A black background with text and a picture of a person

AI-generated content may be incorrect. A black and white logo

AI-generated content may be incorrect.

**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**

**Completed the project named as Phase\_IV\_ 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 IV**

**1. Additional Features**

* Identify pending or “nice-to-have” features from earlier phases.
* Example:
  + Search and filter options
  + User profile customization
  + Notifications or alerts
  + Export/download data functionality
  + Role-based access control

**2. UI/UX Improvements**

* Refine the user interface for better usability.
* Example tasks:
  + Improve responsiveness across devices.
  + Enhance navigation (menus, breadcrumbs, quick actions).
  + Add animations, tooltips, or micro-interactions.
  + Follow accessibility standards (contrast, alt-text, ARIA roles).

**3. API Enhancements**

* Optimize existing APIs for better reliability and speed.
* Example:
  + Add pagination, filtering, and sorting to APIs.
  + Strengthen error handling and validation.
  + Improve API documentation (Swagger/Postman).
  + Secure APIs with authentication (JWT/OAuth).

**4. Performance & Security Checks**

* **Performance Testing:**
  + Optimize queries and database operations.
  + Use caching (Redis, browser cache, CDN).
  + Reduce bundle size and loading time.
* **Security Testing:**
  + Input sanitization to prevent SQL injection/XSS.
  + HTTPS and secure cookies.
  + Role-based access permissions.
  + Vulnerability scanning tools.

**5. Testing of Enhancements**

* Unit testing of new features.
* Integration and regression testing.
* User Acceptance Testing (UAT).
* Automated testing (if possible).

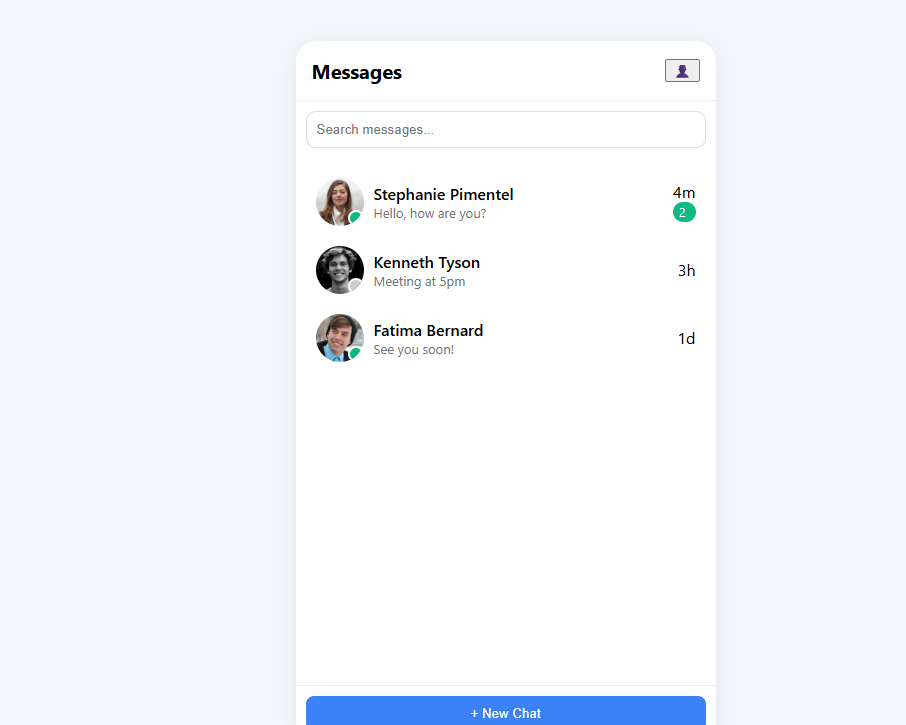
**6. Deployment**

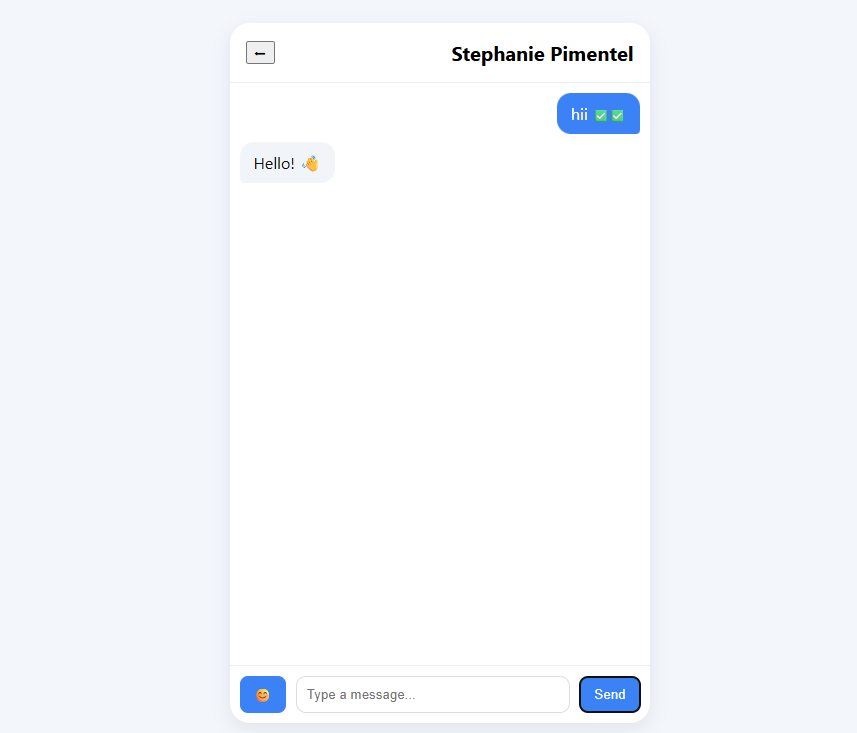
* Choose hosting platform:
  + **Netlify** → Best for frontend React/Vue apps.
  + **Vercel** → Great for Next.js/React with CI/CD.
  + **Cloud Platforms (AWS/GCP/Azure)** → For full-stack apps.
* Set up CI/CD pipelines for automated deployment.
* Configure environment variables securely.
* Monitor logs and performance after deployment.

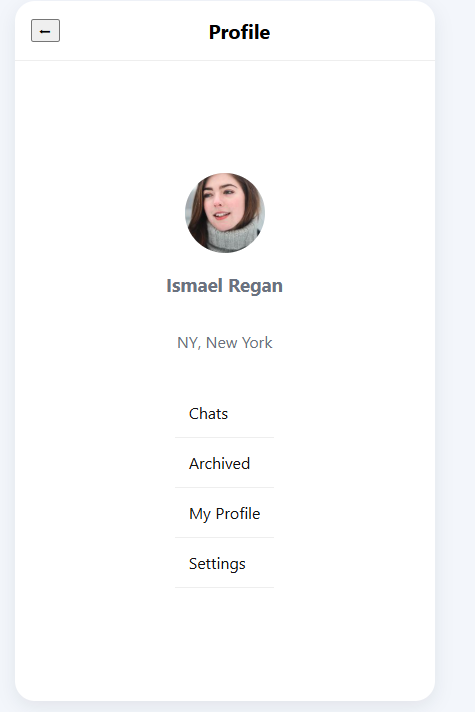
**Code On Chat Application UI**

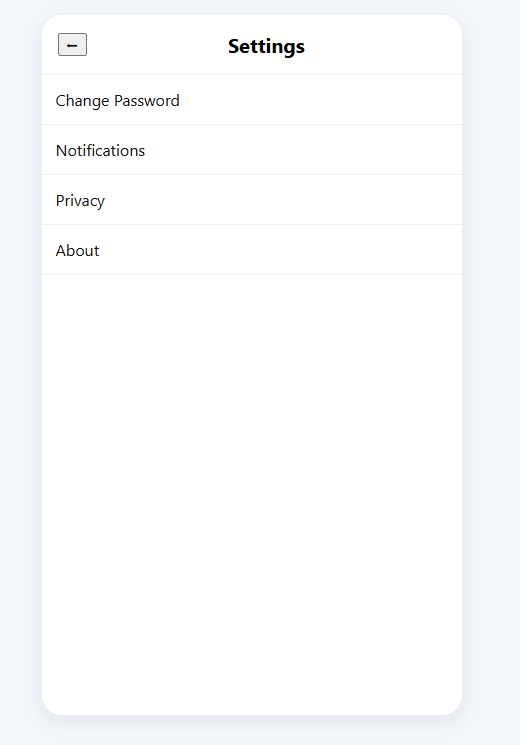
<!doctype html>  
<html lang="en">  
<head>  
<meta charset="utf-8">  
<meta name="viewport" content="width=device-width,initial-scale=1">  
<title>WhatsApp-Style Chat App</title>  
<style>  
:root{  
  --bg:#f3f6fb; --card:#ffffff; --muted:#6b7280; --accent:#3b82f6;  
  --soft:#f1f5f9; --shadow:0 6px 18px rgba(15,23,42,0.08);  
  font-family:Inter,ui-sans-serif,system-ui,"Segoe UI",Roboto,Arial;  
}  
body{margin:0;min-height:100vh;background:var(--bg);display:flex;align-items:center;justify-content:center;padding:20px}  
.app{width:100%;max-width:420px;min-height:700px;background:var(--card);border-radius:20px;box-shadow:var(--shadow);overflow:hidden;display:flex;flex-direction:column}  
header{padding:16px;display:flex;align-items:center;justify-content:space-between;border-bottom:1px solid #eee}  
header h2{margin:0;font-size:20px}  
button{cursor:pointer}  
.screen{flex:1;display:none;flex-direction:column}  
.screen.active{display:flex}  
.list{flex:1;overflow:auto;padding:10px}  
.chat-item{display:flex;gap:10px;padding:10px;border-radius:12px;cursor:pointer;align-items:center}  
.chat-item:hover{background:var(--soft)}  
.avatar{width:48px;height:48px;border-radius:50%;overflow:hidden;position:relative}  
.avatar img{width:100%;height:100%;object-fit:cover}  
.online-status{width:12px;height:12px;border-radius:50%;position:absolute;bottom:0;right:0;border:2px solid #fff;}  
.online-status.online{background:#10b981}  
.online-status.offline{background:#ccc}  
.meta{flex:1;min-width:0}  
.meta .name{font-weight:600}  
.meta .msg{font-size:13px;color:var(--muted);white-space:nowrap;overflow:hidden;text-overflow:ellipsis}  
.meta .time{font-size:12px;color:var(--muted)}  
.unread{background:#10b981;color:white;border-radius:999px;padding:2px 6px;font-size:12px}  
input[type=text],input[type=search]{padding:10px;border-radius:10px;border:1px solid #ddd;width:100%}  
footer{padding:10px;border-top:1px solid #eee;display:flex;gap:10px;align-items:center}  
footer input{flex:1}  
footer button{background:var(--accent);color:white;border:none;padding:10px 14px;border-radius:8px}  
.message-list{flex:1;overflow:auto;padding:10px;display:flex;flex-direction:column;gap:8px;position:relative}  
.bubble{max-width:70%;padding:10px 14px;border-radius:14px;position:relative;word-break:break-word;}  
.[bubble.me](http://bubble.me/){align-self:flex-end;background:var(--accent);color:white;border-bottom-right-radius:4px}  
.bubble.them{align-self:flex-start;background:var(--soft);color:#111;border-bottom-left-radius:4px}  
.menu a{display:block;padding:14px;border-bottom:1px solid #eee;text-decoration:none;color:#111}  
.menu a:hover{background:var(--soft)}  
.center{flex:1;display:flex;align-items:center;justify-content:center;flex-direction:column;color:var(--muted)}  
.reply-text{font-size:12px;color:#555;border-left:2px solid #ccc;padding-left:6px;margin-bottom:4px}  
.status-icon{font-size:12px;margin-left:5px;}  
.emoji-picker{display:flex;flex-wrap:wrap;gap:5px;padding:5px;background:#eee;border-radius:10px;max-height:120px;overflow:auto;margin-top:5px;}  
.emoji-picker span{cursor:pointer;font-size:20px;}  
.typing-indicator{font-size:12px;color:#555;padding:4px;margin-left:10px;}  
</style>  
</head>  
<body>  
<div class="app">  
  
  <!-- Inbox -->  
  <div id="inbox" class="screen active">  
    <header>  
      <h2>Messages</h2>  
      <button onclick="showScreen('profile')"></button>  
    </header>  
    <div style="padding:10px"><input type="search" id="search" placeholder="Search messages..."></div>  
    <div class="list" id="chatList"></div>  
    <footer><button style="flex:1" onclick="showScreen('compose')">+ New Chat</button></footer>  
  </div>  
  
  <!-- Chat Conversation -->  
  <div id="chat" class="screen">  
    <header>  
      <button onclick="showScreen('inbox')">⬅</button>  
      <h2 id="chatTitle">Chat</h2>  
      <span id="typingIndicator" class="typing-indicator" style="display:none;">typing...</span>  
    </header>  
    <div class="message-list" id="messages"></div>  
    <footer>  
      <button onclick="toggleEmojiPicker()">😊</button>  
      <input type="text" id="msgInput" placeholder="Type a message...">  
      <button onclick="sendMessage()">Send</button>  
    </footer>  
    <div class="emoji-picker" id="emojiPicker" style="display:none;"></div>  
  </div>  
  
</div>  
  
<script>  
// Chat data  
const chats = [  
  {id:1,name:"Stephanie Pimentel",msg:"Hello, how are you?",avatar:"<https://i.pravatar.cc/150?img=32>",time:"4m",unread:2, online:true, messages:[]},  
  {id:2,name:"Kenneth Tyson",msg:"Meeting at 5pm",avatar:"<https://i.pravatar.cc/150?img=12>",time:"3h",unread:0, online:false, messages:[]},  
  {id:3,name:"Fatima Bernard",msg:"See you soon!",avatar:"<https://i.pravatar.cc/150?img=8>",time:"1d",unread:0, online:true, messages:[]}  
];  
  
let currentChat = null;  
const emojiList = ["😀","😂","😍","😎","😭","😡","👍","🙏","🔥","💯","🎉","❤️"];  
const typingIndicator = document.getElementById('typingIndicator');  
  
function showScreen(id){  
  document.querySelectorAll('.screen').forEach(s=>s.classList.remove('active'));  
  document.getElementById(id).classList.add('active');  
  if(id==="inbox") renderChats();  
  if(id==="chat" && currentChat) renderMessages();  
}  
  
// Render chat list  
function renderChats(){  
  const list=document.getElementById('chatList');  
  list.innerHTML="";  
  chats.forEach(c=>{  
    const div=document.createElement('div');  
    div.className="chat-item";  
    div.innerHTML=`  
      <div class="avatar">  
        <img src="${c.avatar}">  
        <div class="online-status ${c.online ? 'online':'offline'}"></div>  
      </div>  
      <div class="meta">  
        <div class="name">${[c.name](http://c.name/)}</div>  
        <div class="msg">${c.msg}</div>  
      </div>  
      <div>  
        <div class="time">${c.time}</div>  
        ${c.unread?`<div class="unread">${c.unread}</div>`:""}  
      </div>  
    `;  
    div.onclick=()=>openChat(c);  
    list.appendChild(div);  
  });  
}  
  
// Open chat  
function openChat(chat){  
  currentChat = chat;  
  document.getElementById('chatTitle').textContent = [chat.name](http://chat.name/);  
  showScreen('chat');  
  renderMessages();  
}  
  
// Render messages  
function renderMessages(){  
  const container = document.getElementById('messages');  
  container.innerHTML = "";  
  currentChat.messages.forEach(msg => addMessage(msg.text, msg.who, msg.status, msg.replyTo, false));  
  container.scrollTop = container.scrollHeight;  
}  
  
// Add message  
function addMessage(text, who, status="sent", replyTo=null, store=true){  
  const container = document.getElementById('messages');  
  const div = document.createElement('div');  
  div.className = "bubble " + who;  
  if(replyTo){  
    const replyDiv = document.createElement('div');  
    replyDiv.className = "reply-text";  
    replyDiv.textContent = "Replying to: " + replyTo;  
    div.appendChild(replyDiv);  
  }  
  div.appendChild(document.createTextNode(text));  
  
  if(who==="me"){  
    const statusSpan = document.createElement('span');  
    statusSpan.className="status-icon";  
    statusSpan.textContent = statusIcon(status);  
    div.appendChild(statusSpan);  
  }  
  
  container.appendChild(div);  
  container.scrollTop = container.scrollHeight;  
  if(store) currentChat.messages.push({text, who, status, replyTo});  
}  
  
// Status icons  
function statusIcon(status){  
  switch(status){  
    case "sent": return "✅";  
    case "delivered": return "✅✅";  
    case "seen": return "👁";  
    default: return "";  
  }  
}  
  
// Send message  
function sendMessage(){  
  const input=document.getElementById('msgInput');  
  if(input.value.trim()!==""){  
    const replyTo = input.dataset.replyTo || null;  
    addMessage(input.value,'me','sent',replyTo);  
    input.value = "";  
    input.placeholder = "Type a message...";  
    input.dataset.replyTo = "";  
    // simulate delivery & seen after scrolling to bottom  
    setTimeout(()=>updateLastMessageStatus('delivered'), 1000);  
    setTimeout(()=>updateLastMessageStatus('seen'), 3000);  
    simulateTyping();  
  }  
}  
  
// Update last message status  
function updateLastMessageStatus(newStatus){  
  const msgs = currentChat.messages;  
  if(msgs.length>0){  
    msgs[msgs.length-1].status = newStatus;  
    renderMessages();  
  }  
}  
  
// Reply on message click  
document.getElementById('messages').addEventListener('click', e=>{  
  if(e.target.classList.contains('bubble')){  
    const text = e.target.textContent.replace(/✅|👁/g,'').trim();  
    document.getElementById('msgInput').placeholder = "Replying to: " + text;  
    document.getElementById('msgInput').dataset.replyTo = text;  
  }  
});  
  
// Emoji picker  
function toggleEmojiPicker(){  
  const picker = document.getElementById('emojiPicker');  
  picker.style.display = picker.style.display==='none' ? 'flex' : 'none';  
  if(picker.style.display==='flex') renderEmojiPicker();  
}  
  
function renderEmojiPicker(){  
  const picker = document.getElementById('emojiPicker');  
  picker.innerHTML = "";  
  emojiList.forEach(e=>{  
    const span = document.createElement('span');  
    span.textContent = e;  
    span.onclick = ()=>{  
      const input = document.getElementById('msgInput');  
      input.value += e;  
      input.focus();  
    };  
    picker.appendChild(span);  
  });  
}  
  
// Simulate typing indicator  
function simulateTyping(){  
  typingIndicator.style.display = "inline";  
  setTimeout(()=>{  
    typingIndicator.style.display = "none";  
    // simulate incoming reply  
    addMessage("Hello! 👋",'them','seen');  
  }, 2000);  
}  
  
// Initialize  
renderChats();  
</script>  
</body>  
</html>

**Output**

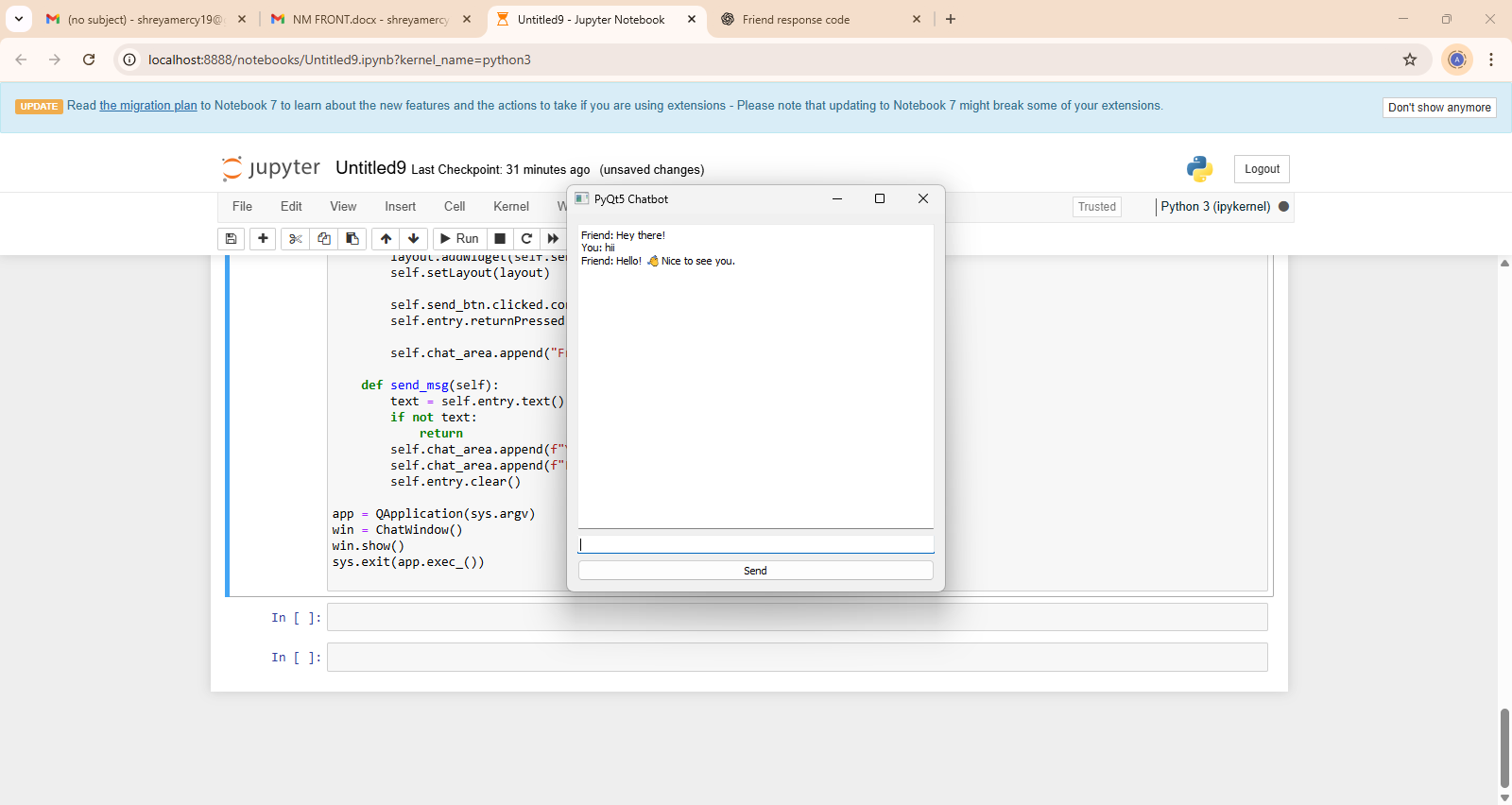


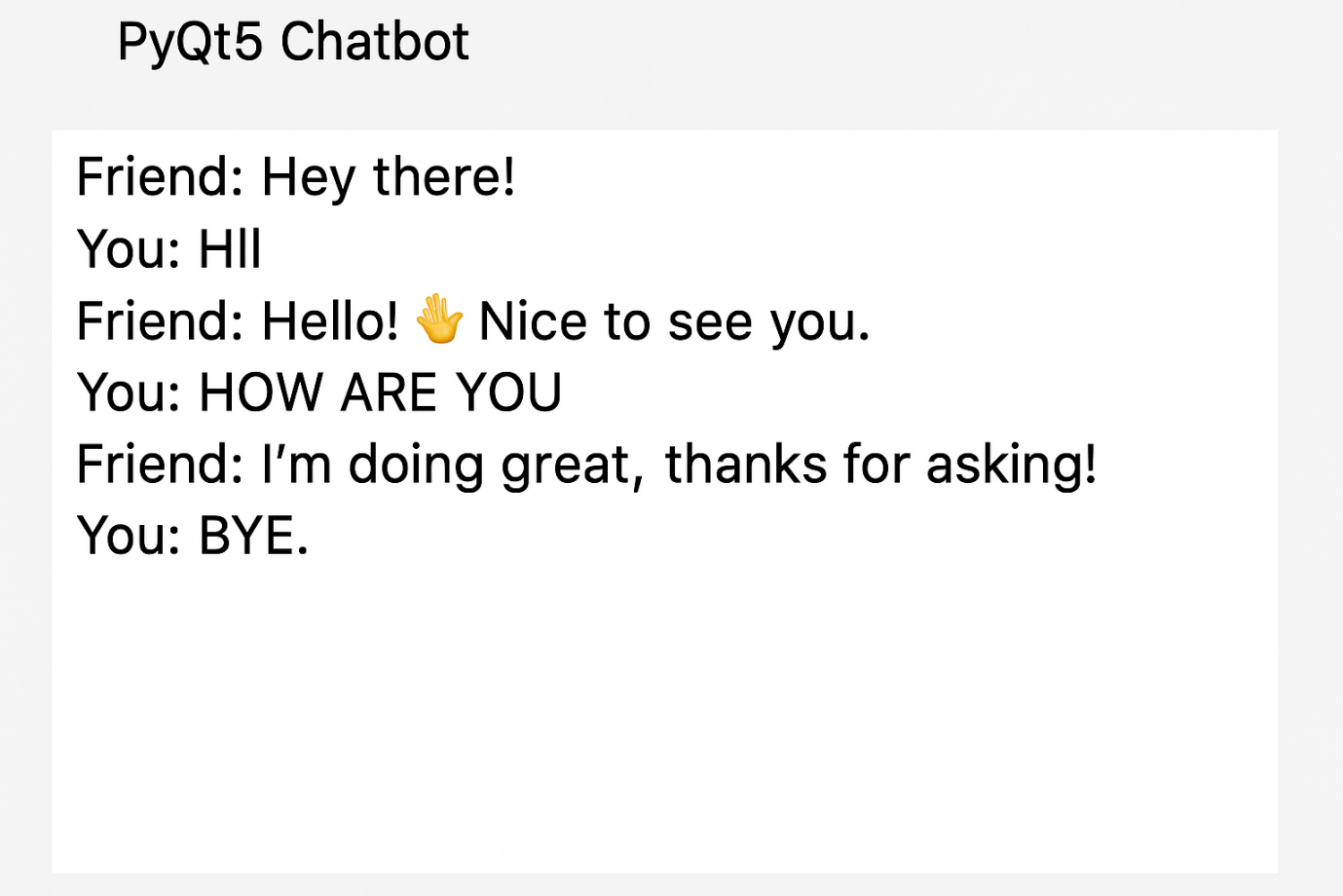




****

**Output of Chat Application UI**

****

****

**Github Link:** [**https://github.com/sm1874/nm-project/blob/main/nm.html**](https://github.com/sm1874/nm-project/blob/main/nm.html)