COLLEGE CODE : 9111

COLLEGE NAME: SRM MADURAI COLLEGE FOR ENGINEERING AND TECHNOLOGY

DEPARTMENT: B.E COMPUTER SCIENCE AND ENGINEERING

STUDENT NM-ID: 29693809323E58D85804E77C620A3B33
E84A706084E253582B34A98D05CA3B22
5783594E2D19B3FA18F47F4C526F12C7
1069BEA60F6B24E85854A583833EB1C6
5EAED41CE1AB1D89AC442AA0A8A642F7

ROLL NO: 911123104021
911123104003
911123104049
911123104035
911123104042

DATE: 06.10.2025

Completed the project named as Phase_V_

TECHNOLOGY PROJECT NAME : CHAT APPLICATION UI

SUBMITTED BY,

NAME : C.AKSHAYA
H.JESIMA YOHAANA
S.SHREYA MERCY
D.PRIYA DHARSHINI
M.K.ROSHINI

CHAT APPLICATION UI—PHASE 5 1. Final Demo Walkthrough

Application Name: Chat App Prototype with Status & Emojis

Features Demonstrated:

1. Inbox / Chat List – Displays all existing chats with user avatars, last messages, timestamps, unread counts, and online/offline status indicators.
2. Real-Time Messaging Simulation – Messages from other users appear automatically every 15 seconds.
3. Chat Conversation Screen – View and send messages with:
    - Message status indicators: Sent ☑, Delivered ☑☑, Seen
    - 👁 Reply functionality (click on a message to reply to it)
    - Emojis integration with an emoji picker
4. Profile Management – View user profile details and navigate to settings or chats.
5. Settings Screen – Basic navigation options like Change Password, Notifications, Privacy, About.
6. Compose New Chat – Search and start a new chat with predefined users.
7. Notifications – Browser notifications for incoming messages if permission is granted.

Walkthrough Flow:

1. Open the app → Inbox screen is visible with all chats.
2. Click on any chat → Opens conversation screen.
3. Type a message → Send → Message appears with “Sent” status, then updates to “Delivered” and “Seen.”

4. Click a message → Reply functionality works.
5. Click emoji → Emoji picker opens → Add emoji to messages.
6. Navigate to Profile/Settings → Access personal details and settings.
7. Start a new chat → Select a user from the compose screen → Open chat with them.

---

2. Project Report

Objective:
To build a simple, interactive chat application prototype that simulates real-time messaging, status tracking, reply functionality, and emoji usage in a single-page interface.

Technologies Used:

- HTML5 – Structure of the app
- CSS3 – Styling, flexbox layout, dark/light themes
- JavaScript – Functionality for messaging, status updates, emoji picker, notifications

Core Components:

1. Inbox/Chat List – Display chat metadata and allow navigation.
2. Chat Screen – Render messages, handle replies, message statuses, emojis.
3. Profile & Settings – User details and app settings navigation.
4. Compose Chat – Start new conversations.
5. Notification System – Simulated incoming messages and browser notifications.
6. Emoji Picker – Add emojis to messages.

---

3. Screenshots / API Documentation

Screenshots:

1. Inbox View: Shows all active chats with unread messages and online/offline status.
2. Chat Conversation: Message bubbles for both sender and receiver, reply feature, emoji picker.
3. Profile Screen: User avatar, name, location, navigation menu.
4. Compose New Chat: Search and select users to start a chat.
5. Settings Screen: Navigation links to settings options.

API / Function Reference:

- `showScreen(id)` – Switches between screens.
- `renderChats()` – Displays all chat items in inbox.
- `openChat(chat)` – Opens a selected chat conversation.
- `renderMessages()` – Renders messages for the current chat.
- `addMessage(text, who, status, replyTo, store)` – Adds a message bubble to chat.
- `statusIcon(status)` – Returns emoji representing message status.
- `sendMessage()` – Sends a new message from input field.
- `updateLastMessageStatus(newStatus)` – Updates the status of the last sent message.
- `notify(chat, msg)` – Simulates incoming message notifications.
- `toggleEmojiPicker()` / `renderEmojiPicker()` – Opens/closes and renders emoji picker.
- `renderUsers()` – Displays users for starting new chats.

---

4. Challenges & Solutions

Challenge	Solution Implemented
Real-time message simulation	Used `setInterval` to simulate incoming messages every 15s.
Message status updates	Implemented `updateLastMessageStatus` with `setTimeout` for Sent → Delivered → Seen.
Reply functionality	Stored `replyTo` property for each message and updated placeholder on click.

Emoji integration	Created dynamic emoji picker and appended selected emojis to input field.
Challenge	Solution Implemented
Browser notifications	Requested notification permission and triggered notifications on message arrival.
Scroll auto-update	Used `scrollTop = scrollHeight` after each message to keep the view at the bottom.

---

5. GitHub README & Setup Guide

README Example:

```
# Chat App Prototype

A simple chat app prototype with messaging, status, emoji support, and simulated notifications.

## Features
- Inbox with chat list and online/offline status
- Real-time simulated messages
- Message status: Sent, Delivered, Seen
- Reply functionality
- Emoji picker
- Profile and settings screens
- Compose new chat

## Setup
1. Clone the repository  git clone <repo-url>
2. Open `index.html` in your browser.
3. Allow notifications when prompted.
4. Start using the app.

## Screenshots
(Add screenshots in the repo)

## Technologies
- HTML5
- CSS3
- JavaScript
```

CODE:

```
<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>ChatZone 💬</title>
<style>
*{margin:0;padding:0;box-sizing:border-box;font-family:'Poppins',sans-serif;}
body{background:lineargradient(135deg,#6a11cb,#2575fc);height:100vh;display:flex;justifycontent:center;align-items:center;}
.app{width:900px;height:600px;background:white;borderradius:15px;overflow:hidden;display:flex;boxshadow:0 8px 25px rgba(0,0,0,0.25);}

/* Sidebar */
.sidebar{width:35%;background:#f7f7f7;border-right:1px solid #ddd;display:flex;flex-direction:column;} .sidebar-header{background:linear-gradient(45deg,#2575fc,#6a11cb);color:white;padding:15px;textalign:center;font-weight:bold;font-size:20px;}
.chat-list{flex:1;overflow-y:auto;}
.chat-item{display:flex;align-items:center;padding:10px;border-bottom:1px solid #ddd;cursor:pointer;transition:0.3s;}
.chat-item:hover{background:#e6e6fa;}
.chat-item img{width:45px;height:45px;border-radius:50%;margin-right:10px;}
.chat-info h4{font-size:16px;color:#333;}
.chat-info p{font-size:13px;color:#777;}

/* Chat window */
.chat-window{flex:1;display:flex;flex-direction:column;background:#ece5dd;}
.chat-header{background:linear-gradient(45deg,#2575fc,#6a11cb);color:white;padding:15px;display:flex;alignitems:center;}
.chat-header img{width:40px;height:40px;border-radius:50%;margin-right:10px;}
.chat-header h3{font-size:18px;}
.chat-box{flex:1;padding:15px;overflow-y:auto;display:flex;flexdirection:column;}
.message{max-width:70%;padding:10px 15px;margin:6px 0;border-radius:15px;fontsize:15px;lineheight:1.4;animation:fadeIn 0.3s ease;word-wrap:break-word;}
.sent{background:#dcf8c6;align-self:flex-end;border-bottom-right-radius:3px;}
.received{background:white;align-self:flex-start;border-bottom-leftradius:3px;} .timestamp{font-size:10px;color:#999;text-align:right;margin-top:2px;}
.chat-input{display:flex;padding:10px;background:#f0f0f0;border-top:1px solid #ccc;}
.chat-input input{flex:1;padding:10px 15px;border:none;borderradius:25px;outline:none;font-size:14px;}
.chat-input button{background:#2575fc;color:white;border:none;padding:10px 14px;border-radius:50%;margin-left:8px;cursor:pointer;fontsize:18px;transition:0.3s;}
.chat-input button:hover{background:#6a11cb;}
.active{background:#e6e6fa;}
```

```
@keyframes
fadeIn{from{opacity:0;transform:translateY(10px);}to{opacity:1;transform:trans lateY(0);}}
</style>
</head>
<body>

<div class="app">
  <!-- Sidebar -->
  <div class="sidebar">
    <div class="sidebar-header">ChatZone 💬</div>
    <div class="chat-list" id="chatList">
      <div class="chat-item active" onclick="openChat('Emma')"><img
src="https://i.pravatar.cc/100?img=1"><div class="chatinfo"><h4>Emma</h4><p>Hey!
Long time 😊</p></div>
      </div>
      <div class="chat-item" onclick="openChat('John')">
        <img src="https://i.pravatar.cc/100?img=5"><div class="chatinfo"><h4>John</h4><p>See you
tomorrow?</p></div>
      </div>
      <div class="chat-item" onclick="openChat('Sophie')">
        <img src="https://i.pravatar.cc/100?img=12"><div class="chatinfo"><h4>Sophie</h4><p>Got your
message 💌</p></div>
      </div>
    </div>
  </div>

  <!-- Chat Window -->
  <div class="chat-window">
    <div class="chat-header">
      <img id="chatImg" src="https://i.pravatar.cc/100?img=1">
      <h3 id="chatName">Emma</h3>
    </div>
    <div class="chat-box" id="chatBox">
      <div class="message received">Hey there! 👋 How have you been?<div
class="timestamp">10:00</div></div>
    </div>
    <div class="chat-input">
      <input id="userInput" type="text" placeholder="Type a message..."
onkeypress="handleKeyPress(event)">
      <button onclick="sendMessage()">➤</button>
    </div>
  </div>
</div>
```

```
<script>
```

```
const chatData = {
"Emma": {
```

```
  img: "https://i.pravatar.cc/100?img=1",    chats: [{sender:'Emma',text:'Hey there! 👋 How have you been?',time:'10:00'}]
 },  "John": {    img: "https://i.pravatar.cc/100?img=5",    chats: [{sender:'John',text:'Yo! Ready for tomorrow's plan?',time:'09:45'}]
 },
 "Sophie": {    img: "https://i.pravatar.cc/100?img=12",    chats: [{sender:'Sophie',text:'Got your message 💌 Can we talk later?',time:'11:15'}]
 }
}; let currentChat = "Emma"; const chatBox = document.getElementById('chatBox'); const userInput = document.getElementById('userInput');  function openChat(name){  currentChat = name; document.getElementById('chatName').innerText = name; document.getElementById('chatImg').src = chatData[name].img; document.querySelectorAll('.chat-item').forEach(i=>i.classList.remove('active'));
 [...document.querySelectorAll('.chatitem')].find(i=>i.innerText.includes(name)).classList.add('active');

 chatBox.innerHTML = '';  chatData[name].chats.forEach(msg=>{    const div = document.createElement('div');    div.classList.add('message', msg.sender === name ? 'received':'sent');    div.innerHTML = `${msg.text}<div class="timestamp">${msg.time}</div>`;    chatBox.appendChild(div);
 });
 chatBox.scrollTop = chatBox.scrollHeight;
}  function sendMessage(){    const text = userInput.value.trim();  if(!text) return;  const time = new Date().toLocaleTimeString([], {hour:'2-digit', minute:'2digit'});  const msgDiv = document.createElement('div');
```

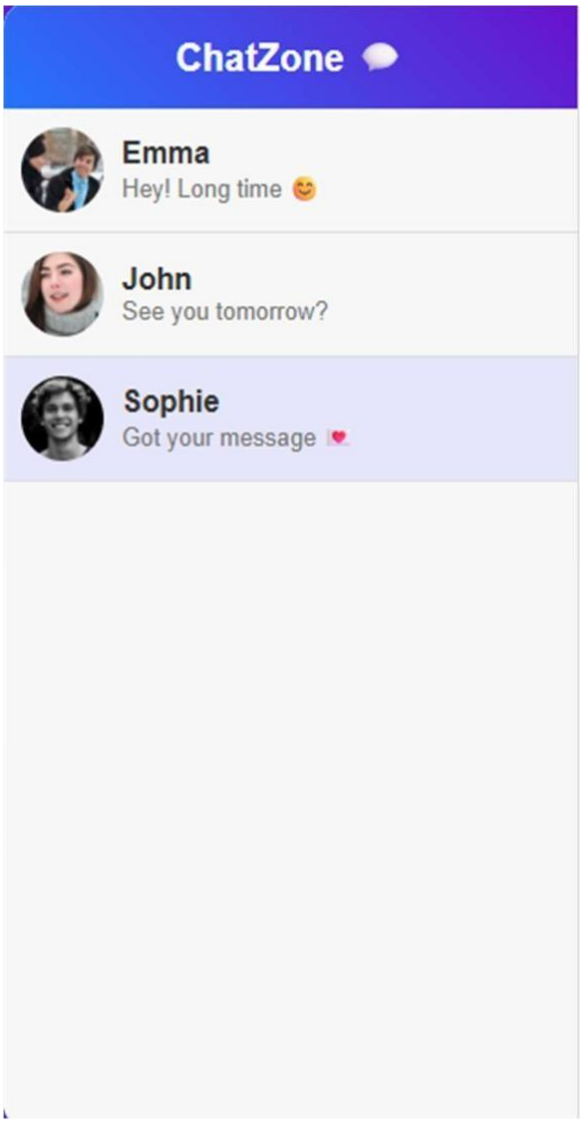
```
  msgDiv.classList.add('message','sent');    msgDiv.innerHTML = `${text}<div
class="timestamp">${time}</div>`;
```

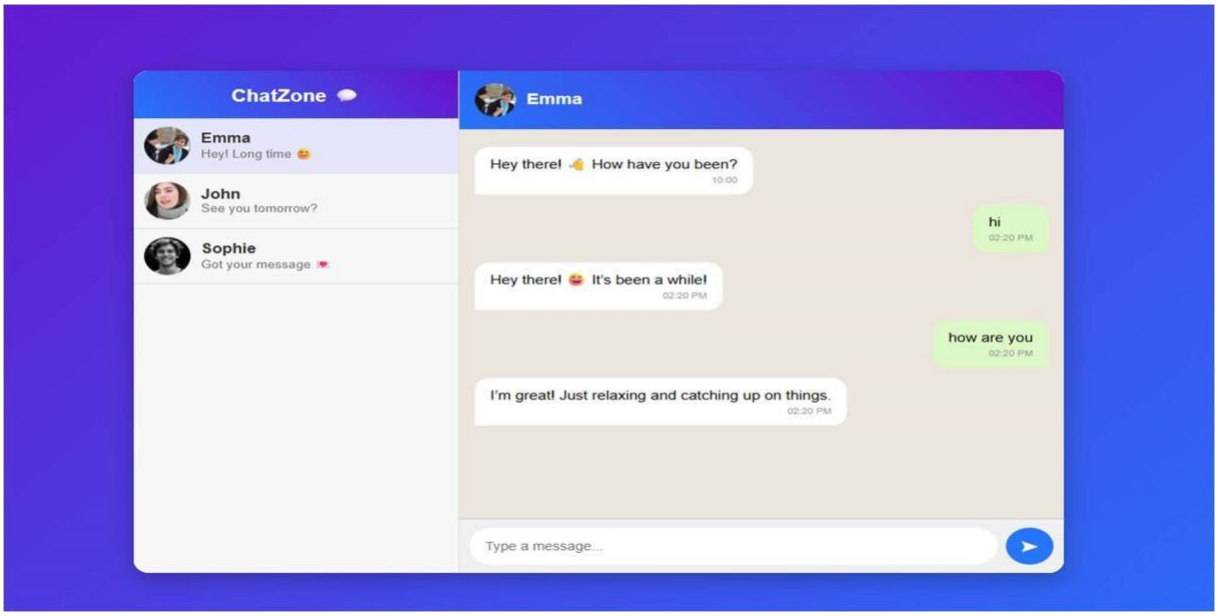
```
  chatBox.appendChild(msgDiv);          chatBox.scrollTop    =
chatBox.scrollHeight;
chatData[currentChat].chats.push({sender:'You',text,time});
userInput.value                                =                                 '';
setTimeout(()=>aiReply(currentChat,text),1000);
}                        function             handleKeyPress(e){
if(e.key===='Enter') sendMessage();
}
function aiReply(person,userText){   userText = userText.toLowerCase();   let reply =
"";     if(userText.includes("hi")||userText.includes("hello")){     reply = `Hey there! 😄
It's been a while!`;   } else if(userText.includes("how are you")){     reply = `I'm great!
Just relaxing and catching up on things.`;   } else
if(userText.includes("plan")||userText.includes("meet")){     reply = `Sounds good!
Let's plan something soon 🤌 `;
  } else if(userText.includes("bye")){     reply = `See you soon!
Take care 💖`;
  } else {      const
replies = [
     "Haha, that's interesting 😄",
     "Oh really? Tell me more!",
     "Wow, I didn't expect that!",
     "I totally agree with you  ",
     "That's awesome!"
    ];          reply  =  replies[Math.floor(Math.random()*replies.length)];       }         const  time  =  new
Date().toLocaleTimeString([],                                {hour:'2-digit',                                minute:'2digit'});
chatData[person].chats.push({sender:person,text:reply,time});     if(person === currentChat){      const
msgDiv     =     document.createElement('div');                    msgDiv.classList.add('message','received');
msgDiv.innerHTML = `${reply}<div
class="timestamp">${time}</div>`;     chatBox.appendChild(msgDiv);     chatBox.scrollTop =
chatBox.scrollHeight;
```

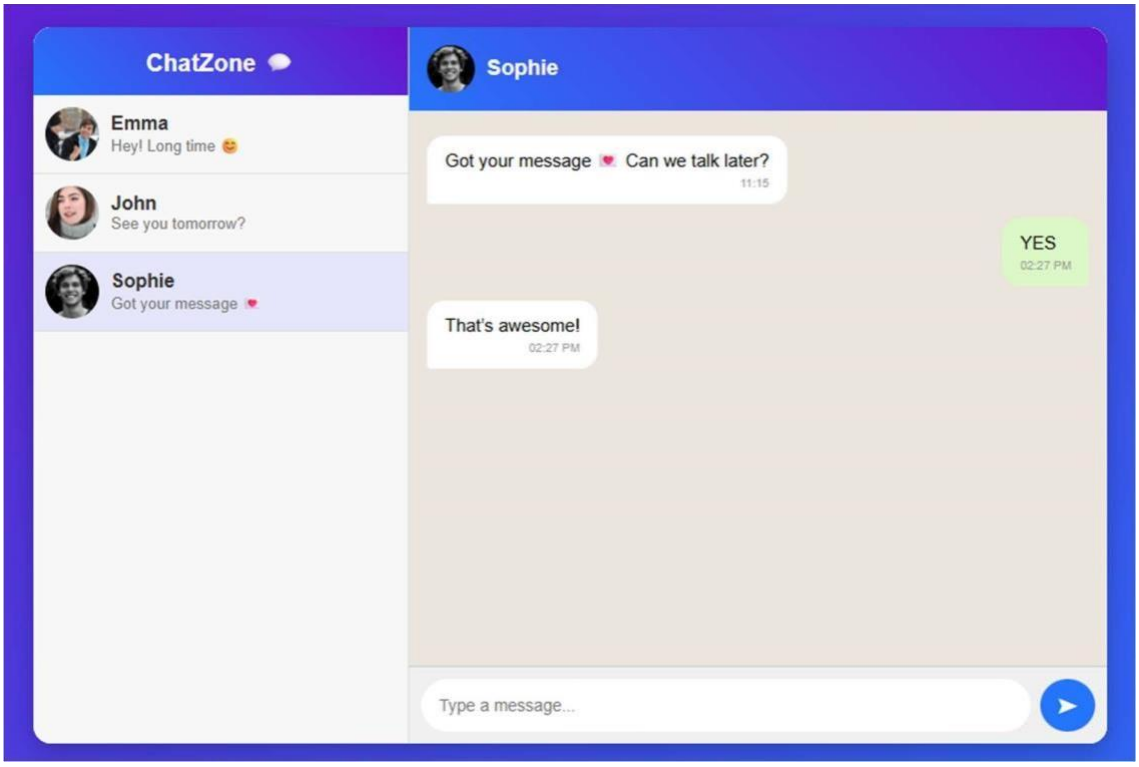
```
}
}
</script>

</body>
</html>
```

OUTPUT IMAGES:




ChatZone
Emma
Hey! Long time
John
See you tomorrow?
Sophie
Got your message
Emma
Hey there! How have you been?
10:00
hi
02:20 PM
Hey there! It's been a while!
02:20 PM
how are you
02:20 PM
I'm great! Just relaxing and catching up on things.
02:20 PM
Type a message...



Github Links:

Shreyamercy S- https://github.com/sm1874/nm-phases-of-the-projects

Jesima Yohaana H- https://github.com/jesijesi150515/Nm-project.git

Priyadharshini D-
https://github.com/priyadharshinidhanasekar03/Nm-phases

Akshaya C- https://github.com/akshaya21st-blip/NM-phases

Roshini M K- https://github.com/roshini177/SCT_WD_4