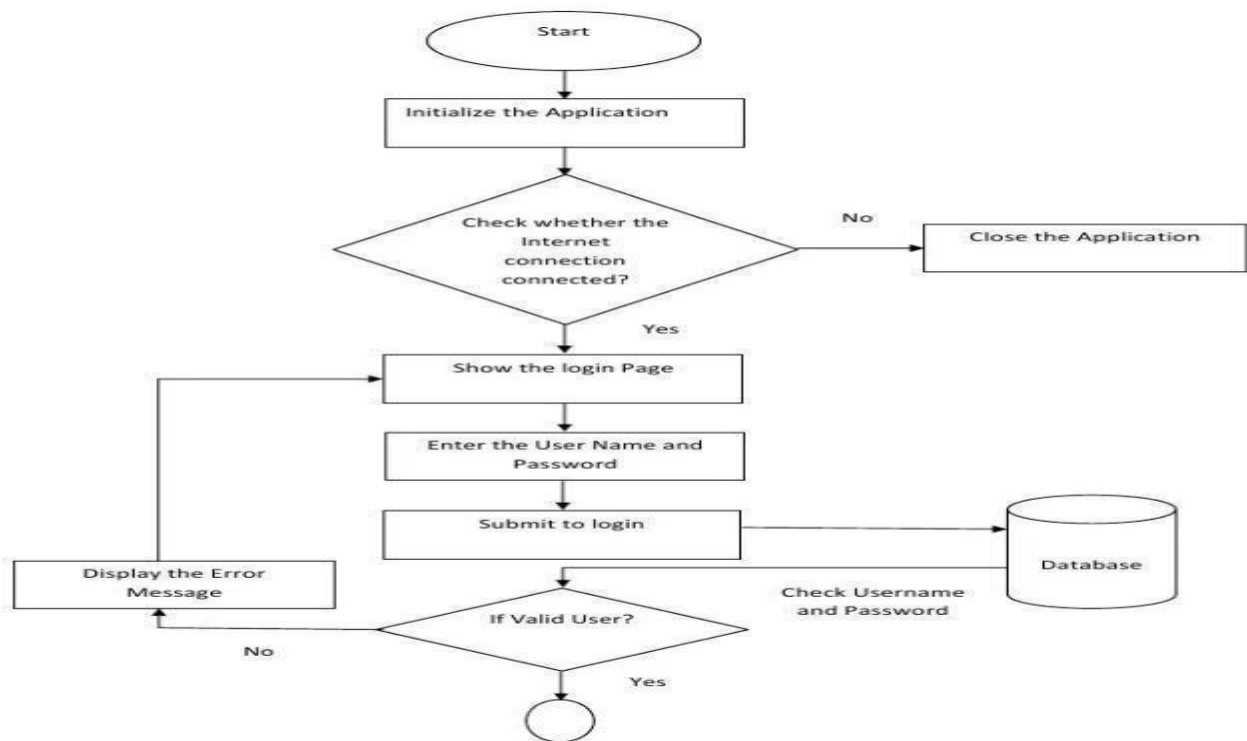


### 3.3.3 CLASS DIAGRAM

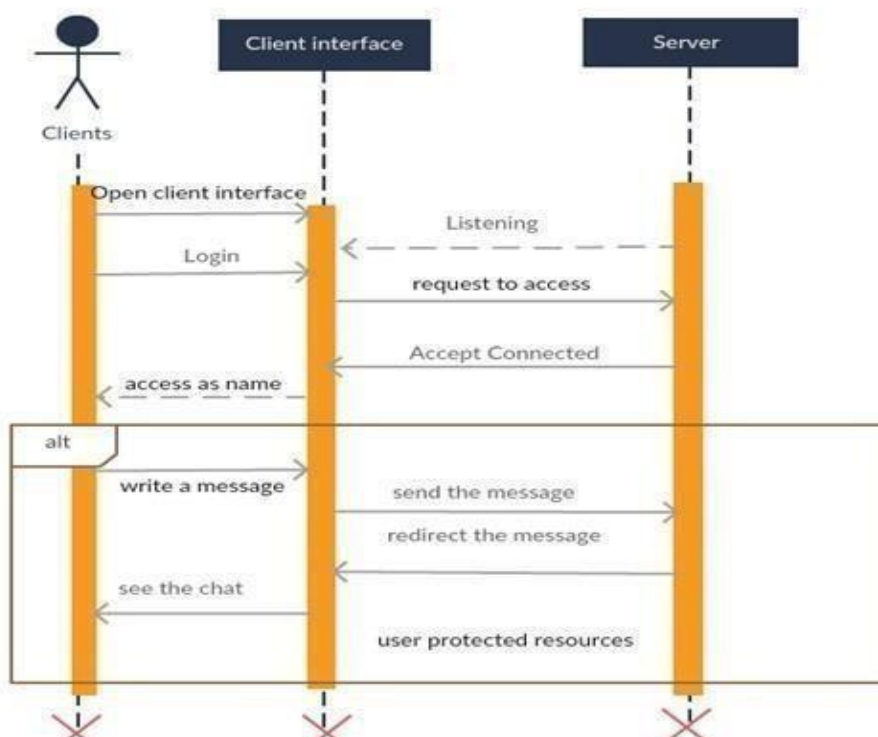




### 3.3.4 FLOW DIAGRAM



### 3.3.5 SEQUENCE DIAGRAM



## **4. SYSTEM SPECIFICATION**

### **4.1 HARDWARE REQUIREMENT**

- System : Pentium IV 2.4GHz
- Hard Disk : 200GB
- Mouse : Logitech
- Keyboard : 110keys enhanced
- Ram : 4GB

### **4.2 SOFTWARE REQUIREMENT**

Software's can be defined as programs which run on our computer .it act as petrol in the vehicle. It provides the relationship between the human and a computer. It is very important to run software to function the computer. Various software's are needed in this project for its development.

- Operating system : Windows 7,10
- Others : Visual Studio, nodejs
- Data base : mongo db

## **5. SYSTEM IMPEMETATION**

### **5.1 MODULUS**

1. User Authentication
2. Registration
3. Login Interface

### **5.2 MODULUS DESCRIPTION**

#### **User Authentication:**

Handles user authentication and authorization processes, ensuring secure access to the chat website. This module verifies the identity of users and grants access only to authorized individuals. It employs encryption and secure protocols to protect sensitive user data during authentication. User authentication is a crucial aspect of the website's security framework, preventing unauthorized access and protecting user privacy. By implementing robust authentication mechanisms, the module enhances the overall security posture of the chat website. Additionally, it supports multi-factor authentication for added security layers, further bolstering user authentication processes.

#### **Registration:**

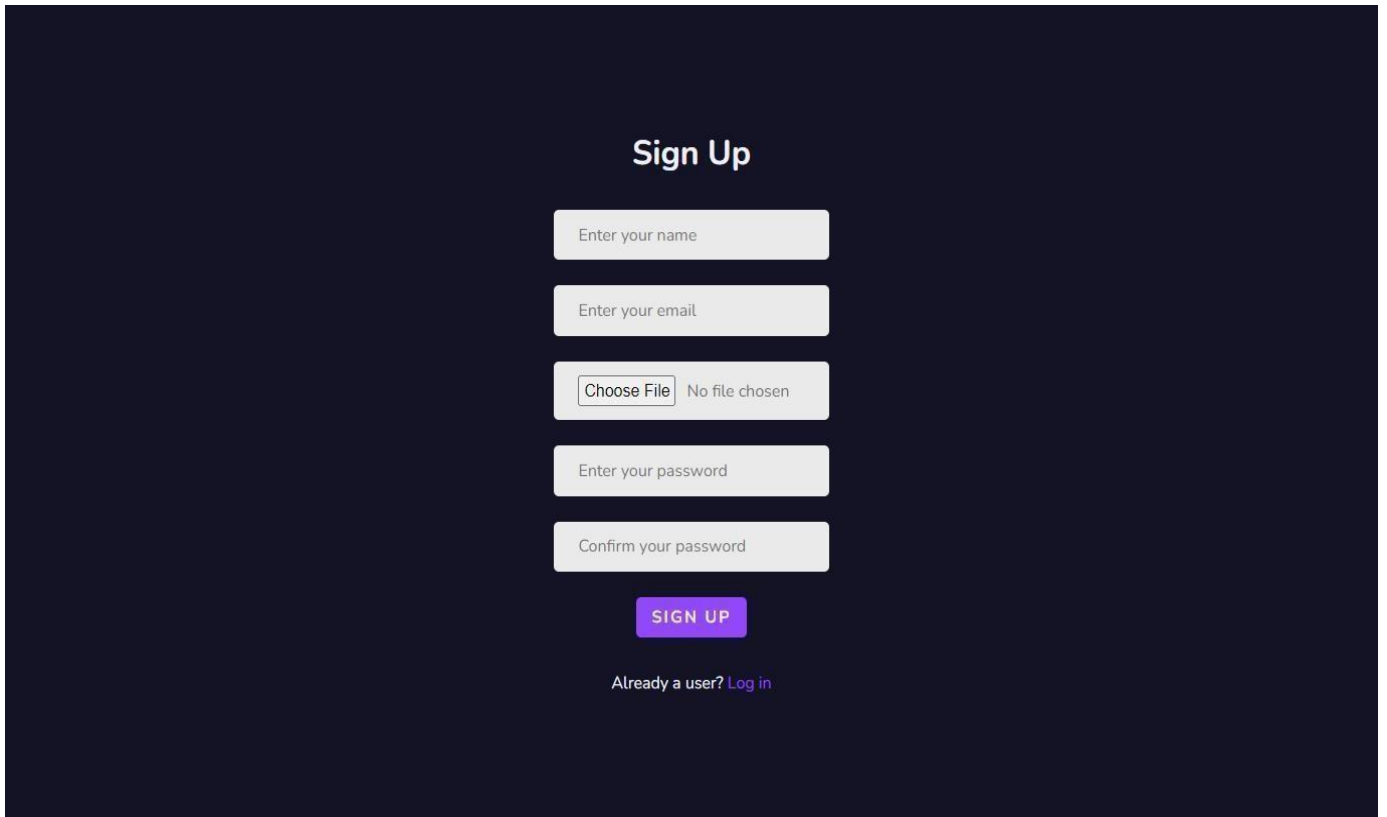
Allows users to create new accounts by providing required information such as username, email, and password. This module facilitates the onboarding process for new users, enabling them to register and gain access to the chat website's features. During registration, users input personal details and choose credentials for their accounts. The module validates user input to ensure data accuracy and completeness, enhancing the registration experience. It also implements measures to prevent fraudulent or malicious registrations, such as email confirmation. Upon successful registration, users receive

confirmation messages and can proceed to log in to their accounts. The registration module contributes to user acquisition and engagement, fostering growth and community building on the chat website.

### **Login Interface:**

Provides a user-friendly interface for users to enter their credentials and log in to their accounts securely. This module offers an intuitive login experience, featuring input fields for username/email and password. Users can access the login interface from the website's homepage or designated login page. The interface incorporates responsive design principles, ensuring compatibility across devices and screen sizes. It also includes error handling mechanisms to alert users about incorrect credentials or login failures. By prioritizing usability and accessibility, the login interface enhances user satisfaction and retention. Additionally, the module may offer optional features such as social login integration for convenience. Overall, the login interface serves as the primary gateway for users to access the chat website's features securely.

## 6. RESULT



The image shows a 'Sign Up' form on a dark blue background. The form is centered and consists of several input fields and a button. The title 'Sign Up' is at the top in white. Below it are five light gray input fields: 'Enter your name', 'Enter your email', a file upload field with a 'Choose File' button and 'No file chosen' text, 'Enter your password', and 'Confirm your password'. A purple 'SIGN UP' button is below the fields. At the bottom, there is a link 'Already a user? Log in'.

Sign Up

Enter your name

Enter your email

Choose File No file chosen

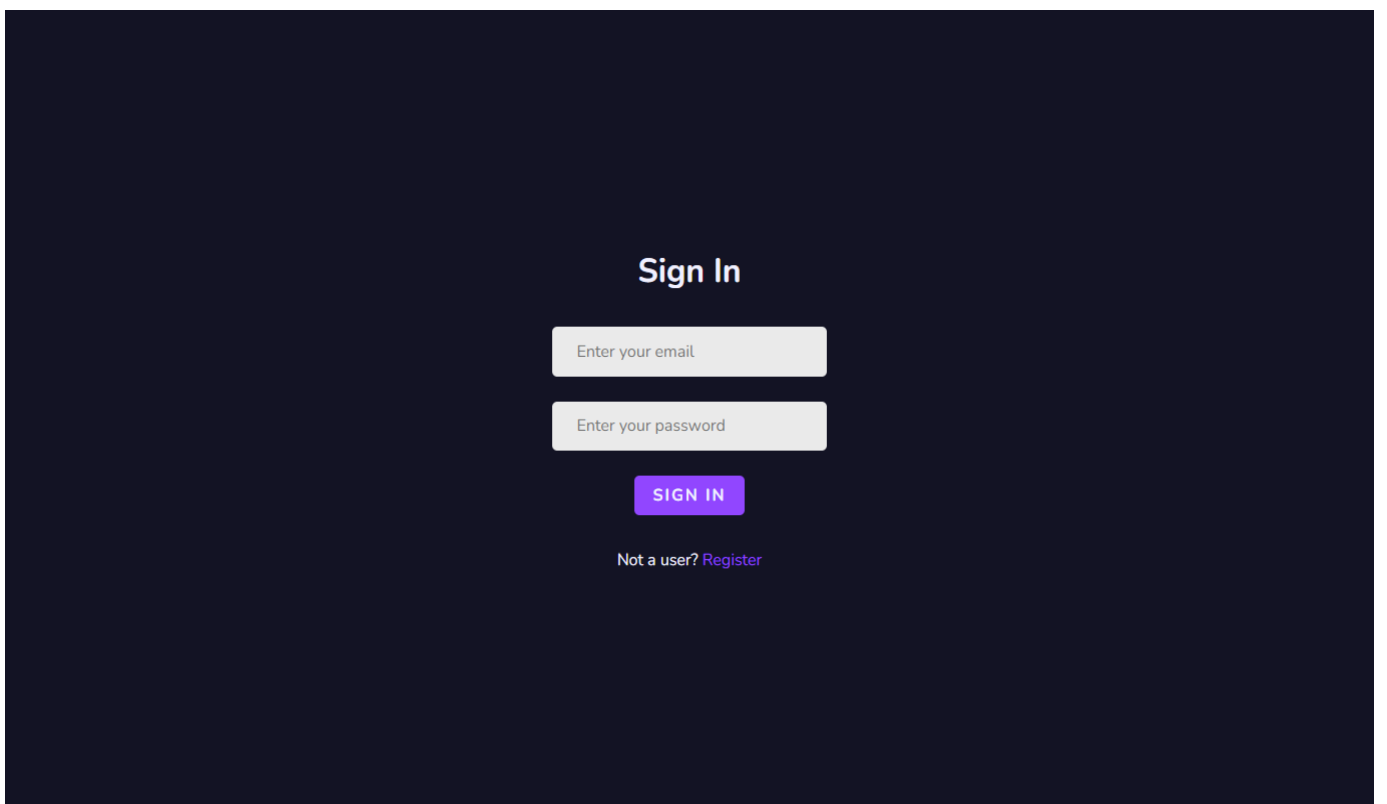
Enter your password

Confirm your password

SIGN UP

Already a user? [Log in](#)

Figure 1



The image shows a 'Sign In' form on a dark blue background. The form is centered and consists of two input fields and a button. The title 'Sign In' is at the top in white. Below it are two light gray input fields: 'Enter your email' and 'Enter your password'. A purple 'SIGN IN' button is below the fields. At the bottom, there is a link 'Not a user? Register'.

Sign In

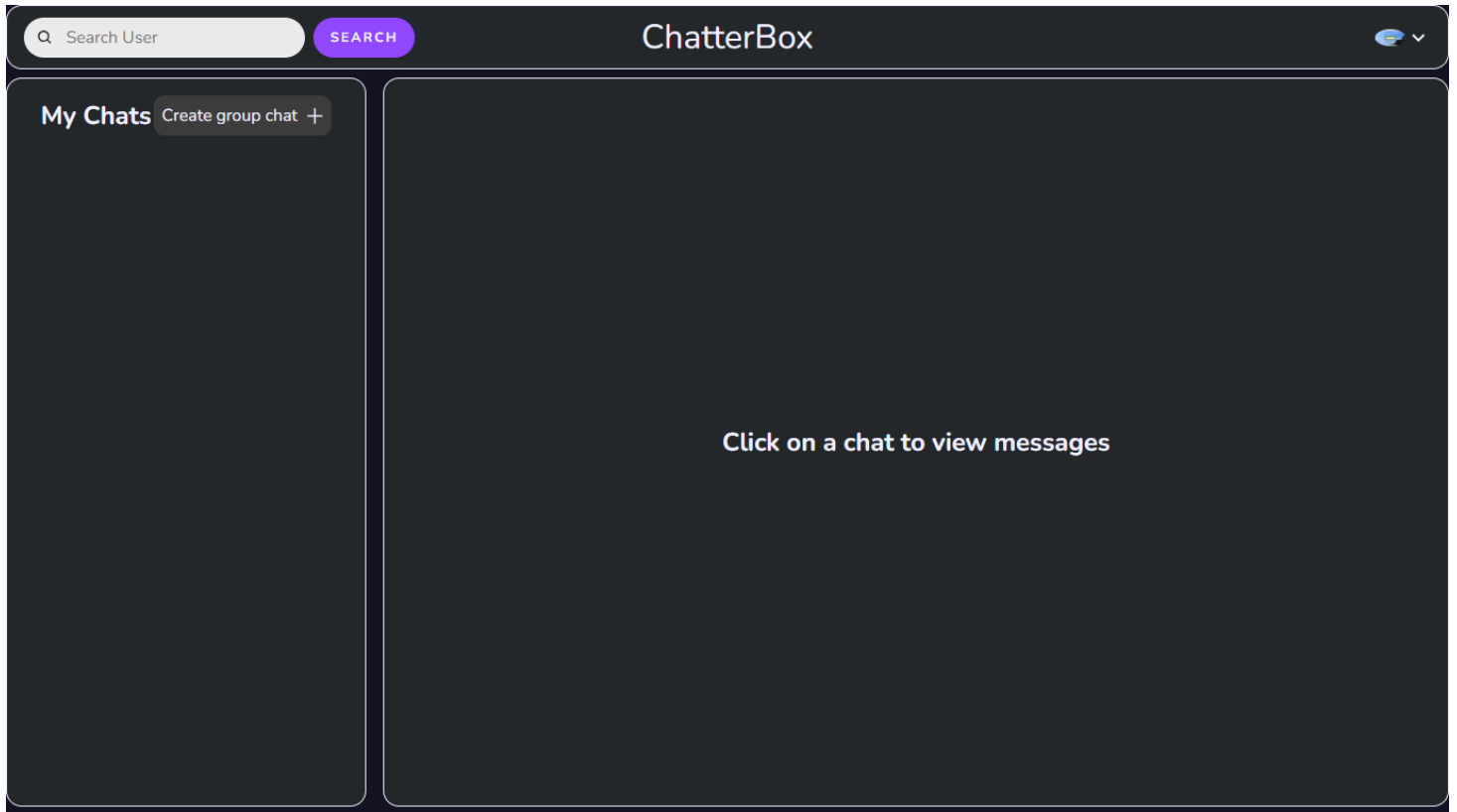
Enter your email

Enter your password

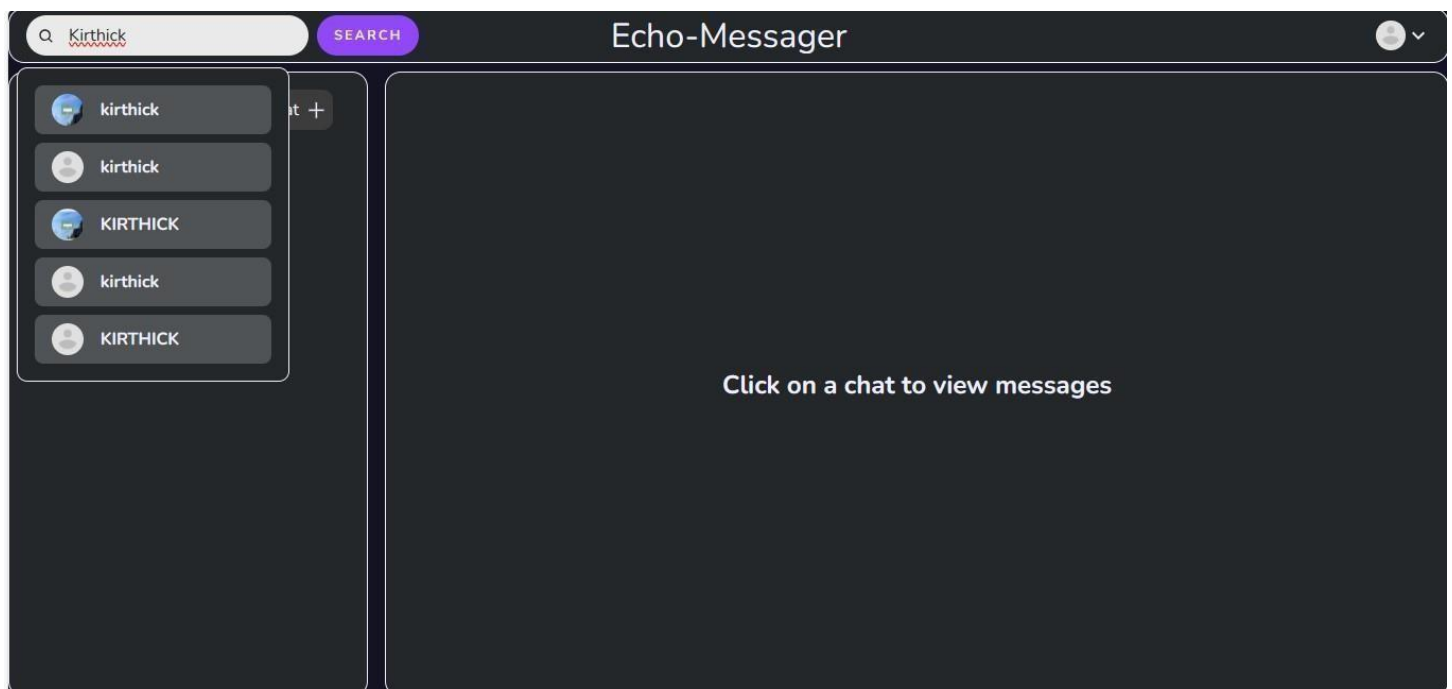
SIGN IN

Not a user? [Register](#)

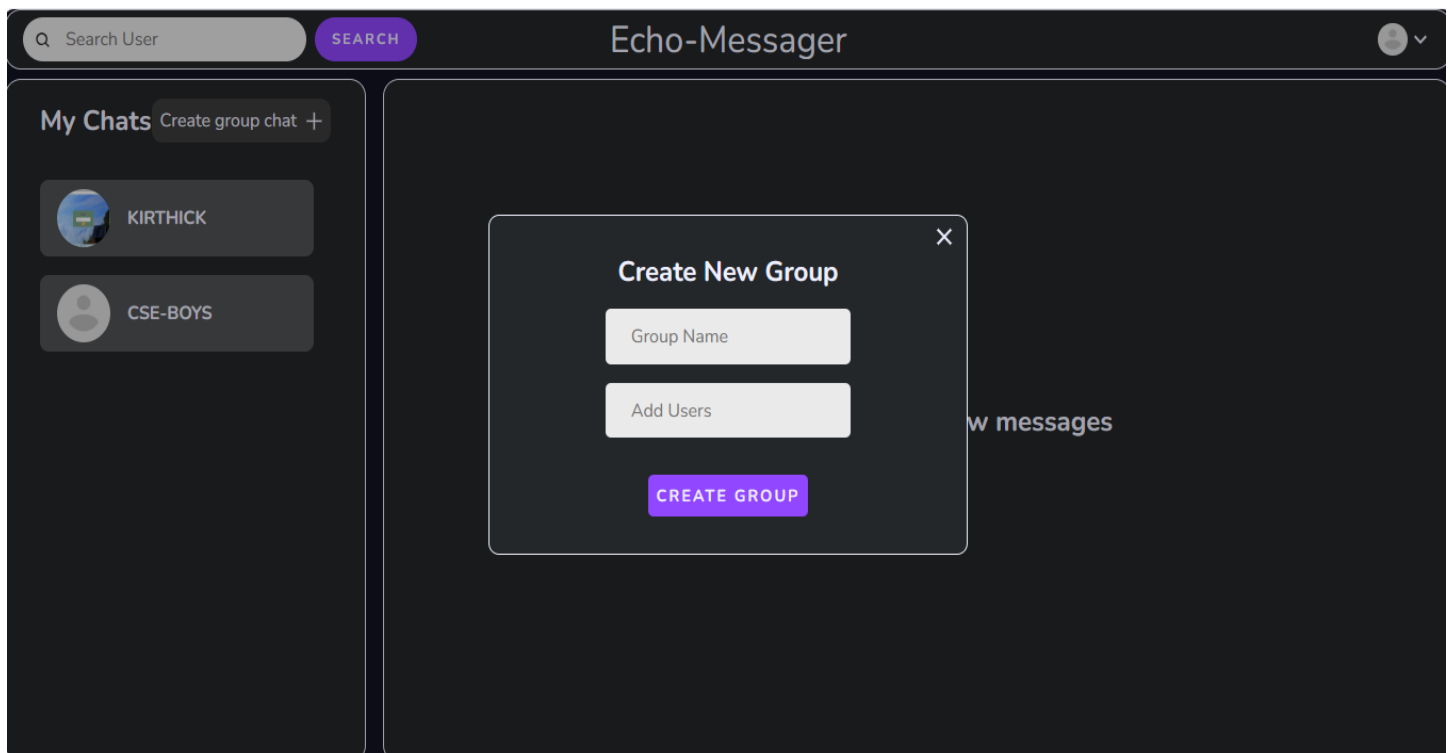
Figure 2



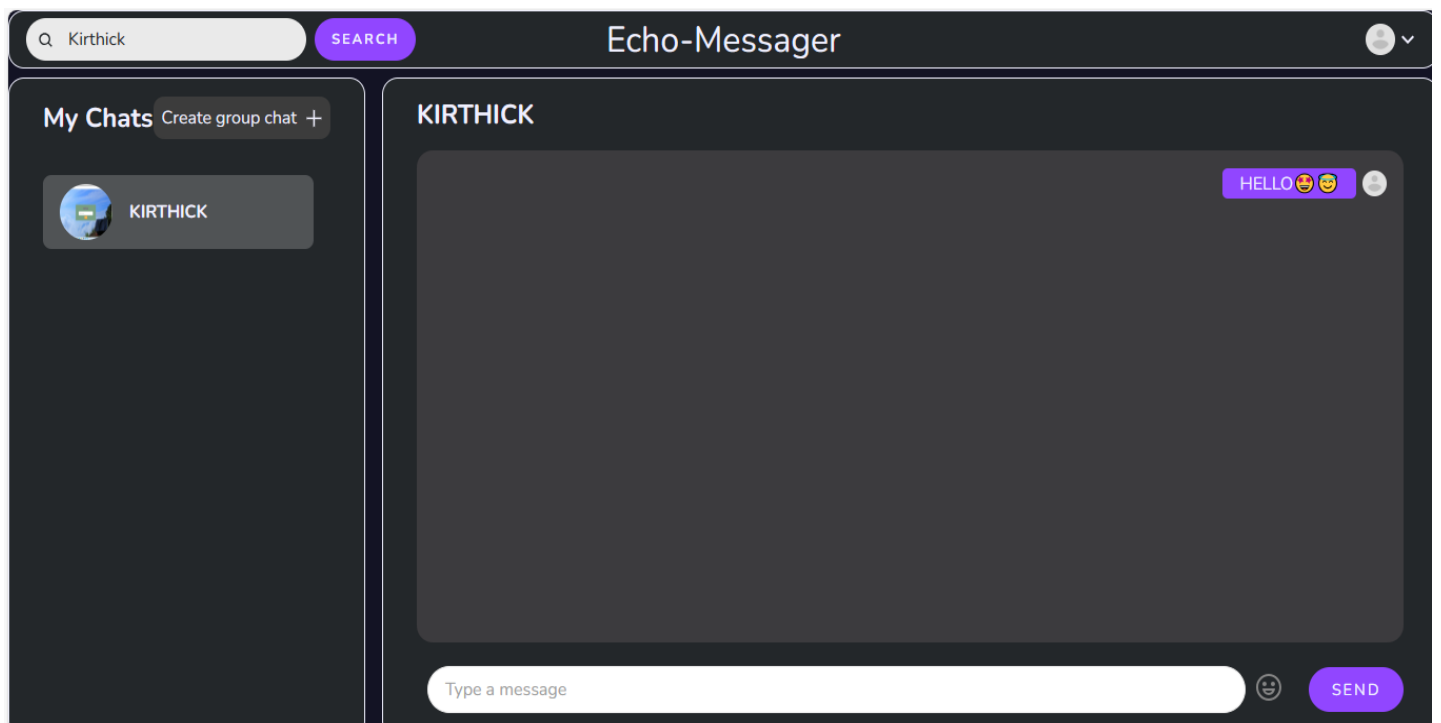
**Figure 3**



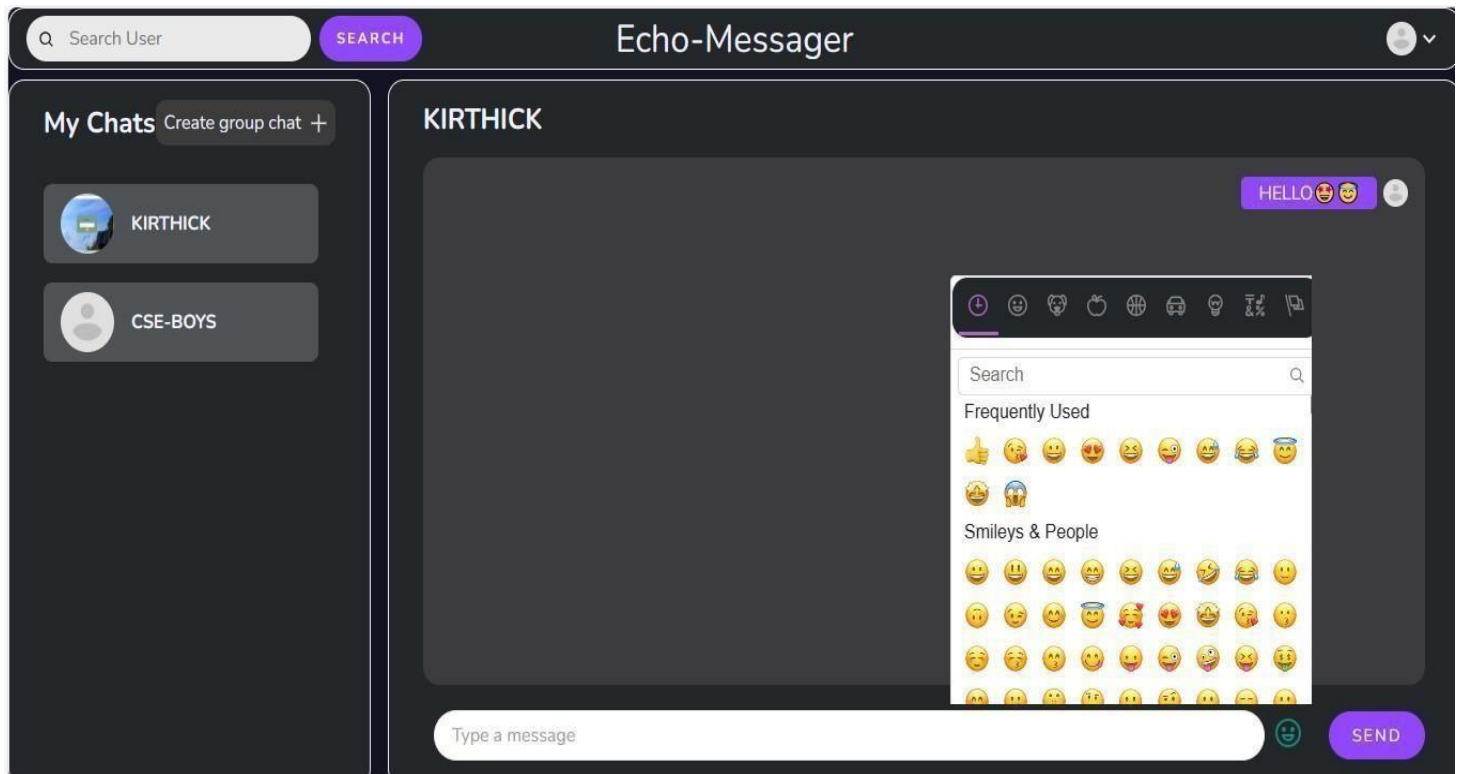
**Figure 4**



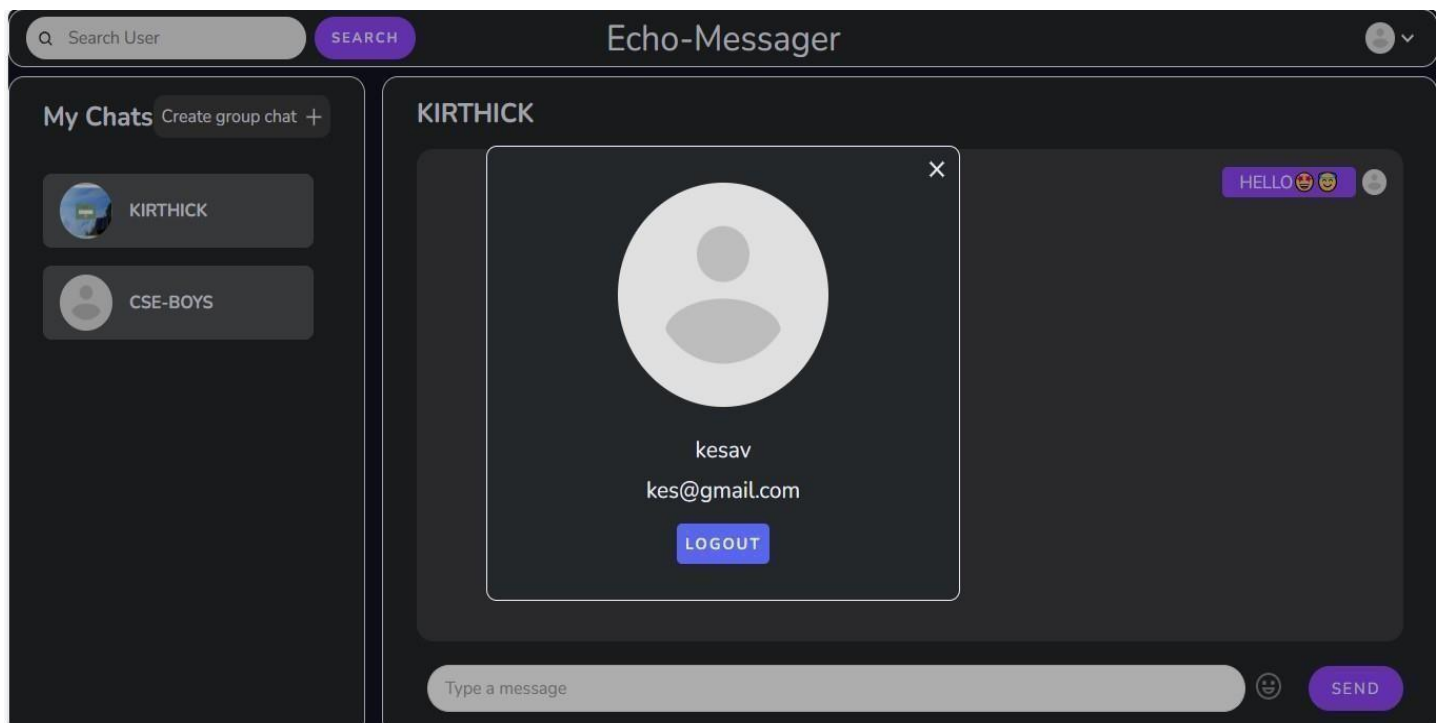
**Figure 5**



**Figure 6**



**Figure 7**



**Figure 8**



## **7. CONCLUSION AND FUTURE ENHANCEMENT**

In conclusion, Echo Messenger has established itself as a trusted and efficient chat application, offering users a secure and intuitive platform for communication. To further enhance its capabilities, future enhancements could focus on advanced security features, enhanced user engagement tools, and improved personalization options. Integration with AI and chatbots, cross- platform compatibility, and community-building features are also areas of potential development. By prioritizing user feedback and innovation, Echo Messenger can continue to evolve and meet the evolving needs of its users while maintaining its commitment to simplicity, security, and efficiency.

### **Future Enhancements:**

Future enhancements for Echo Messenger could include integrating end-to-end encryption for enhanced security, implementing voice and video calling features for richer communication experiences, introducing AI-powered chatbots for automated assistance, and expanding cross-platform compatibility to ensure seamless communication across devices. Additionally, enhancing user customization options, such as customizable themes and stickers, and introducing community-building features like public chat rooms or interest-based groups, could further enrich the user experience. By prioritizing these enhancements, Echo Messenger can continue to evolve as a leading chat application while meeting the diverse needs of its users.

## 8. APPENDIX(PROGRAM)

```
import React, { useContext, useEffect, useState } from "react";
import "../styles/navbar.css";
import { MdKeyboardArrowDown } from "react-icons/md";
import { BiSearch } from "react-icons/bi";
import jwt_decode from "jwt-decode";
import fetchData from "../helper/apiCall";
import AppContext from "../context/userContext";
import Modal from "../Modal";
import toast from "react-hot-toast";
import { useNavigate } from "react-router-dom";
import axios from "axios";

const Navbar = () => {
  const navigate = useNavigate();
  const [search, setSearch] = useState("");
  const { id } = jwt_decode(localStorage.getItem("token"));
  const [searchedUsers, setSearchedUsers] = useState([]);
  const { userInfo, setUserInfo, setLoading, resetStates } =
    useContext(AppContext);
  const [modalOpen, setModalOpen] = useState(false);

  const searchBtn = async () => {
    const data = await fetchData(`/user/searchuser/${id}?search=${search}`);
    setSearchedUsers(data);
    setSearch("");
  };
};
```

```

const fetch = async () => {
  const data = await fetchData(`/user/getuser/${id}/${id}`);
  setUserInfo(data);
};

useEffect(() => {
  fetch();
}, []);

const children = (
  <div className="profile__modal">
    <img
      src={
        userInfo.pic ||
        "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar-icon-25.jpg"
      }
      alt="profile-pic"
    />
    <span className="profile__name">{userInfo.name}</span>
    <span className="profile__email">{userInfo.email}</span>
  </div>
);

const logout = () => {
  localStorage.clear();
  resetStates();
  navigate("/");
  toast.success("Logged out successfully");
};

```

```

const createChat = async (ele) => {
  try {
    setLoading(true);
    const { data } = await axios.post(
      `/chat/createchat/${id}`,
      {
        otherid: ele._id,
      },
      {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
        },
      }
    );
    setLoading(false);
    setSearchedUsers([]);
    return;
  } catch (error) {
    return error;
  }
};

```

```

return (
  <nav>
    <div className="search">
      <input
        type="text"
        placeholder="Search User"

```

```

    className="form-input search-input"
    value={search}
    onChange={(e) => {
      setSearch(e.target.value);
    }}
  />
<BiSearch />
<button
  className="btn search__btn"
  onClick={searchBtn}
>
  Search
</button>
</div>
{searchedUsers.length > 0 ? (
  <div className="searched-users">
    {searchedUsers.map((ele) => {
      return (
        <div
          className="usercard"
          key={ele._id}
          onClick={() => {
            createChat(ele);
          }}
        >
          <div
            className={`usercard__image flex-center search-card__image`}
            >
            <img

```

```
src={
    ele.pic ||
    "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar-icon-
25.jpg"
}
alt="contact-pic"
/>
</div>
<div className="usercontent__info">
    <h4 className="usercontent__name">{ele.name}</h4>
</div>
</div>
);
}}) : (
    <&&</>
))}
<div className="app__name">Echo-Messenger</div>
<div className="navbar__icons">
    <div className="profile__image">
        <img
            src={
                userInfo.pic ||
                "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar-icon-
25.jpg"
            }
            alt="profile-pic"
        />
```

```

</div>
<MdKeyboardArrowDown
  className="arrow-icon"
  onClick={() => {
    setModalOpen(!modalOpen);
  }}
/>
</div>
{ modalOpen && (
  <Modal
    setModalOpen={ setModalOpen }
    children={ children }
    handleClick={ logout }
    btnname={ "Logout" }
    mClass="nav-profile-modal"
  />
)}
</nav>

```

```
export default Navbar;
```

```

import React, { useEffect, useState, useContext, useRef } from "react";
import jwt_decode from "jwt-decode";
import "../styles/chat.css";
import SendChat from "../SendChat";
import AppContext from "../context/userContext";
import fetchData from "../helper/apiCall";

```

```

import { AiFillSetting } from "react-icons/ai";
import GroupChatModal from "../GroupChatModal";
import io from "socket.io-client";

const Chat = () => {
  const { currentChat, loading, setLoading } = useContext(AppContext);
  const [messages, setMessages] = useState([]);
  const { id } = jwt_decode(localStorage.getItem("token"));
  const [userInfo, setUserInfo] = useState({ });
  const [modalOpen, setModalOpen] = useState(false);
  const [newmsg, setNewmsg] = useState({ });
  let socket = useRef();

  const getAllMessages = async () => {
    setLoading(true);
    const data = await fetchData(
      `/message/getallmessages/${id}/${currentChat._id}`
    );

    setMessages(data);
    socket.current.emit("join-chat", currentChat._id);
    setLoading(false);
  };

  useEffect(() => {
    if (currentChat._id) {
      getAllMessages();
    }
  }, [currentChat]);

```



```

useEffect(() => {
  socket.current?.on("message-recieved", (newMessage) => {
    setMessages([...messages, newMessage.data]);
  });
});

```

```

useEffect(() => {
  socket.current = io(process.env.REACT_APP_SOCKET_ENDPOINT);
  socket.current.emit("setup", id);
}, []);

```

```

useEffect(() => {
  scroll.current?.scrollIntoView({ behavior: "smooth" });
}, [messages]);

```

```

const scroll = useRef();

```

```

const fetchUser = async () => {
  const otheruser = currentChat.users.filter((user) => user._id !== id)[0];
  setUserInfo(otheruser);
};

```

```

useEffect(() => {
  if (currentChat._id) {
    fetchUser();
  }
}, [currentChat]);

```

```
if (loading) return <h2 className="flex-center loading">Loading...</h2>;
```

```
if (!currentChat._id)
```

```
  return (
```

```
    <section className="chat">
```

```
      <h2
```

```
        className="flex-center"
```

```
        style={{ height: "100%" }}>
```

```
      >
```

```
        Click on a chat to view messages
```

```
      </h2>
```

```
    </section>
```

```
  );
```

```
return (
```

```
  <section className="chat">
```

```
    <div className="chat__top">
```

```
      <h2 className="chat__name">
```

```
        {currentChat.isGroupChat ? currentChat.chatName : userInfo.name}
```

```
      </h2>
```

```
      {currentChat.isGroupChat && (
```

```
        <AiFillSetting
```

```
          className="gear-icon"
```

```
          onClick={() => {
```

```
            setModalOpen(true);
```

```
          }}>
```

```
        />
```

```
      )}
```

```
    </div>
```

```

<GroupChatModal
  modalOpen={modalOpen}
  setModalOpen={setModalOpen}
  type={"update"}
  groupname={currentChat.chatName}
  group={currentChat}
/>

<div className="chat__box">
  {messages.map((ele, i) => {
    return ele.senderId === id ? (
      <div
        className="single-chat own__chat"
        key={i}
        ref={scroll}
      >
        <div className="single-chat____content flex-center">
          <p>{ele.content}</p>
        </div>
        <div className="single-chat____pic flex-center">
          <img
            src={
              ele.senderPic ||
              "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar-icon-
25.jpg"
            }
            alt="pic"
          />
        </div>
      </div>
    ) : null
  })}
</div>

```

```

): (
  <div
    className="single-chat"
    key={i}
    ref={scroll}
  >
    <div className="single-chat____pic flex-center">
      <img
        src={
          ele.senderPic ||
          "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar-icon-
25.jpg"
        }
        alt="pic"
      />
    </div>
    <div className="single-chat____content flex-center">
      <p>{ele.content}</p>
    </div>
  </div>
);
)}}
</div>
<SendChat
  socket={socket}
  setMessages={setMessages}
  messages={messages}
  setNewmsg={setNewmsg}
/>

```

```

    </section>

    );
};

export default Chat;

import React, { useState } from "react";
import { NavLink, useNavigate } from "react-router-dom";
import "../styles/register.css";
import axios from "axios";
import toast from "react-hot-toast";

axios.defaults.baseURL = process.env.REACT_APP_SERVER_DOMAIN;

function Login() {
  const [formDetails, setFormDetails] = useState({
    email: "",
    password: "",
  });
  const navigate = useNavigate();

  const inputChange = (e) => {
    const { name, value } = e.target;
    return setFormDetails({
      ...formDetails,
      [name]: value,
    });
  };
};

```

```

const formSubmit = async (e) => {
  try {
    e.preventDefault();
    const { email, password } = formDetails;
    if (!email || !password) {
      return toast.error("Input field should not be empty");
    } else if (password.length < 5) {
      return toast.error("Password must be at least 5 characters long");
    }

    const { data } = await toast.promise(
      axios.post("/user/login", {
        email,
        password,
      }),
      {
        pending: "Logging in...",
        success: "Login successfully",
        error: "Invalid credentials",
        loading: "Logging user...",
      }
    );
    localStorage.setItem("token", data.token);
    return navigate("/chats");
  } catch (error) {
    return error;
  }
};

```

```

return (
  <section className="register-section flex-center">
    <div className="register-container flex-center">
      <h2 className="form-heading">Sign In</h2>
      <form onSubmit={formSubmit} className="register-form">
        <input
          type="email"
          name="email"
          className="form-input"
          placeholder="Enter your email"
          value={formDetails.email}
          onChange={inputChange}
        />
        <input
          type="password"
          name="password"
          className="form-input"
          placeholder="Enter your password"
          value={formDetails.password}
          onChange={inputChange}
        />
        <button type="submit" className="btn form-btn">
          sign in
        </button>
      </form>
      <p>
        Not a user?{ " "}
        <NavLink className="login-link" to={"/register"}>

```

```

        Register
      </NavLink>
    </p>
  </div>
</section>
);
}

export default Login;

import React, { useState } from "react";
import { NavLink, useNavigate } from "react-router-dom";
import "../styles/register.css";
import axios from "axios";
import toast from "react-hot-toast";

axios.defaults.baseURL = process.env.REACT_APP_SERVER_DOMAIN;

function Register() {
  const [loading, setLoading] = useState(false);
  const [files, setFiles] = useState("");
  const [formDetails, setFormDetails] = useState({
    name: "",
    email: "",
    password: "",
    confpassword: "",
  });
  const navigate = useNavigate();

```



```

const inputChange = async (e) => {
  const { name } = e.target;
  return setFormDetails({
    ...formDetails,
    [name]: e.target.value,
  });
};

```

```

const onUpload = (element) => {
  setLoading(true);
  if (element.type === "image/jpeg" || element.type === "image/png") {
    const data = new FormData();
    data.append("file", element);
    data.append("upload_preset", "zenstore");
    data.append("cloud_name", process.env.REACT_APP_CLOUDINARY_CLOUD_NAME);
    fetch(process.env.REACT_APP_CLOUDINARY_BASE_URL, {
      method: "POST",
      body: data,
    })
      .then((res) => res.json())
      .then((data) => setFiles(data.url.toString()));
    setLoading(false);
  } else {
    setLoading(false);
    toast.error("Please select an image in jpeg or png format");
  }
};

```

```

const formSubmit = async (e) => {

```

```

try {
  e.preventDefault();
  let { name, email, password, confpassword } = formDetails;
  if (!name || !email || !password || !confpassword) {
    return toast.error("Input field should not be empty");
  } else if (password.length < 5) {
    return toast.error("Password must be at least 5 characters long");
  } else if (password !== confpassword) {
    return toast.error("Passwords do not match");
  }
  if (files === "") {
    setFiles(
      "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar-icon-25.jpg"
    );
  }
  const { data } = await toast.promise(
    axios.post("/user/register", {
      name,
      pic: files,
      email,
      password,
    }),
    {
      pending: "Registering user...",
      success: "User registered successfully",
      error: "Unable to register user",
      loading: "Registering user...",
    }
  );
};

```

```

    return navigate("/");
  } catch (error) {
    return error;
  }
};

```

```

return (
  <section className="register-section flex-center">
    <div className="register-container flex-center">
      <h2 className="form-heading">Sign Up</h2>
      <form
        onSubmit={ formSubmit }
        className="register-form"
      >
        <input
          type="text"
          name="name"
          className="form-input"
          placeholder="Enter your name"
          value={ formDetails.name }
          onChange={ inputChange }
        />
        <input
          type="email"
          name="email"
          className="form-input"
          placeholder="Enter your email"
          value={ formDetails.email }
          onChange={ inputChange }

```

```

/>
<input
  type="file"
  name="pic"
  className="form-input"
  onChange={(e) => {
    onUpload(e.target.files[0]);
  }}
/>
<input
  type="password"
  name="password"
  className="form-input"
  placeholder="Enter your password"
  value={ formDetails.password }
  onChange={ inputChange }
/>
<input
  type="password"
  name="confpassword"
  className="form-input"
  placeholder="Confirm your password"
  value={ formDetails.confpassword }
  onChange={ inputChange }
/>
<button
  type="submit"
  className="btn form-btn"
  disabled={ loading ? true : false }

```

```

    >
      sign up
    </button>
  </form>
  <p>
    Already a user?{ " " }
    <NavLink
      className="login-link"
      to={"/"}
    >
      Log in
    </NavLink>
  </p>
</div>
</section>
);
}

```

```
export default Register;
```

```

import Chat from '../components/Chat'
import MyChats from '../components/MyChats'
import '../styles/chats.css'
import React from 'react'
import Navbar from '../components/Navbar'

const Chats = () => {

```

```

return (
  <>
    <Navbar />
    <div className='chats__layout'>
      <MyChats />
      <Chat />
    </div>
  </>
)
}

```

export default Chats

```

import React, { useContext, useEffect, useState } from "react";
import "../styles/mychats.css";
import UserCard from "../UserCard";
import jwt_decode from "jwt-decode";
import { AiOutlinePlus } from "react-icons/ai";
import GroupChatModal from "../GroupChatModal";
import fetchData from "../helper/apiCall";
import AppContext from "../context/userContext";

```

```

const MyChats = () => {
  const { id } = jwt_decode(localStorage.getItem("token"));
  const [myChats, setMyChats] = useState([]);
  const [modalOpen, setModalOpen] = useState(false);
  const { setCurrentChat, loading } = useContext(AppContext);

  const fetchAllChats = async () => {

```

```

const data = await fetchData(`/chat/getchats/${id}`);
setMyChats(data);
};

useEffect(() => {
  fetchAllChats();
}, [loading]);

const clickFunc = (other) => {
  setCurrentChat(other);
};

// if (loading) return <h2 className="flex-center loading">Loading...</h2>;

return (
  <section className="mychats">
    <div className="mychats-top">
      <h2 className="mychats__heading">My Chats</h2>
      <div
        className="create-group"
        onClick={() => {
          setModalOpen(true);
        }}
      >
        <span>Create group chat</span>
        <AiOutlinePlus className="plus__icon" />
      </div>
    </div>
    <div className="allusers">

```

```

    <GroupChatModal
      modalOpen={ modalOpen }
      setModalOpen={ setModalOpen }
      type={ "create" }
      myChats={ myChats }
      setMyChats={ setMyChats }
    />
  </section>
);
};

export default MyChats;

const mongoose = require("mongoose");
mongoose.set("strictQuery", false);
const mongodbconn = mongoose
  .connect(process.env.MONGO_URI)
  .then(() => { })
  .catch(() => { });

module.exports = mongodbconn;

```



## **9. REFERENCE**

1. Interface Design, Alan Dix, Janet Finlay, Gregory D.Abowd 2023, Mobile Human Computer Interaction Journal.
2. The Mobile Human-Computer Interaction Journal, IEEE Access Volume 8, Dan Jurafsky,James H. Martin,2022, Wiley Social Platform.
3. A Comprehensive Survey of Echo Messaging Applications, Dr. Michael Smith,2021, Pearson Education.
4. Evolution and Social Implications, Ross Anderson,2020,IEEE Xplore Volume15,International Journal of Communication.
5. In 2023, Jhalak Mittal, Arushi Garg, and Shivani Sharma titled "Enhancing User Engagement in Online Chat Systems" published in the International Journal of Advanced Computer Science.