UNIVERSITY OF MUMBAI

ADITYA INSTITUTE OF MANAGEMENT STUDIES AND RESEARCH

AIMSR

CREATING EMPLOYABLE PROFESSIONALS

NAAC Accredited || ISO 21001 : 2018 Certified

Master Of Computer Application

Mini Project Report on

WeChat : A Chat Application

Submitted By Kajal Purbiya & Sundaram Gupta

Roll No. 32 & 10.

Under the Guidance of Prof. Rashmi Pathak.

1. Introduction	3
Introduction of Project	4
Project Purpose	4
Project Objective	4
Scope of Project	5
2. System Study	6
Requirement Analysis	7
Planning and Scheduling	7
Preliminary Product Description	8
Justification of Platform	8
3. Analysis and Design	9
Hardware Requirements	10
Software Requirements	10
Actual Gantt Chart	10
System Design	11
UML Diagrams	11
4. Testing and Validation	12
Code Efficiency	13
Testing Approach	13
Unit Testing	13
Integration Testing	13
5.User Manual	14
User Documentation	15
Outputs	15
Code Snippets	18
6.Conclusion	22
Conclusion	23
Limitations	23
Future Scope	23

MODULE 1

1. Introduction

a. Introduction of Project

b. Project Purpose

c. Project Objective

d. Scope of Project

1. Introduction

A. Introduction of Project

The project aims to develop a robust and efficient Chat Application named "WeChat" In an era where digital security and Information is of paramount importance, WeChat serves as a secure platform for transporting sensitive messages from one user to another. With a core focus on security, the system employs AES-128 Encryption for the purpose of securing and Organizing user data. This application is designed to streamline the process of managing and retrieving messages while prioritizing data confidentiality. Leveraging advanced cryptographic techniques and arithmetic encoding, this Chat app provides users with a reliable solution to safeguard their chatting Info.

B. Project Purpose

Traditional chat management systems may fall short in providing adequate protection. The project addresses these issues by offering a messaging application with end-to-end encryption solution with enhanced security features. The primary purpose of the password vault is to address the common challenges users face in managing multiple passwords across various online platforms. It provides a secure and user-friendly environment for storing, retrieving, and organizing passwords. The use of arithmetic encoding contributes to the overall security of the system, ensuring that stored passwords remain confidential.

C. Project Objective

The primary objective of the WeChat project is to design and implement a state-of-the-art password vault that ensures the highest standards of security for user credentials. The system aims to:

- Provide a user-friendly interface for effortless password management.
- Implement strong encryption algorithms to protect stored passwords.
- Utilize arithmetic encoding to enhance the encoding and decoding processes.
- Mitigate the risk of password-related security breaches

D. Scope of Project

The scope of the WeChat project encompasses various dimensions:

• **Security:**
Implementing robust encryption mechanisms to safeguard user passwords and sensitive data from unauthorized access.

• **Usability:**
Creating an intuitive and user-friendly interface for easy navigation and efficient password management. Integration: Exploring the integration of arithmetic encoding to enhance the security of password encoding and decoding processes.

• **Compatibility:**
Ensuring compatibility with multiple platforms and devices to accommodate diverse user preferences

Module 2

2. System Study

Requirement Analysis
Planning and Scheduling
Preliminary Product Description
Justification of Platform
Conceptual Model

A. Requirement Analysis:

Requirement analysis is a crucial step in the development of any software application, including a chat application. It helps in understanding and defining the scope of the project, identifying user needs, and establishing the features and functionalities that the application must have. Here's a comprehensive list of requirements for a chat application.

B. Planning and Scheduling

Planning and scheduling are critical aspects of developing a chat application. Below is a suggested plan and schedule for the development of a chat application. Keep in mind that the timeline and tasks may vary based on the complexity of your specific project.

B.1. Project Kick-off and Planning:

- Define project goals, objectives, and success criteria.
- Identify stakeholders and establish communication channels.
- Conduct a thorough requirement analysis.

B.2. Technical Feasibility:

- Assess the technical feasibility of the project.
- Choose the technology stack (backend, frontend, database).
- Set up development environments.

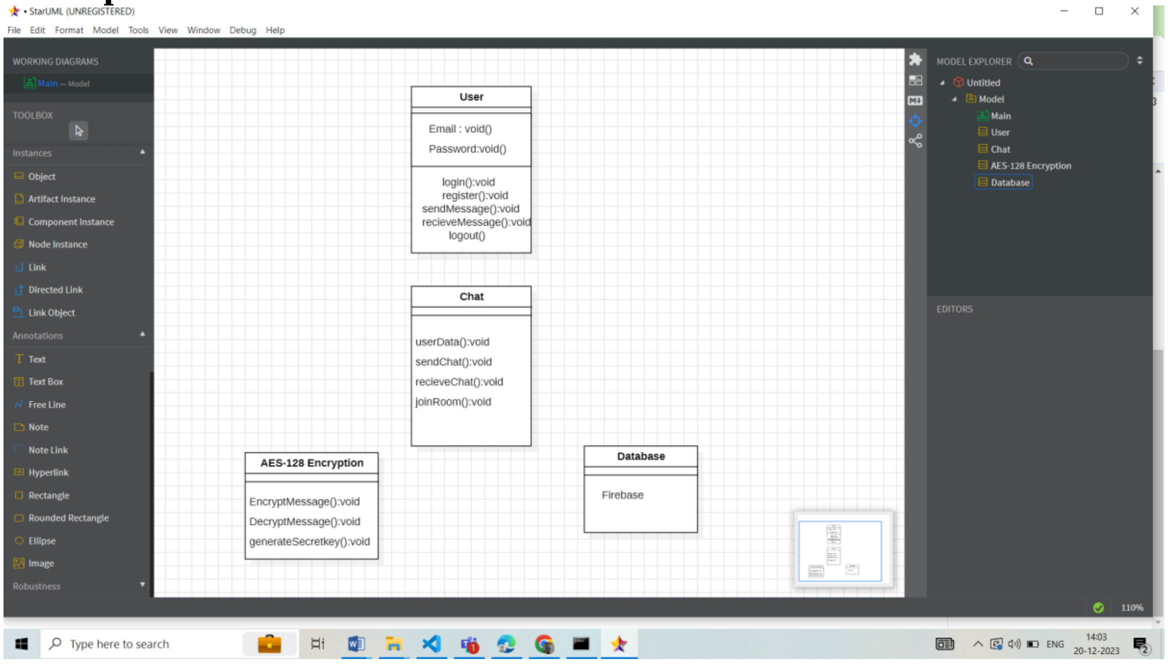
C. Preliminary Product Description

Describe the core features, such as secure password storage, retrieval, and user authentication.
Highlight the security protocols in place, including encryption methods and access controls.
Briefly mention how the system ensures compatibility across different browsers and devices.

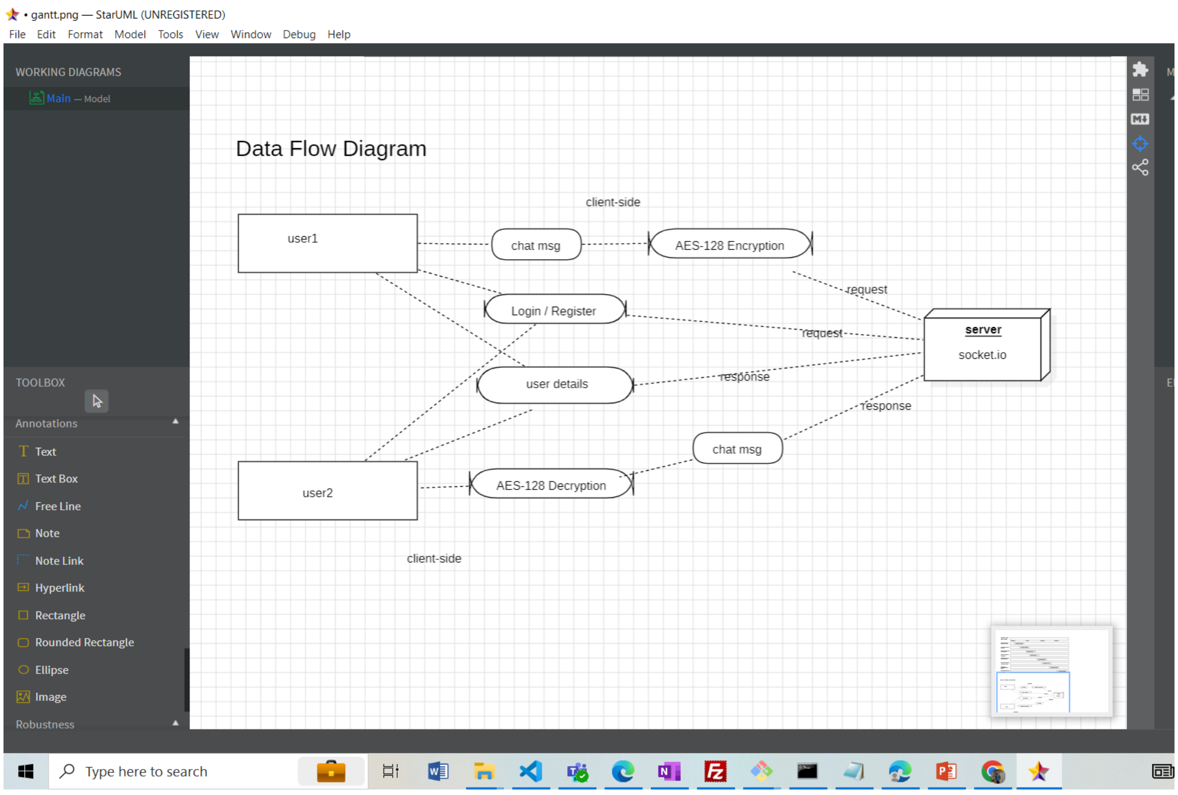
D. Justification of Platform

Node and Express: Justify the use of Django and Python for the backend, emphasizing their security features, scalability, and ease of development. HTML, CSS, and JavaScript: Explain the choice of these front-end technologies, highlighting their role in creating an intuitive and responsive user interface.

E. Conceptual Model



F. Data Flow Diagram



MODULE 3

3. Analysis and Design

Hardware Requirements
Software Requirements
Actual Gantt Chart
System Design
UML Diagrams

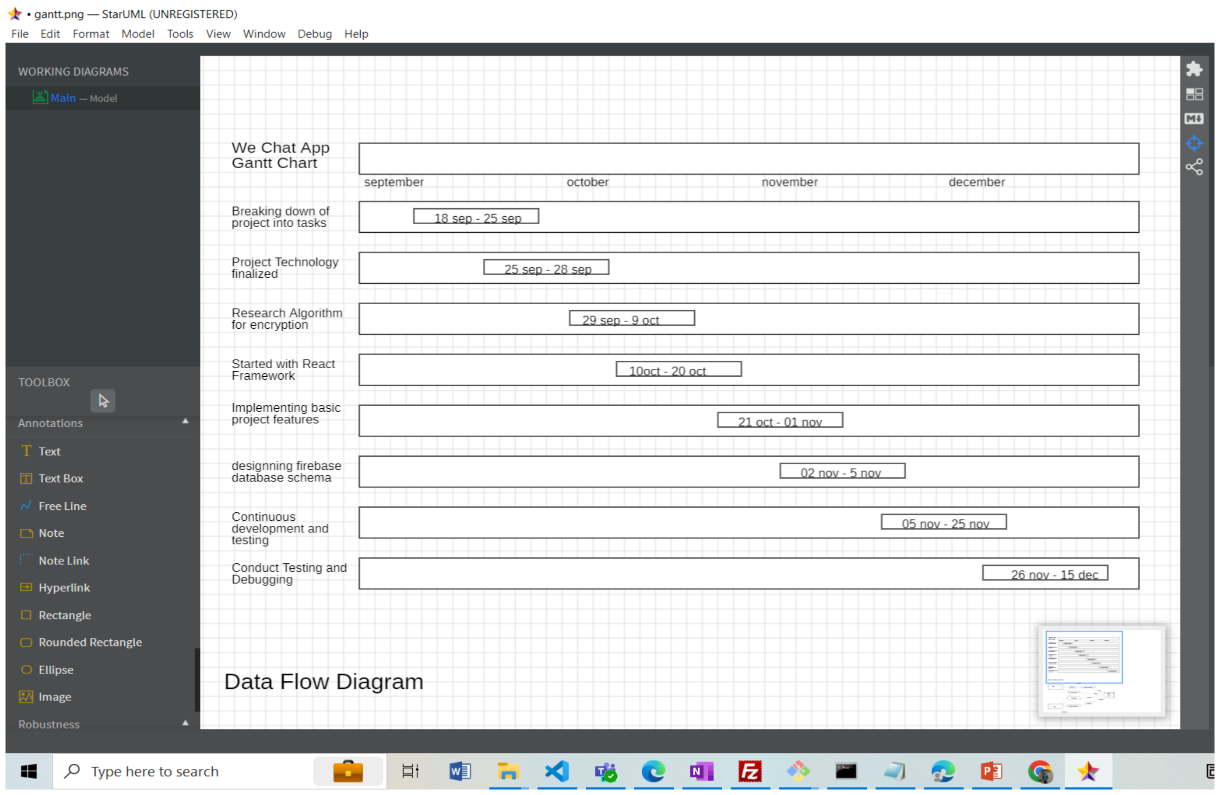
A. Hardware Requirements

• Processor: Dual-core or higher
• RAM: 8 GB or higher
• Storage: SSD for better performance

B. Software Requirements

• Operating System:
 • Windows
• Web Browser:
• Google Chrome, Mozilla Firefox, Safari
• Backend Development:
• Programming Language: Node
• Framework: Socket.io & Websockets
• Database: Firebase
• Frontend Development:
• HTML5, CSS3, JavaScript
• AES-128 Encryption & Decryption
• Version Control: Git and GitHub for source code management

C. Actual Gantt Chart



We Chat App Gantt Chart

september october november december

Breaking down of project into tasks — 18 sep - 25 sep
Project Technology finalized — 25 sep - 28 sep
Research Algorithm for encryption — 29 sep - 9 oct
Started with React Framework — 10oct - 20 oct
Implementing basic project features — 21 oct - 01 nov
designning firebase database schema — 02 nov - 5 nov
Continuous development and testing — 05 nov - 25 nov
Conduct Testing and Debugging — 26 nov - 15 dec

Data Flow Diagram

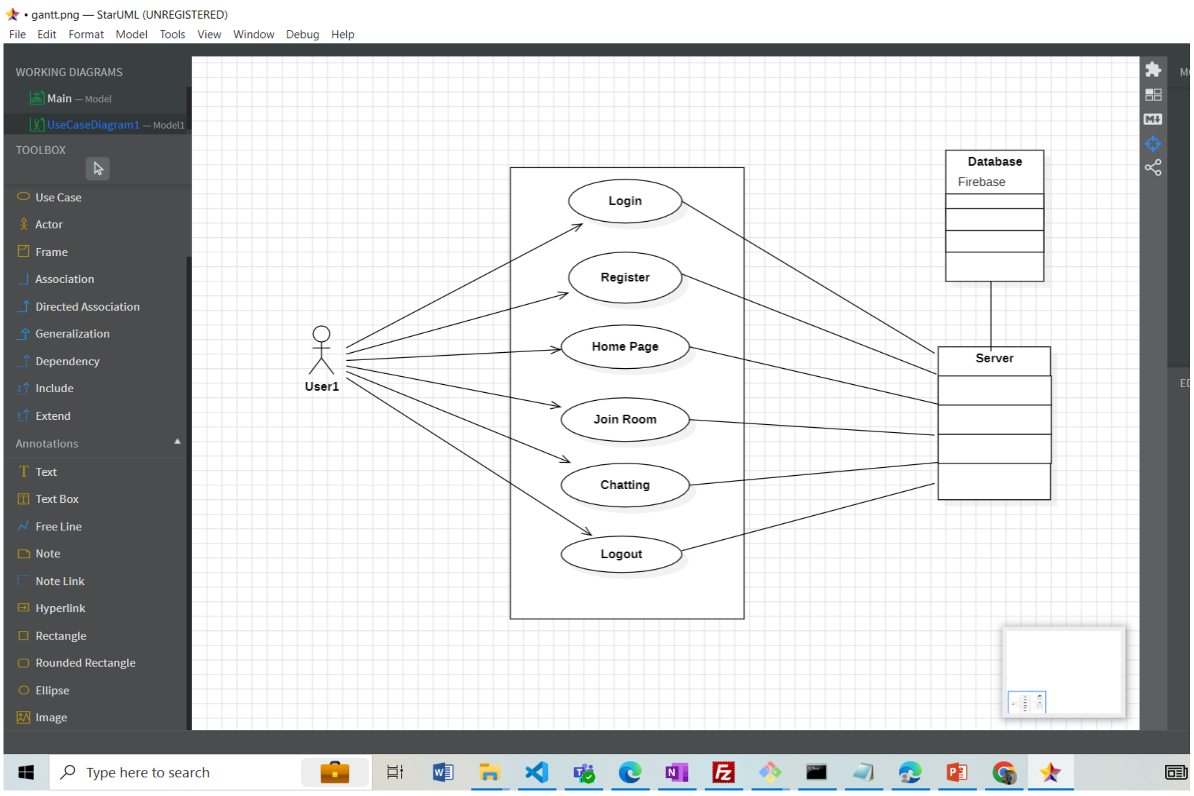
D. System Design

1. **User Interaction**: Users initiate the process by registering an account. Registered users access the system through secure login credentials. Users have the option to update account details, change passwords, or delete their accounts.

2. **Data Flow**: Users input data through the user interface, including passwords and account details. Passwords undergo encryption using arithmetic encoding before storage. The encrypted data is then stored securely in the MySQL database. When requested, the system retrieves encrypted data from the database. The retrieved data is decrypted using the appropriate algorithms before being presented to the user.

3. **Security Layers**: We have used Arithmetic Encoding to implement the security layer for our project. This helps us to encrypt our message and store it for retrieval for when we need to use it.

E. UML Use Case Diagrams



MODULE 4

4. Testing and Validation

Code Efficiency
Testing Approach
Unit Testing
Integration Testing

A. Code Efficiency

Arithmetic Encoding Algorithm:

Understand the arithmetic encoding algorithm thoroughly, and choose an efficient implementation that minimizes the number of arithmetic operations required.

Error Handling:

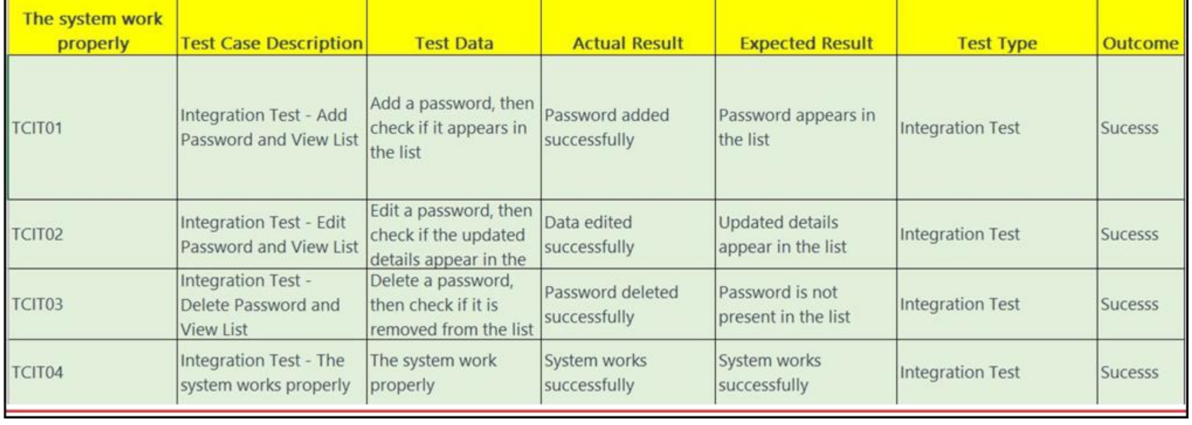
Implement efficient error-handling mechanisms without sacrificing performance. Be cautious about the impact of error checking on the overall efficiency of the encoding/decoding process.

Platform and Language Optimization:

Consider platform-specific and language-specific optimizations. Take advantage of language features or libraries that may provide performance benefits.

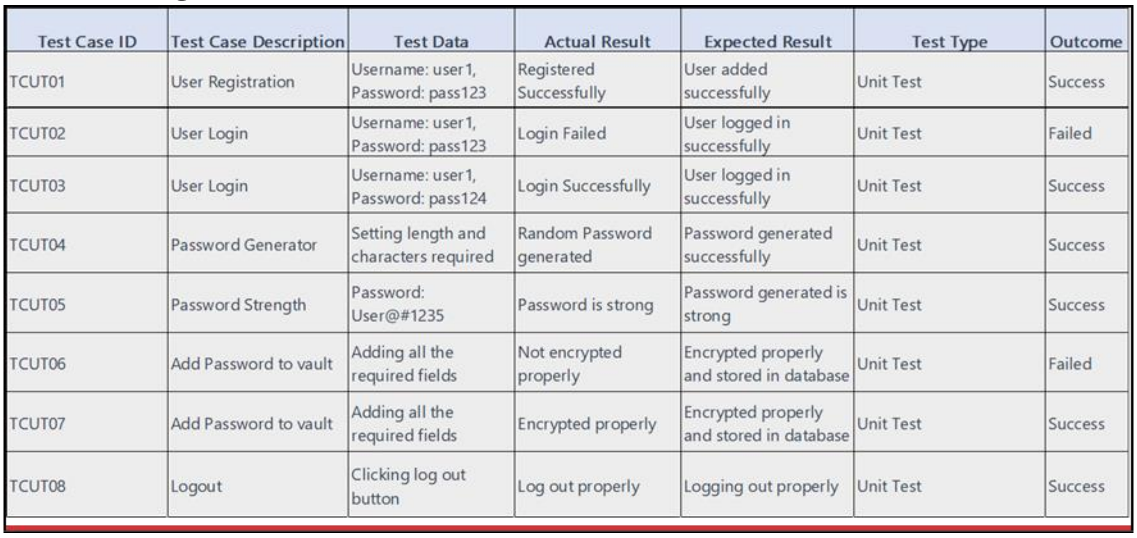
B. Testing Approach

- Integration Testing



The system work properly	Test Case Description	Test Data	Actual Result	Expected Result	Test Type	Outcome
TCIT01	Integration Test - Add Password and View List	Add a password, then check if it appears in the list	Password added successfully	Password appears in the list	Integration Test	Sucesss
TCIT02	Integration Test - Edit Password and View List	Edit a password, then check if the updated details appear in the	Data edited successfully	Updated details appear in the list	Integration Test	Sucesss
TCIT03	Integration Test - Delete Password and View List	Delete a password, then check if it is removed from the list	Password deleted successfully	Password is not present in the list	Integration Test	Sucesss
TCIT04	Integration Test - The system works properly	The system work properly	System works successfully	System works successfully	Integration Test	Sucesss

- Unit Testing



Test Case ID	Test Case Description	Test Data	Actual Result	Expected Result	Test Type	Outcome
TCUT01	User Registration	Username: user1, Password: pass123	Registered Successfully	User added successfully	Unit Test	Success
TCUT02	User Login	Username: user1, Password: pass123	Login Failed	User logged in successfully	Unit Test	Failed
TCUT03	User Login	Username: user1, Password: pass124	Login Successfully	User logged in successfully	Unit Test	Success
TCUT04	Password Generator	Setting length and characters required	Random Password generated	Password generated successfully	Unit Test	Success
TCUT05	Password Strength	Password: User@#1235	Password is strong	Password generated is strong	Unit Test	Success
TCUT06	Add Password to vault	Adding all the required fields	Not encrypted properly	Encrypted properly and stored in database	Unit Test	Failed
TCUT07	Add Password to vault	Adding all the required fields	Encrypted properly	Encrypted properly and stored in database	Unit Test	Success
TCUT08	Logout	Clicking log out button	Log out properly	Logging out properly	Unit Test	Success

MODULE 5

5. User Manual

User Documentation

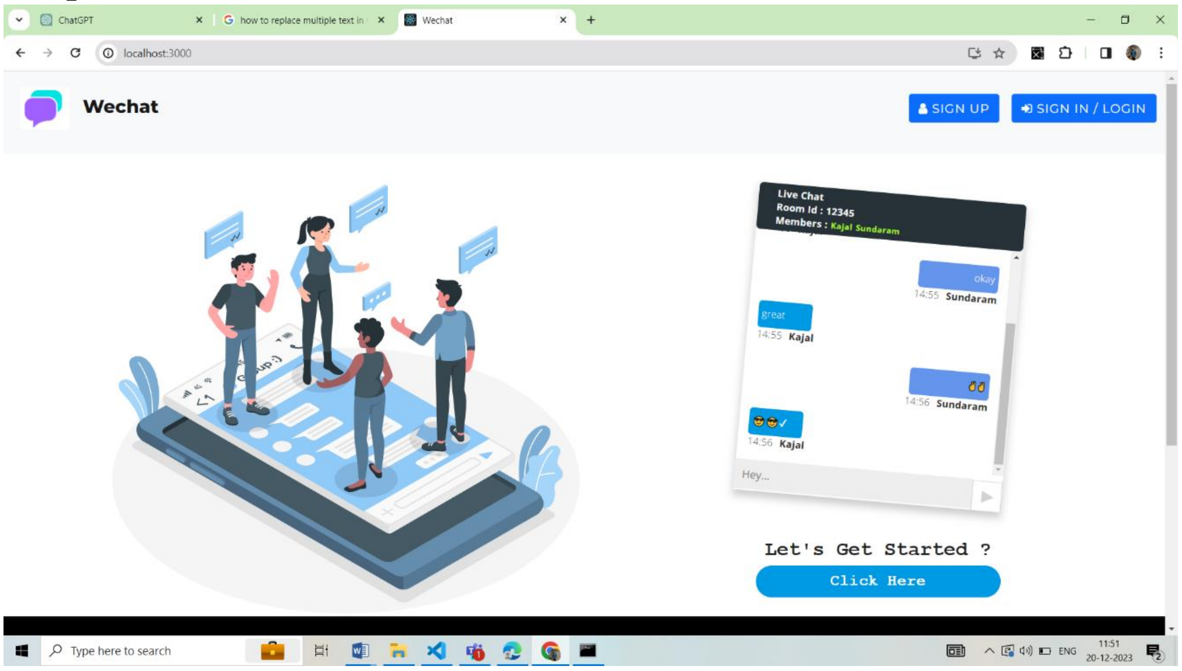
Outputs

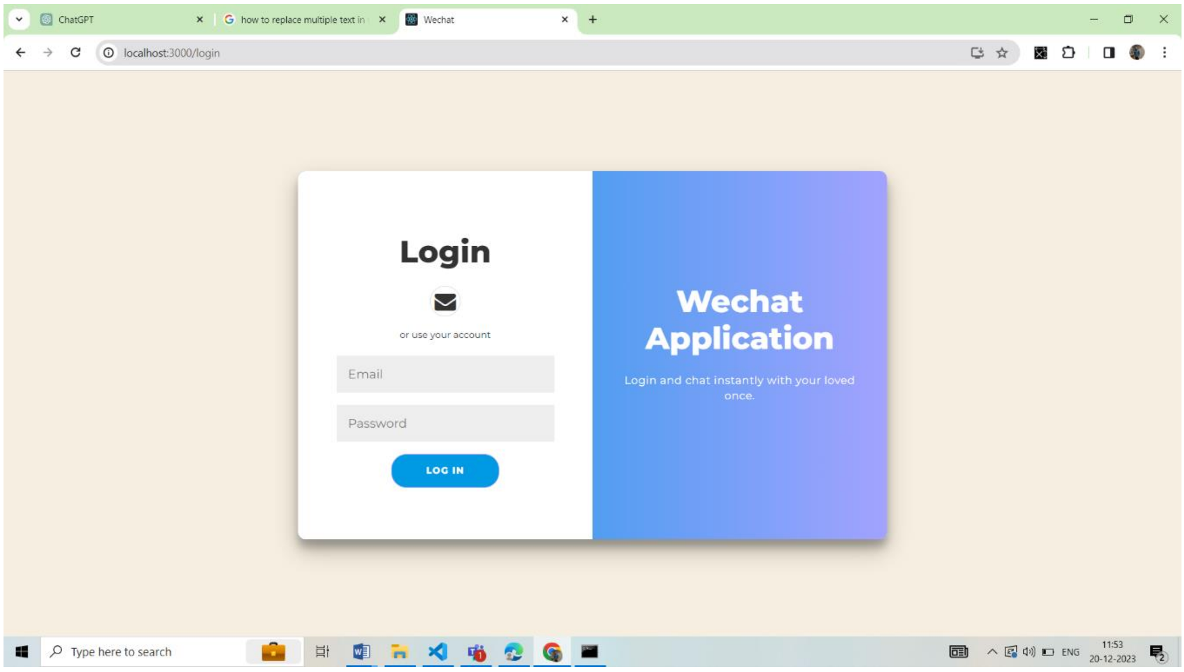
Code Snippets

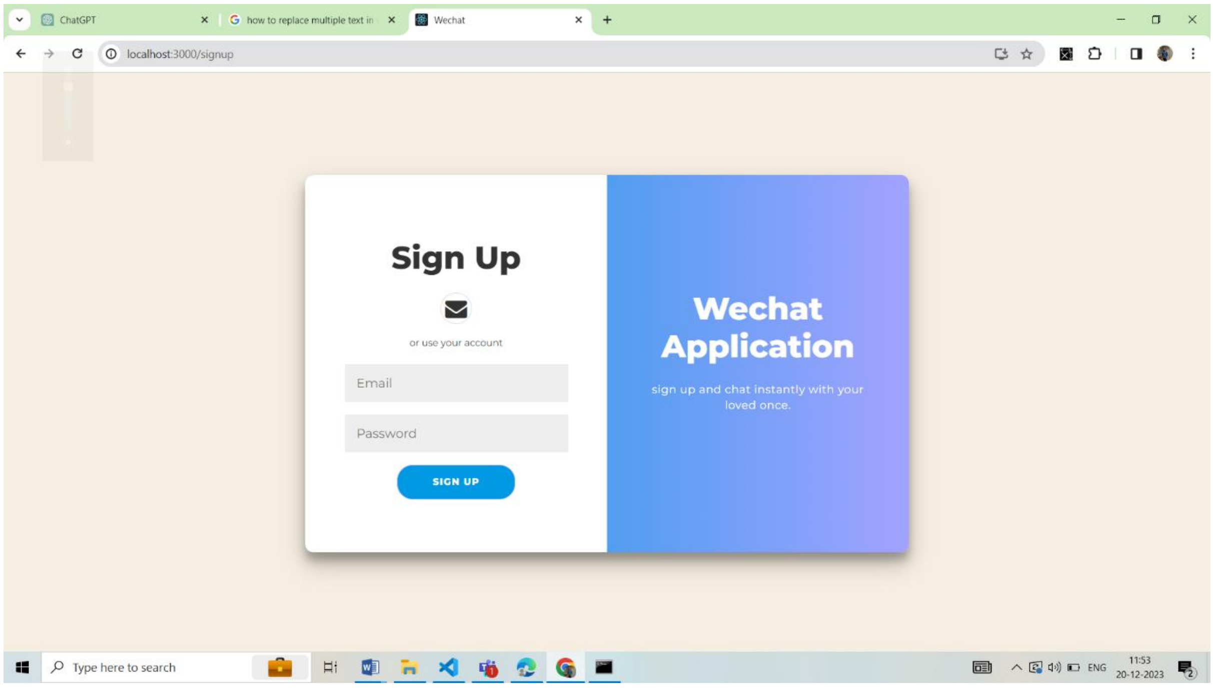
A. User Documentation

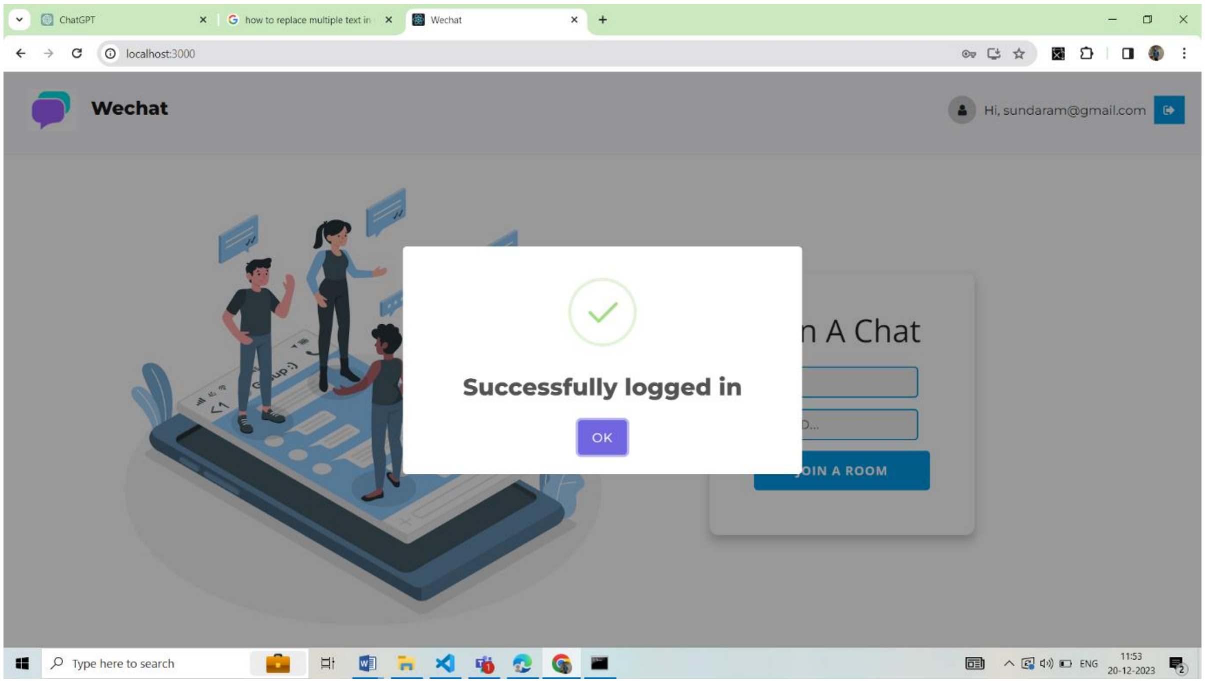
The user need to directly access the page through command npm start to runserver and can start navigating their way through the entire page.

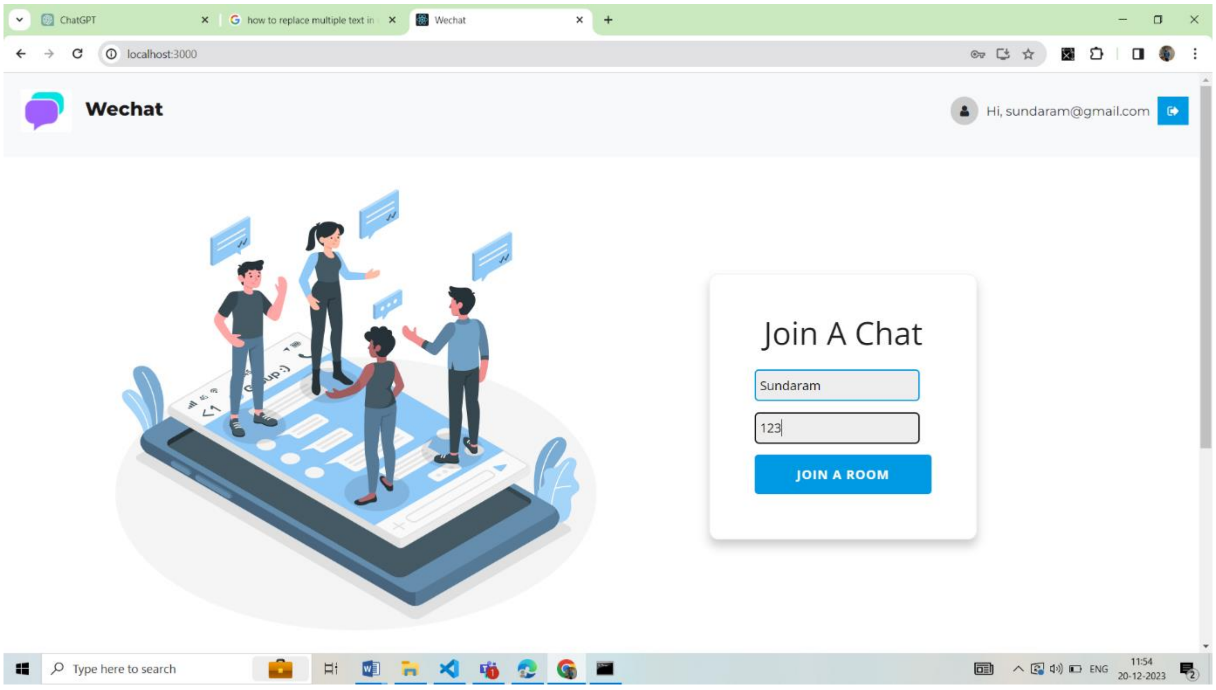
B. Output






localhost:3000/signup
Sign Up
or use your account
Email
Password
SIGN UP
Wechat Application
sign up and chat instantly with your loved once.


localhost:3000
Wechat
Hi, sundaram@gmail.com
n A Chat
JOIN A ROOM
Successfully logged in
OK

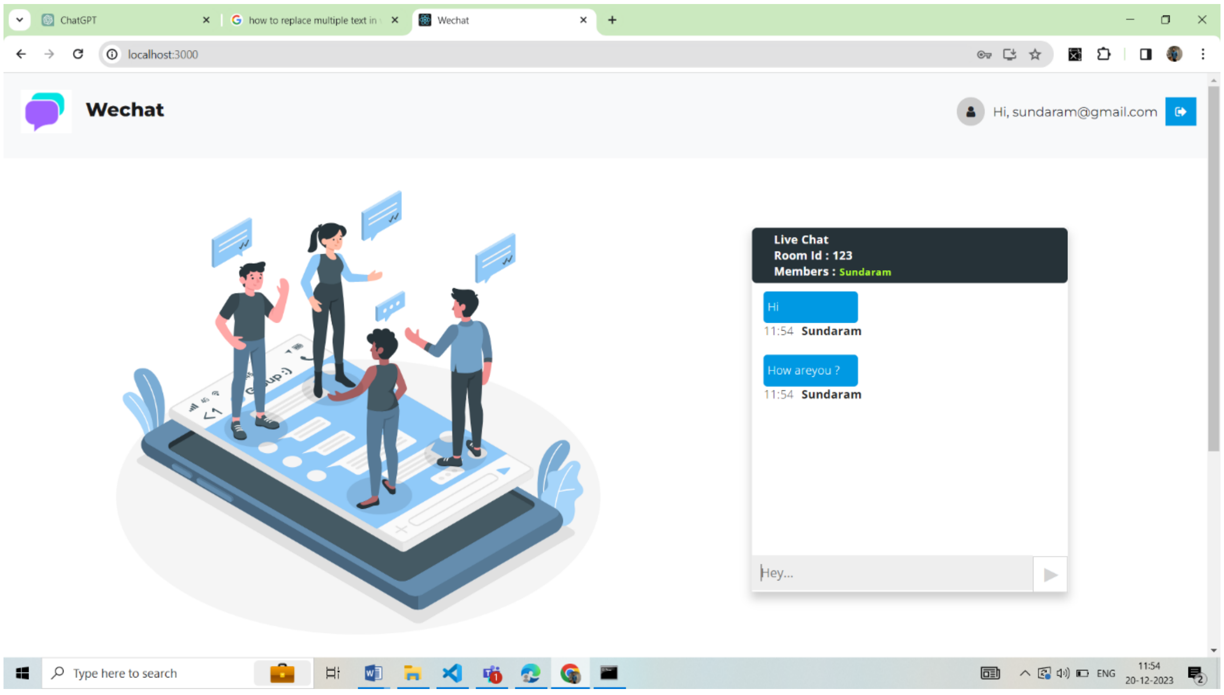


Join A Chat

Sundaram

123

JOIN A ROOM

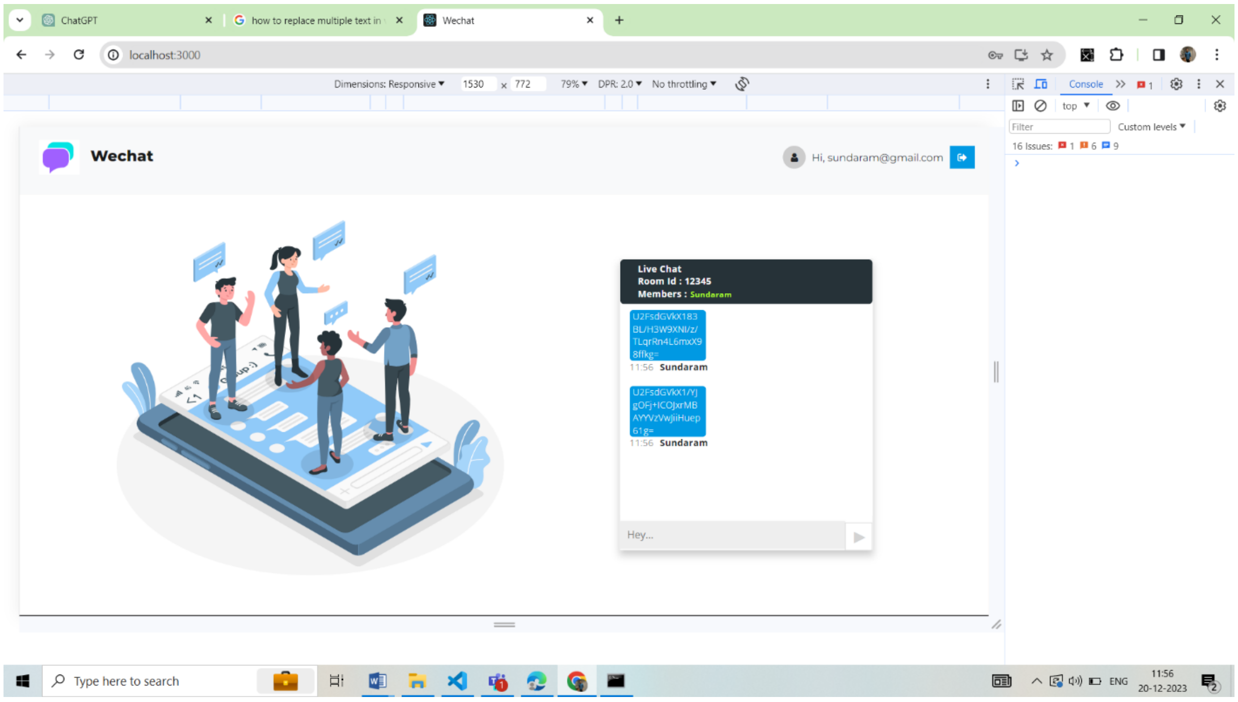


Live Chat
Room Id : 123
Members : Sundaram

Hi
11:54 Sundaram

How areyou ?
11:54 Sundaram

Hey...

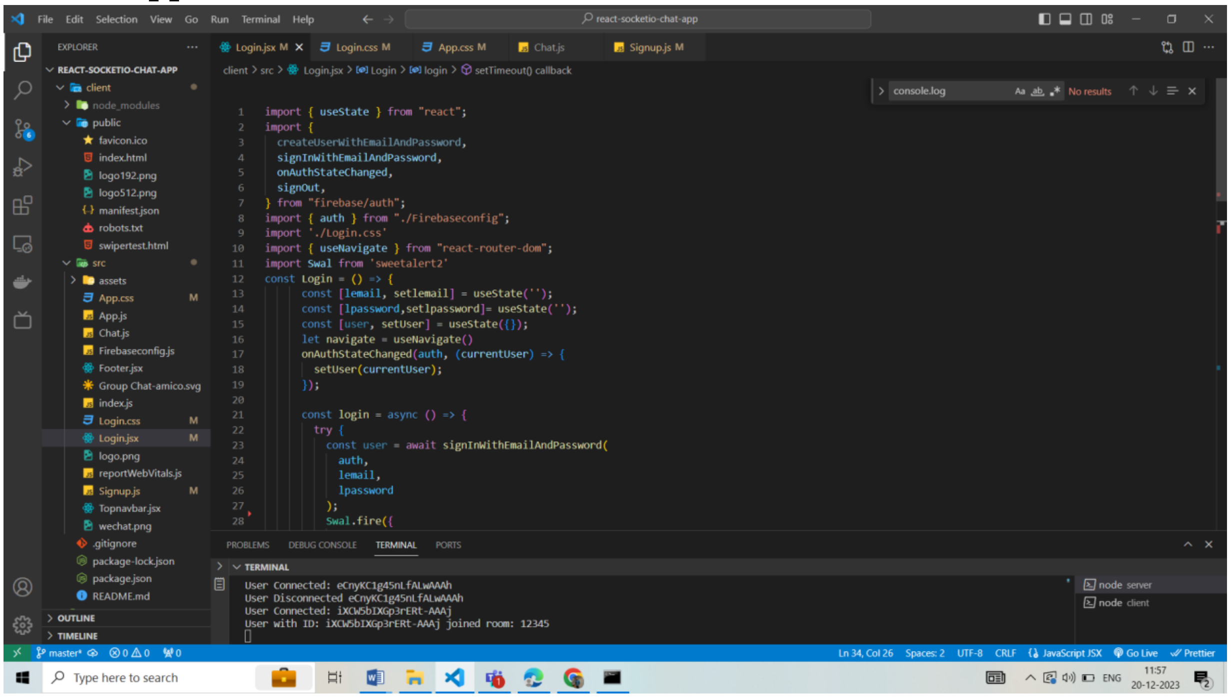


Live Chat
Room Id : 12345
Members : Sundaram

U2FsdGVkX183BL/H3W9XNI/z/TLqrRn4L6mxX98ffkg=
11:56 Sundaram

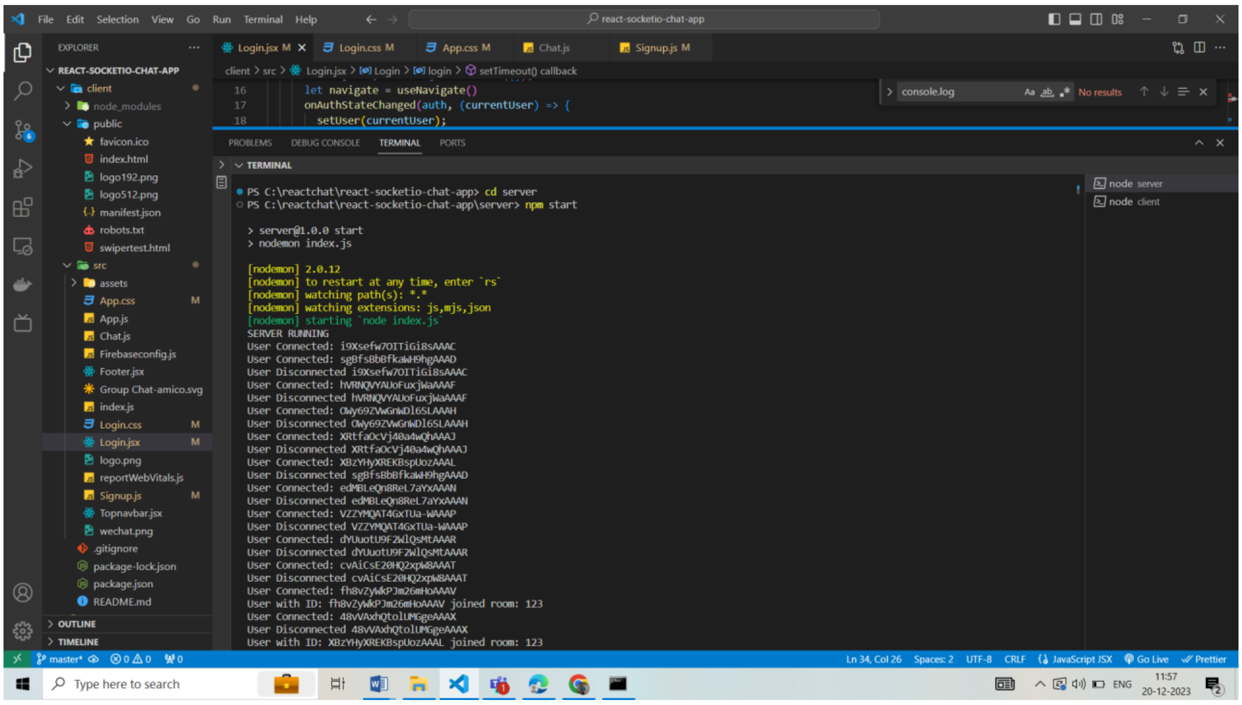
U2FsdGVkX1/YJgOFj+ICOJxrMBAYYVzVwJiiHuep61g=
11:56 Sundaram

C. Code Snippets



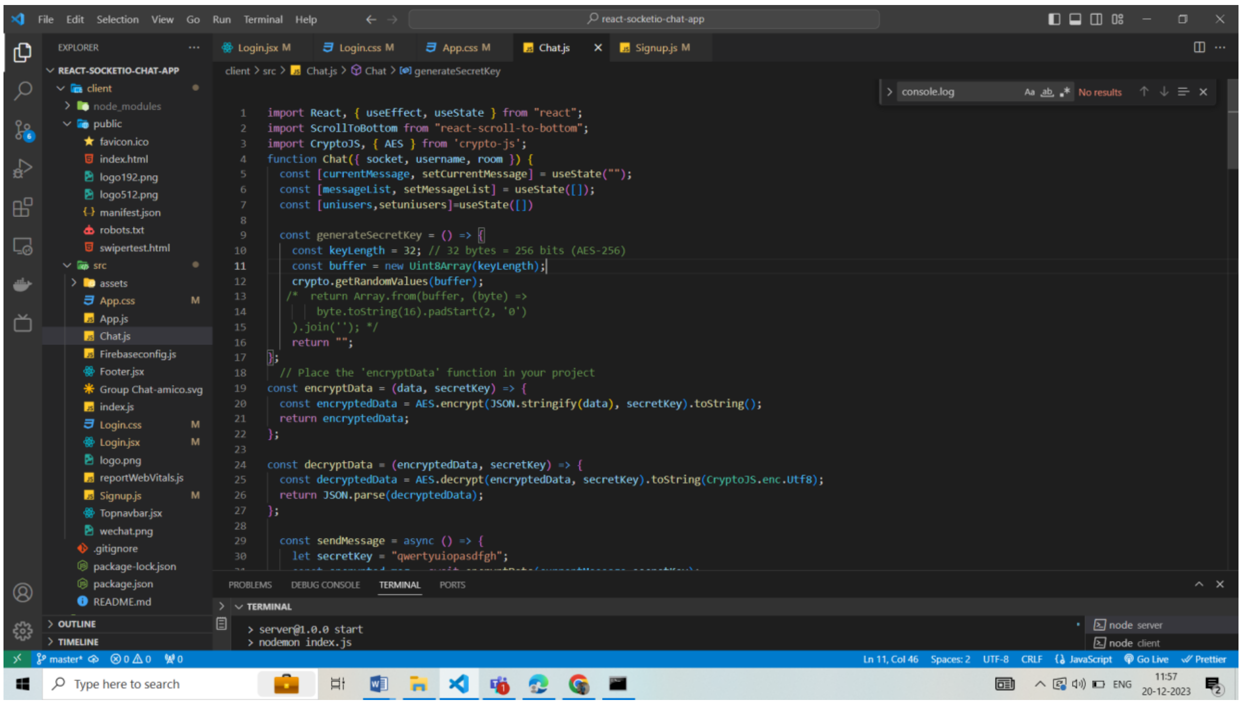
```
import { useState } from "react";
import {
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  onAuthStateChanged,
  signOut,
} from "firebase/auth";
import { auth } from "./Firebaseconfig";
import './Login.css'
import { useNavigate } from "react-router-dom";
import Swal from 'sweetalert2'
const Login = () => {
      const [lemail, setlemail] = useState('');
      const [lpassword,setlpassword]= useState('');
      const [user, setUser] = useState({});
      let navigate = useNavigate()
      onAuthStateChanged(auth, (currentUser) => {
        setUser(currentUser);
      });

      const login = async () => {
        try {
          const user = await signInWithEmailAndPassword(
            auth,
            lemail,
            lpassword
          );
          Swal.fire({
```

```
User Connected: eCnyKC1g45nLfALwAAAh
User Disconnected eCnyKC1g45nLfALwAAAh
User Connected: iXCW5bIXGp3rERt-AAAj
User with ID: iXCW5bIXGp3rERt-AAAj joined room: 12345
```



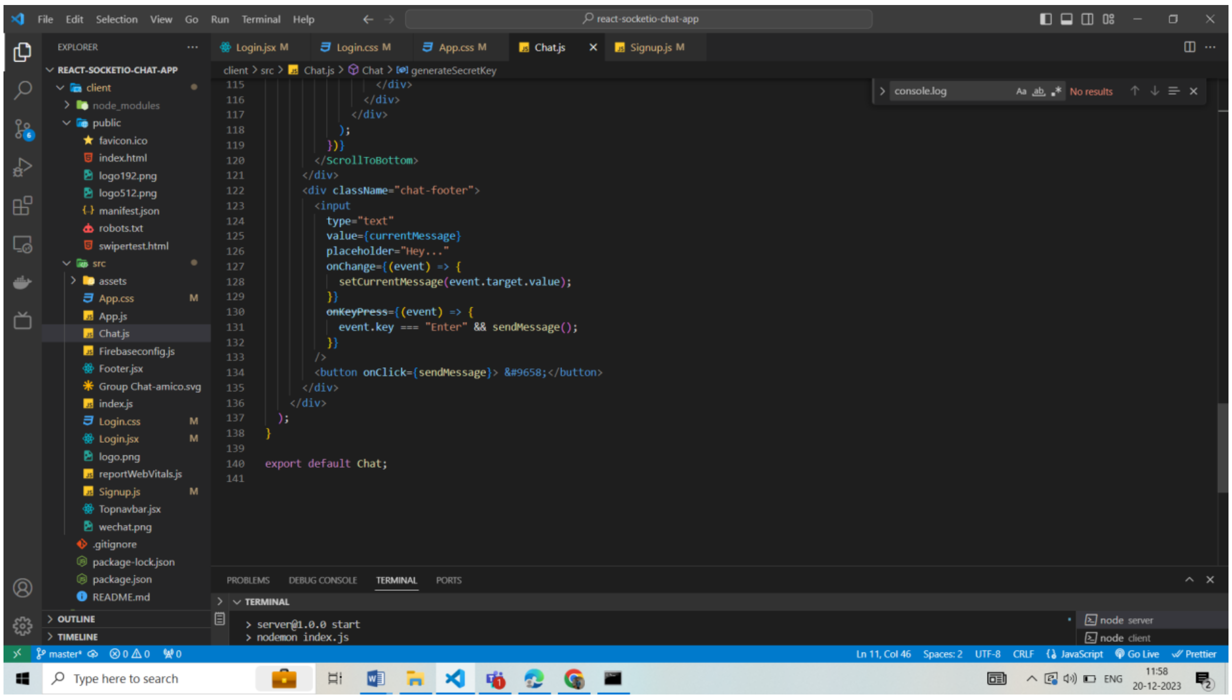
```
    let navigate = useNavigate()
    onAuthStateChanged(auth, (currentUser) => {
      setUser(currentUser);
```

```
PS C:\reactchat\react-socketio-chat-app> cd server
PS C:\reactchat\react-socketio-chat-app\server> npm start

 > server@1.0.0 start
 > nodemon index.js

[nodemon] 2.0.12
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
SERVER RUNNING
User Connected: i9Xsefw7OITiGi8sAAAC
User Connected: sgBfsBbBfkaWH9hgAAAD
User Disconnected i9Xsefw7OITiGi8sAAAC
User Connected: hVRNQVYAUoFuxjWaAAAF
User Disconnected hVRNQVYAUoFuxjWaAAAF
User Connected: OWy69ZVwGnWDl6SLAAAH
User Disconnected OWy69ZVwGnWDl6SLAAAH
User Connected: XRtfaOcVj40a4wQhAAAJ
User Disconnected XRtfaOcVj40a4wQhAAAJ
User Connected: XBzYHyXREKBspUozAAAL
User Disconnected sgBfsBbBfkaWH9hgAAAD
User Connected: edMBLeQn8ReL7aYxAAAN
User Disconnected edMBLeQn8ReL7aYxAAAN
User Connected: VZZYMQAT4GxTUa-WAAAP
User Disconnected VZZYMQAT4GxTUa-WAAAP
User Connected: dYUuotU9F2WlQsMtAAAR
User Disconnected dYUuotU9F2WlQsMtAAAR
User Connected: cvAiCsE20HQ2xpW8AAAT
User Disconnected cvAiCsE20HQ2xpW8AAAT
User Connected: fh8vZyWkPJm26mHoAAAV
User with ID: fh8vZyWkPJm26mHoAAAV joined room: 123
User Connected: 48vVAxhQtolUMGgeAAAX
User Disconnected 48vVAxhQtolUMGgeAAAX
User with ID: XBzYHyXREKBspUozAAAL joined room: 123
```



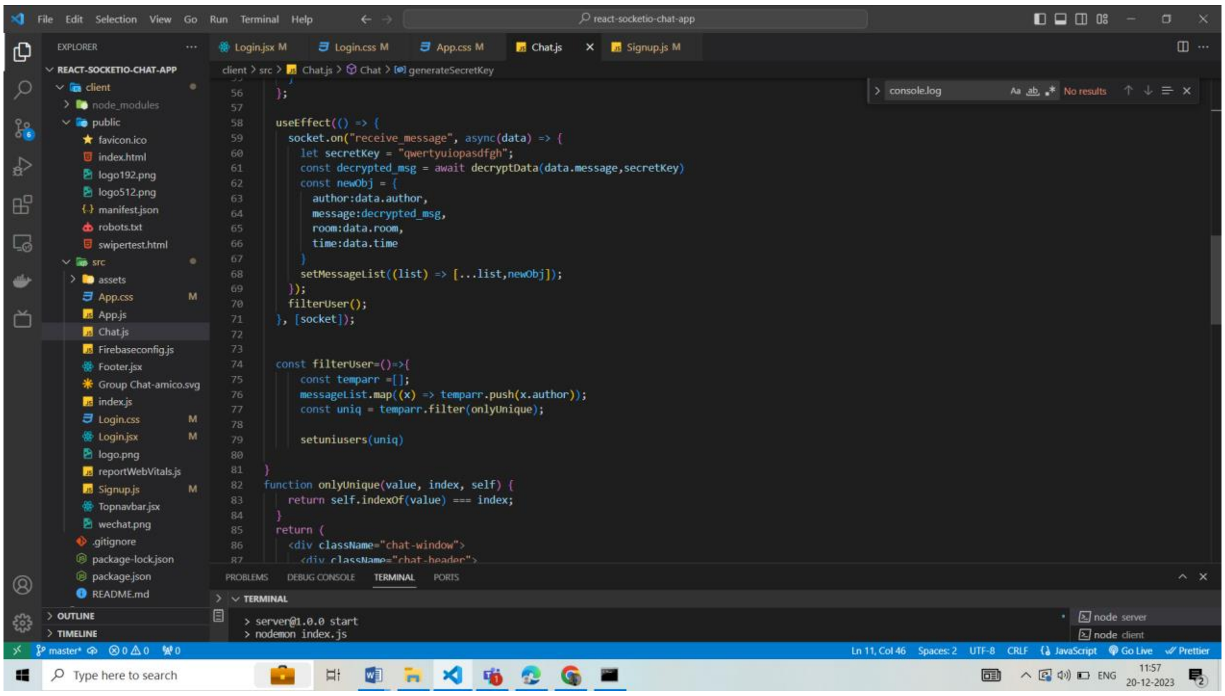
```
import React, { useEffect, useState } from "react";
import ScrollToBottom from "react-scroll-to-bottom";
import CryptoJS, { AES } from 'crypto-js';
function Chat({ socket, username, room }) {
  const [currentMessage, setCurrentMessage] = useState("");
  const [messageList, setMessageList] = useState([]);
  const [uniusers,setuniusers]=useState([])

  const generateSecretKey = () => {
    const keyLength = 32; // 32 bytes = 256 bits (AES-256)
    const buffer = new Uint8Array(keyLength);
    crypto.getRandomValues(buffer);
   /*  return Array.from(buffer, (byte) =>
        byte.toString(16).padStart(2, '0')
    ).join(''); */
    return "";
};
  // Place the 'encryptData' function in your project
const encryptData = (data, secretKey) => {
  const encryptedData = AES.encrypt(JSON.stringify(data), secretKey).toString();
  return encryptedData;
};

const decryptData = (encryptedData, secretKey) => {
  const decryptedData = AES.decrypt(encryptedData, secretKey).toString(CryptoJS.enc.Utf8);
  return JSON.parse(decryptedData);
};

  const sendMessage = async () => {
    let secretKey = "qwertyuiopasdfgh";
```

```
 > server@1.0.0 start
 > nodemon index.js
```



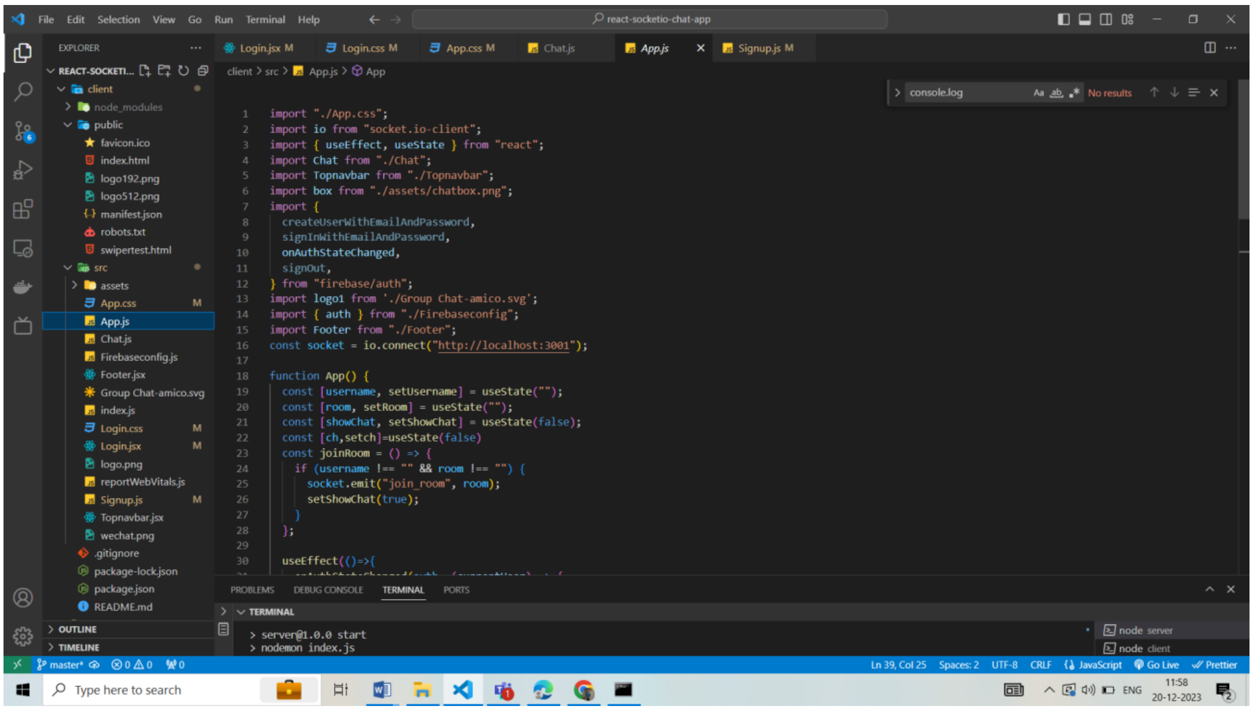
```
                    </div>
                  </div>
                </div>
              );
            })}
          </ScrollToBottom>
        </div>
        <div className="chat-footer">
          <input
            type="text"
            value={currentMessage}
            placeholder="Hey..."
            onChange={(event) => {
              setCurrentMessage(event.target.value);
            }}
            onKeyPress={(event) => {
              event.key === "Enter" && sendMessage();
            }}
          />
          <button onClick={sendMessage}> &#9658;</button>
        </div>
      </div>
    );
  }

  export default Chat;
```

```
> server@1.0.0 start
> nodemon index.js
```



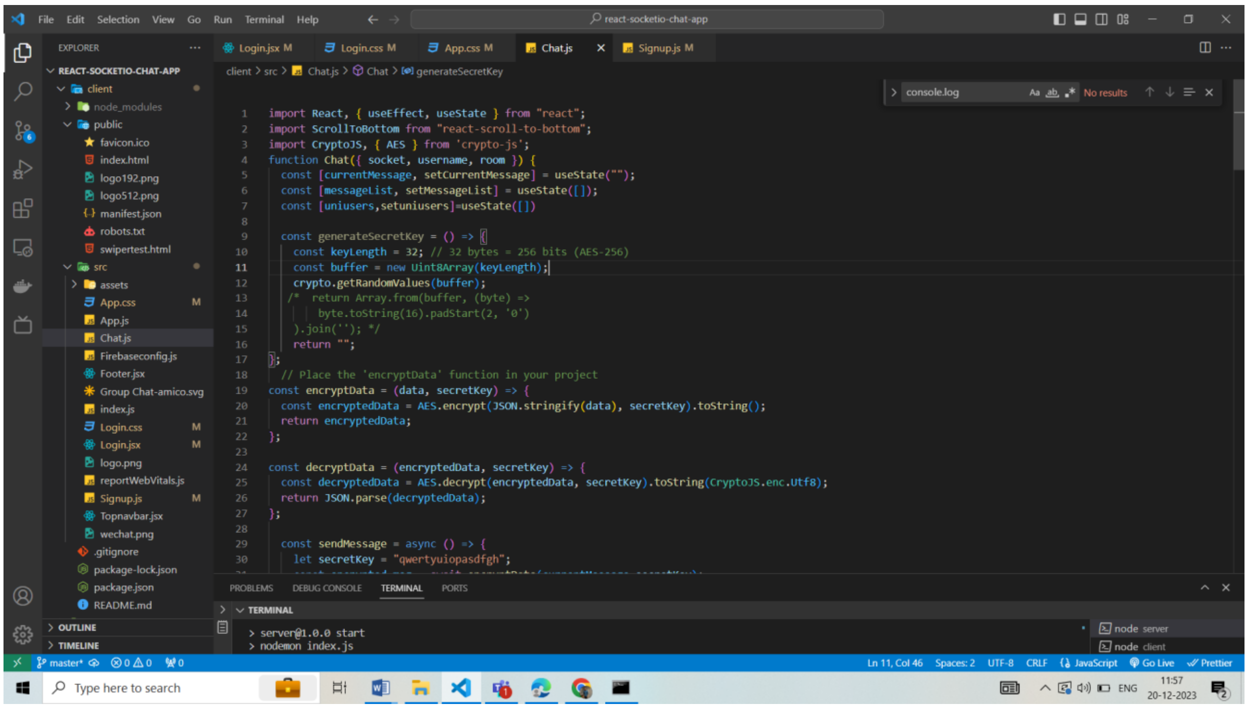
```
  };

  useEffect(() => {
    socket.on("receive_message", async(data) => {
      let secretKey = "qwertyuiopasdfgh";
      const decrypted_msg = await decryptData(data.message,secretKey)
      const newObj = {
        author:data.author,
        message:decrypted_msg,
        room:data.room,
        time:data.time
      }
      setMessageList((list) => [...list,newObj]);
    });
    filterUser();
  }, [socket]);


  const filterUser=()=>{
      const temparr =[];
      messageList.map((x) => temparr.push(x.author));
      const uniq = temparr.filter(onlyUnique);

      setuniusers(uniq)

}
function onlyUnique(value, index, self) {
    return self.indexOf(value) === index;
  }
  return (
    <div className="chat-window">
      <div className="chat-header">
```

```
> server@1.0.0 start
> nodemon index.js
```



```
import "./App.css";
import io from "socket.io-client";
import { useEffect, useState } from "react";
import Chat from "./Chat";
import Topnavbar from "./Topnavbar";
import box from "./assets/chatbox.png";
import {
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  onAuthStateChanged,
  signOut,
} from "firebase/auth";
import logo1 from './Group Chat-amico.svg';
import { auth } from "./Firebaseconfig";
import Footer from "./Footer";
const socket = io.connect("http://localhost:3001");

function App() {
  const [username, setUsername] = useState("");
  const [room, setRoom] = useState("");
  const [showChat, setShowChat] = useState(false);
  const [ch,setch]=useState(false)
  const joinRoom = () => {
    if (username !== "" && room !== "") {
      socket.emit("join_room", room);
      setShowChat(true);
    }
  };

  useEffect(()=>{
```

```
> server@1.0.0 start
> nodemon index.js
```



```
import React, { useEffect, useState } from "react";
import ScrollToBottom from "react-scroll-to-bottom";
import CryptoJS, { AES } from 'crypto-js';
function Chat({ socket, username, room }) {
  const [currentMessage, setCurrentMessage] = useState("");
  const [messageList, setMessageList] = useState([]);
  const [uniusers,setuniusers]=useState([])

  const generateSecretKey = () => {
    const keyLength = 32; // 32 bytes = 256 bits (AES-256)
    const buffer = new Uint8Array(keyLength);
    crypto.getRandomValues(buffer);
   /*  return Array.from(buffer, (byte) =>
      byte.toString(16).padStart(2, '0')
    ).join(''); */
    return "";
};
  // Place the 'encryptData' function in your project
const encryptData = (data, secretKey) => {
  const encryptedData = AES.encrypt(JSON.stringify(data), secretKey).toString();
  return encryptedData;
};

const decryptData = (encryptedData, secretKey) => {
  const decryptedData = AES.decrypt(encryptedData, secretKey).toString(CryptoJS.enc.Utf8);
  return JSON.parse(decryptedData);
};

  const sendMessage = async () => {
    let secretKey = "qwertyuiopasdfgh";
```

```
> server@1.0.0 start
> nodemon index.js
```

MODULE 6

6.Conclusion

Conclusion

Limitations

Future Scope

A. Conclusion :

The project aimed to address the challenges associated with password security and management by incorporating advanced cryptographic techniques. The system provides a secure and user-friendly environment for storing and managing passwords while leveraging arithmetic encoding for enhanced encryption. Throughout the development lifecycle, rigorous testing and quality assurance measures were implemented to ensure the robustness and reliability of the system. The project adhered to the specified requirements and timelines, resulting in a functional and well-documented password vault. The user authentication module, password management module, backup and recovery module, user interface module, and security module were integrated seamlessly to provide a comprehensive solution. The use of arithmetic encoding adds an extra layer of security to the stored passwords, enhancing the confidentiality and integrity of user data.

B. Limitations:

While encryption methods, including arithmetic encoding, are implemented for password security, no system is entirely risk-free. Constant vigilance and updates are necessary to mitigate emerging security threats. If a user forgets their master password recovery options may be limited, potentially resulting in the loss of stored passwords. The centralization of password storage in one system creates a single point of failure. A breach or failure in the system could have significant consequences. Users may have concerns about the privacy and security of their stored passwords, particularly if the system faces a security breach or unauthorized access.

C. Future Scope Password Expiry Notifications:

• Provide users with notifications for password expiration, encouraging them to update passwords regularly and adhere to security best practices. Enhanced Backup Options:
• Expand backup and recovery capabilities by incorporating cloud-based solutions, allowing users to sync and access their password vaults across multiple devices. Cross-Platform Compatibility:
• Develop dedicated mobile applications for iOS and Android platforms, ensuring users can securely access their password vaults on various devices.