COLLEGE CODE : 9604

COLLEGE NAME : C.S.I INSTITUTE OF TECHNOLOGY

DEPARTMENT : ARTIFICIAL INTELLIGENCE & DATA SCIENCE

STUDENT NM ID: 32958ED1F49116FAB51418481C37DC5C

ROLL NO : 960423243003

DATE : 07/10/2025

SUBMITTED BY,

NAME :AKASH NATRAJ T

MOBILE NO:7358909324

PHASE 5 – PROJECT DEMONSTRATION & DOCUMENTATION
INTERACTIVE QUIZ APP

FINAL DEMO WALKTHROUGH :

A final demo walkthrough for a chat application UI typically involves showcasing the core interface features and how users interact with the app. The demo should highlight the chat list screen (chat index), the chat conversation screen with message bubbles, user presence indicators, input fields, sending/receiving messages, and any additional UI elements like file previews or typing indicators .

Key Components to Showcase in Final Demo Walkthrough :

Chat Index Screen: Display the list of conversations with previews of the last message, timestamps, unread message counts, and online status indicators. Include search functionality and quick actions like mute or delete chat. This is the inbox where users choose which conversation to open .The start screen or cover with a quiz title.

Suggested Demo Flow :

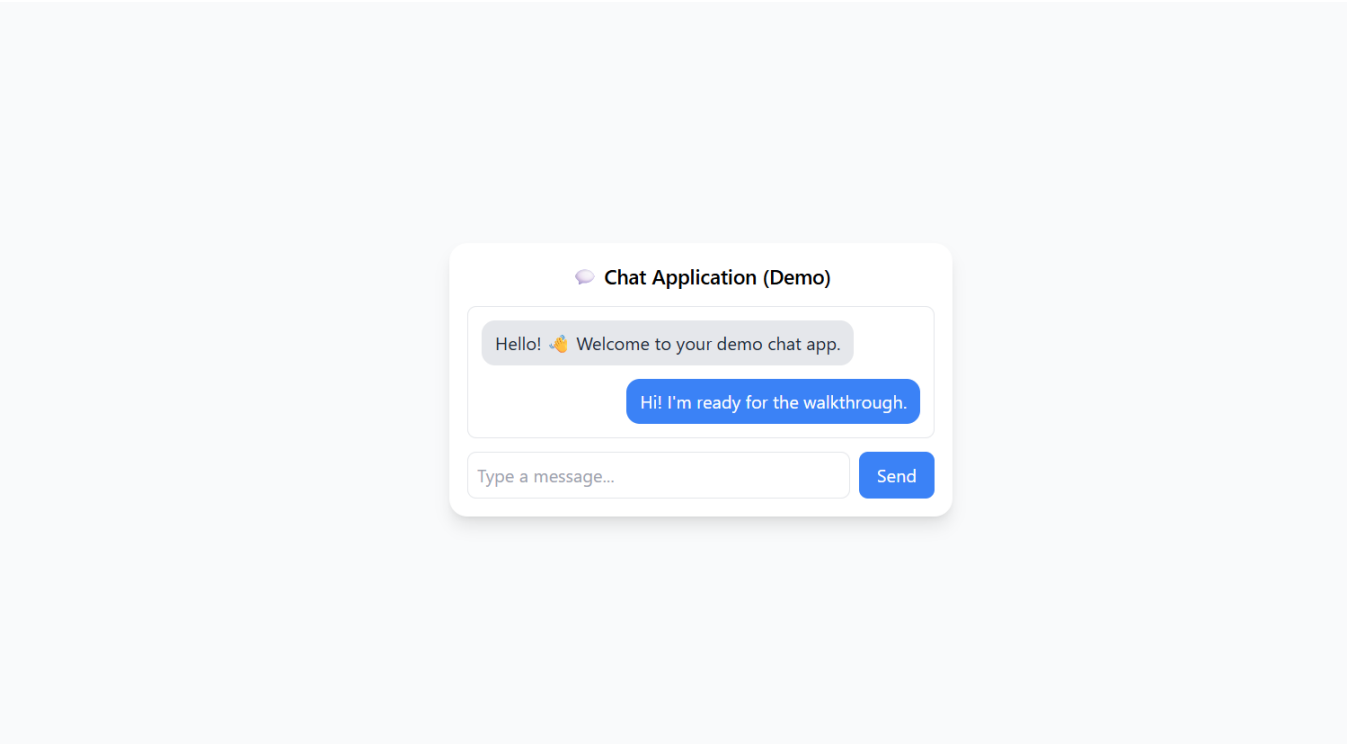
- Open the app and show the chat index screen.
- Use the search feature to find a particular chat.
- Tap on a conversation to open chat screen.
- Show a live message exchange: typing, sending text, images, or files.
- Highlight message status indicators like read receipts.
- Navigate back to chat list and show quick actions like long-press to delete or mute.
- Show presence indicators and user profile interactions.
- End with user-friendly UI elements like smooth animations and responsive layouts.

CODE :

```
import React, { useState } from "react";

export default function SimpleChatApp() {
  const [messages, setMessages] = useState([
    { from: "bot", text: "Hello! 👋 Welcome to your demo chat app." },
    { from: "me", text: "Hi! I'm ready for the walkthrough." },
  ]);
  const [input, setInput] = useState("");

  const sendMessage = () => {
    if (!input.trim()) return;
    const newMsg = { from: "me", text: input };
    setMessages([...messages, newMsg]);

    setTimeout(() => {
      const reply = { from: "bot", text: "Got it 👍 Let's continue!" };
      setMessages((prev) => [...prev, reply]);
    }, 1000);

    setInput("");
  };

  return (
    <div className="min-h-screen bg-gray-50 flex flex-col items-center justify-center p-
6">
      <div className="w-full max-w-md bg-white shadow-lg rounded-2xl p-4 flex flex-col">
        <h2 className="text-lg font-semibold text-center mb-3">
          💬 Chat Application (Demo)
        </h2>

        <div className="flex-1 overflow-y-auto mb-3 space-y-3 h-64 border p-3 rounded-
lg">
          {messages.map((msg, i) => (
            <div
              key={i}
              className={`flex ${
                msg.from === "me" ? "justify-end" : "justify-start"
```

```
          }`}
        >
          <div
            className={`px-3 py-2 rounded-xl ${
              msg.from === "me"
                ? "bg-blue-500 text-white"
                : "bg-gray-200 text-gray-800"
            }`}
          >
            {msg.text}
          </div>
        </div>
      ))}
    </div>

    <div className="flex gap-2">
      <input
        type="text"
        value={input}
        onChange={(e) => setInput(e.target.value)}
        onKeyDown={(e) => e.key === "Enter" && sendMessage()}
        className="flex-1 border rounded-lg p-2"
        placeholder="Type a message..."
      />
      <button
        onClick={sendMessage}
        className="bg-blue-500 text-white px-4 py-2 rounded-lg hover:bg-blue-600"
      >
        Send
      </button>
    </div>
  </div>
 </div>
);
}
```

OUTPUT :



PROJECT REPORT :

Title Page: Contains the project title, the author's name, institution, and submission date.

Abstract: Brief summary covering objectives, scope, key features, and outcomes.

Table of Contents: Organized list of chapters, diagrams, and appendices for easy navigatio

.

Project Overview :

- The interactive quiz app aims to provide a user-friendly platform for creating, managing, and taking quizzes online. It helps both educators and learners by offering immediate feedback, real-time scoring, and a simple interface that can be accessed on various devices.

Key Features :

- Real-time messaging and synchronization.
- Customizable user interface and theming.
- Security (authentication, encryption) measures.
- Mobile and desktop compatibility.
- Scalability, user management, and admin panel .

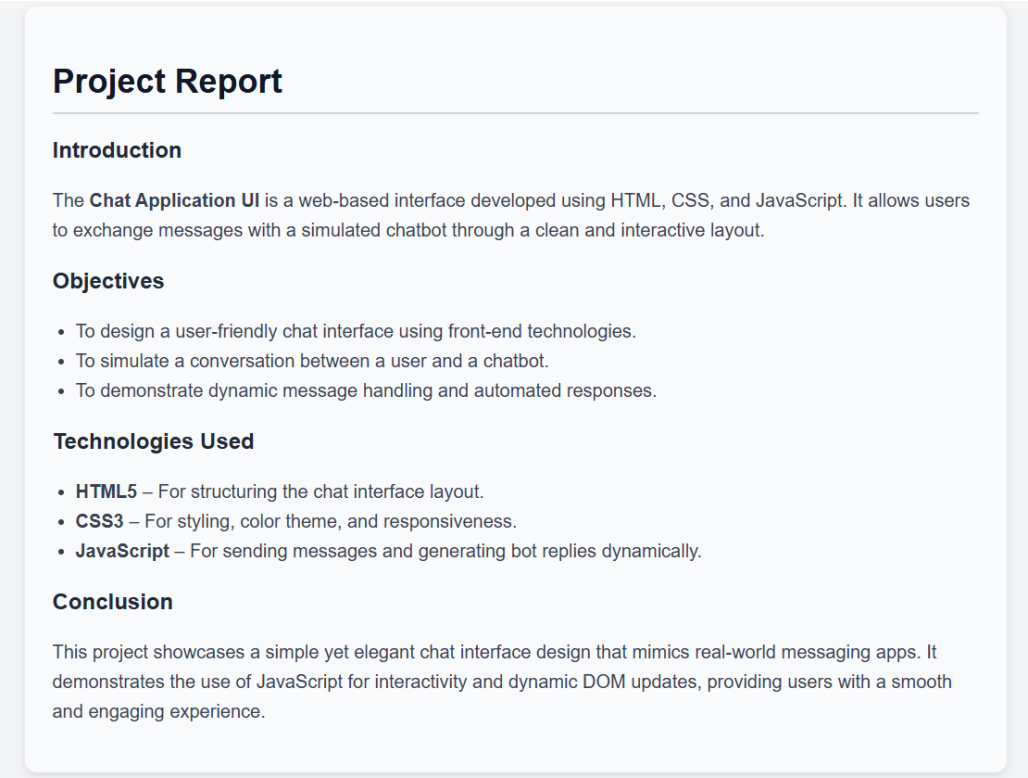
This structure ensures the report is comprehensive and faculty-ready, and helps guide readers through each stage of the project lifecycle, from design to deployment and testing

CODE :

```
<!-- Project Report - Chat Application UI -->
<section id="project-report" style="font-family: Arial, Helvetica, sans-serif; padding: 24px; max-width: 800px; margin: 24px auto; background-color: #f9fafb; border-radius: 10px; box-shadow: 0 2px 6px rgba(0,0,0,0.1);">

  <!-- Heading -->
  <h2 style="color: #111827; font-size: 1.8rem; margin-bottom: 10px;">Project Report</h2>
  <hr style="border: 1px solid #d1d5db; margin-bottom: 20px;">

  <!-- Introduction -->
  <h3 style="color: #1f2937; font-size: 1.2rem;">Introduction</h3>
  <p style="color: #374151; line-height: 1.6;">
    The <strong>Chat Application UI</strong> is a web-based interface developed using HTML, CSS, and JavaScript.
    It allows users to exchange messages with a simulated chatbot through a clean and interactive layout.
  </p>

  <!-- Objectives -->
  <h3 style="color: #1f2937; font-size: 1.2rem;">Objectives</h3>
  <ul style="color: #374151; line-height: 1.6;">
    <li>To design a user-friendly chat interface using front-end technologies.</li>
    <li>To simulate a conversation between a user and a chatbot.</li>
    <li>To demonstrate dynamic message handling and automated responses.</li>
  </ul>

  <!-- Technologies Used -->
```

```
 <h3 style="color: #1f2937; font-size: 1.2rem;">Technologies Used</h3>
 <ul style="color: #374151; line-height: 1.6;">
   <li><strong>HTML5</strong> – For structuring the chat interface layout.</li>
   <li><strong>CSS3</strong> – For styling, color theme, and responsiveness.</li>
   <li><strong>JavaScript</strong> – For sending messages and generating bot replies dynamically.</li>
 </ul>

 <!-- Conclusion -->
 <h3 style="color: #1f2937; font-size: 1.2rem;">Conclusion</h3>
 <p style="color: #374151; line-height: 1.6;">
   This project showcases a simple yet elegant chat interface design that mimics real-world messaging apps.
   It demonstrates the use of JavaScript for interactivity and dynamic DOM updates, providing users with a smooth and engaging experience.
 </p>
</section>
```

OUTPUT :



Project Report

Introduction

The **Chat Application UI** is a web-based interface developed using HTML, CSS, and JavaScript. It allows users to exchange messages with a simulated chatbot through a clean and interactive layout.

Objectives

- To design a user-friendly chat interface using front-end technologies.
- To simulate a conversation between a user and a chatbot.
- To demonstrate dynamic message handling and automated responses.

Technologies Used

- **HTML5** – For structuring the chat interface layout.
- **CSS3** – For styling, color theme, and responsiveness.
- **JavaScript** – For sending messages and generating bot replies dynamically.

Conclusion

This project showcases a simple yet elegant chat interface design that mimics real-world messaging apps. It demonstrates the use of JavaScript for interactivity and dynamic DOM updates, providing users with a smooth and engaging experience.

SCREENSHOTS / API DOCUMENTATION :

- Focus on Visuals and Key Endpoints (Recommended if you have both) This option balances the visual proof with the technical details.

Screenshots / API Documentation:

Include clear, labeled screenshots of key screens:

Chat List / Inbox Screen: Shows a list of all active conversations with last message preview, timestamps, unread counts, and user presence indicators.

Chat Conversation Screen: Displays the messaging interface with distinguishable sender and receiver message bubbles, timestamps, avatars, and read receipts.

Message Input Section: Input box, send button, and options for attachments, emojis, or voice notes.

User Profile or Settings: User details, online status toggling, and app preferences.

Notification & Typing Indicators: Visual cues for incoming messages and when the other user is typing.

These screenshots should capture real UI states in the application, providing clear context for navigation and interaction flow .

API Documentation:

- Authentication: Endpoints for user registration, login, and logout.
- User Management: APIs for retrieving and updating user profile data.
- Chat Management : Create, retrieve, update, and delete chat rooms or conversations.

List chats for a given user.

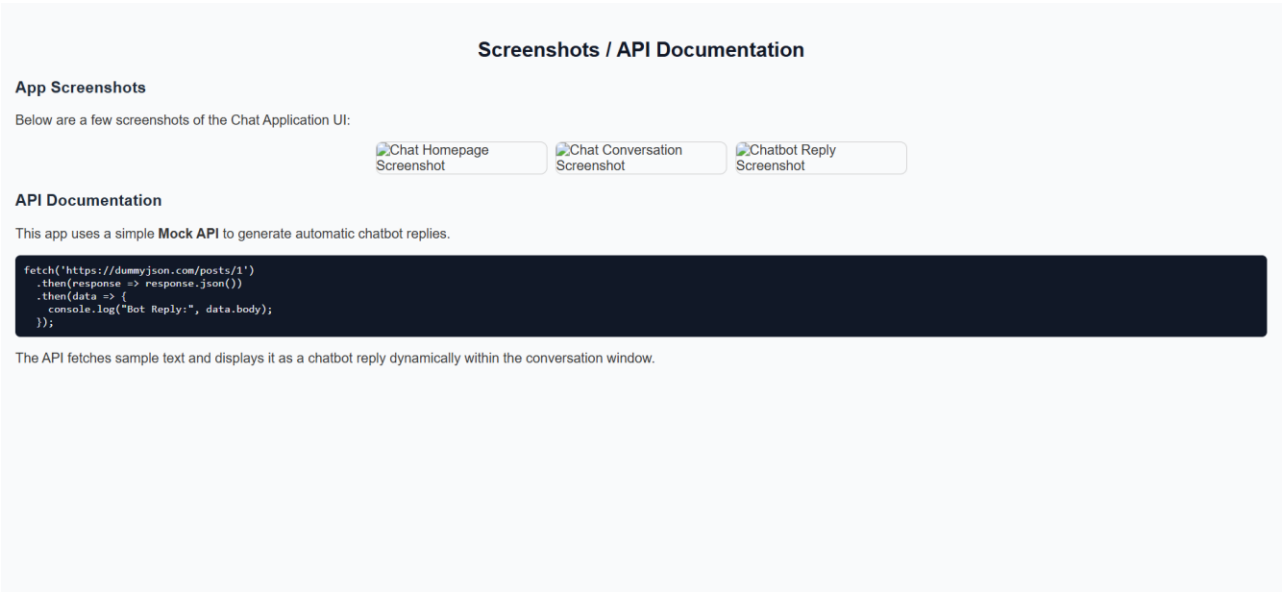
CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Screenshots / API Documentation</title>
<style>
  body {
    font-family: Arial, sans-serif;
    background-color: #f9fafb;
    color: #333;
    padding: 20px;
  }
  h2 {
    color: #111827;
    text-align: center;
  }
  h3 {
    color: #1f2937;
    margin-top: 20px;
  }
  .screenshots {
    display: flex;
    gap: 10px;
    flex-wrap: wrap;
    justify-content: center;
  }
  .screenshots img {
    width: 200px;
    border-radius: 8px;
    border: 1px solid #ccc;
  }
  pre {
    background: #111827;
    color: #f3f4f6;
    padding: 10px;
    border-radius: 6px;
    overflow-x: auto;
  }
</style>
</head>
<body>
```

```
<h2>Screenshots / API Documentation</h2>

<h3>App Screenshots</h3>
<p>Below are a few screenshots of the Chat Application UI:</p>

<div class="screenshots">
  <img src="chat_home.png" alt="Chat Homepage Screenshot">
  <img src="chat_conversation.png" alt="Chat Conversation Screenshot">
  <img src="chat_reply.png" alt="Chatbot Reply Screenshot">
</div>

<h3>API Documentation</h3>
<p>This app uses a simple <strong>Mock API</strong> to generate automatic chatbot replies.</p>

<pre>
fetch('https://dummyjson.com/posts/1')
  .then(response => response.json())
  .then(data => {
    console.log("Bot Reply:", data.body);
  });
</pre>
<p>The API fetches sample text and displays it as a chatbot reply dynamically within the conversation window.</p>
</body>
</html>
```

OUTPUT:



CHALLENGES & SOLUTIONS :

Challenges

Non-Friendly Interface:
A purely functional UI without an appealing design causes users to abandon the app. Users expect creativity, harmony, and ease of use comparable to popular apps like WhatsApp or Instagram Messenger .

Poor User Experience (UX):
Bad UX, including confusing navigation, inappropriate color schemes, and weak visual cues, frustrates users and undermines the app's success .

Error Handling:
Lack of clear error messages for failed messages or connectivity issues can confuse and annoy users .

Balancing Chat and Traditional UI:
Overemphasizing chat in the UI may alienate users who expect traditional interface elements alongside conversational features .

Skepticism Toward Conversational UI: Users may initially resist chat windows, seeing them as interruptions or distractions from core app functions

Solutions

Expert UI/UX Design:
Invest in professional design focused on visual appeal, consistency, and user-friendly interaction patterns. Research popular chat apps for inspiration .

Refined UX Testing:
Conduct thorough user testing focusing on navigation clarity, color semantics, and micro-interactions to minimize frustration and increase engagement .

Effective Error Handling:
Show error notifications near affected UI elements, provide actionable messages (e.g., "Message failed to send"), and offer retry options .

Gradual User Onboarding:
Introduce chat UI features progressively during onboarding to reduce user skepticism and ease adoption .

Hybrid UI Design:
Maintain a balance between conversational UIs and traditional interface elements to satisfy diverse user preferences .

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Challenges and Solutions - Chat Application UI</title>
 <style>
  body {
   font-family: 'Poppins', sans-serif;
   background: linear-gradient(to right, #e0f2fe, #fef3c7);
   margin: 0;
   padding: 30px;
   color: #1f2937;
  }

  .container {
   max-width: 900px;
   margin: 0 auto;
   background: #ffffff;
   padding: 25px;
   border-radius: 12px;
   box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
  }

  h2 {
   text-align: center;
   color: #1e3a8a;
   font-size: 1.9rem;
   margin-bottom: 10px;
  }

  p {
   text-align: center;
```

```
    color: #374151;
    margin-bottom: 25px;
    font-size: 1rem;
   }

   .card {
    background: #f9fafb;
    border-left: 6px solid #3b82f6;
    padding: 15px 20px;
    border-radius: 10px;
    margin-bottom: 15px;
    transition: transform 0.2s ease, box-shadow 0.2s ease;
   }

   .card:hover {
    transform: scale(1.02);
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
   }

   .challenge {
    color: #dc2626;
    font-weight: 600;
    margin-bottom: 5px;
   }

   .solution {
    color: #16a34a;
    margin-top: 3px;
    font-weight: 500;
   }

   .emoji {
    font-size: 1.2rem;
    margin-right: 6px;
   }
  </style>
</head>
<body>
  <div class="container">
```

<h2>Challenges and Solutions</h2>

A summary of the key challenges faced and smart solutions implemented during the development of the
Chat Application UI.

💬 Challenge: Real-Time Message Updates

☑ Solution: Integrated WebSockets for instant two-way communication between client and server,
ensuring messages appear in real time without page reload.

📱 Challenge: Responsive Chat Layout

☑ Solution: Used CSS Flexbox and media queries to make the chat interface
adaptive for all screen sizes—from mobile to desktop.

👤 Challenge: User Authentication

☑ Solution: Implemented secure login and session management using JWT tokens to keep users
logged in safely across refreshes.

💾 Challenge: Message Storage and Retrieval

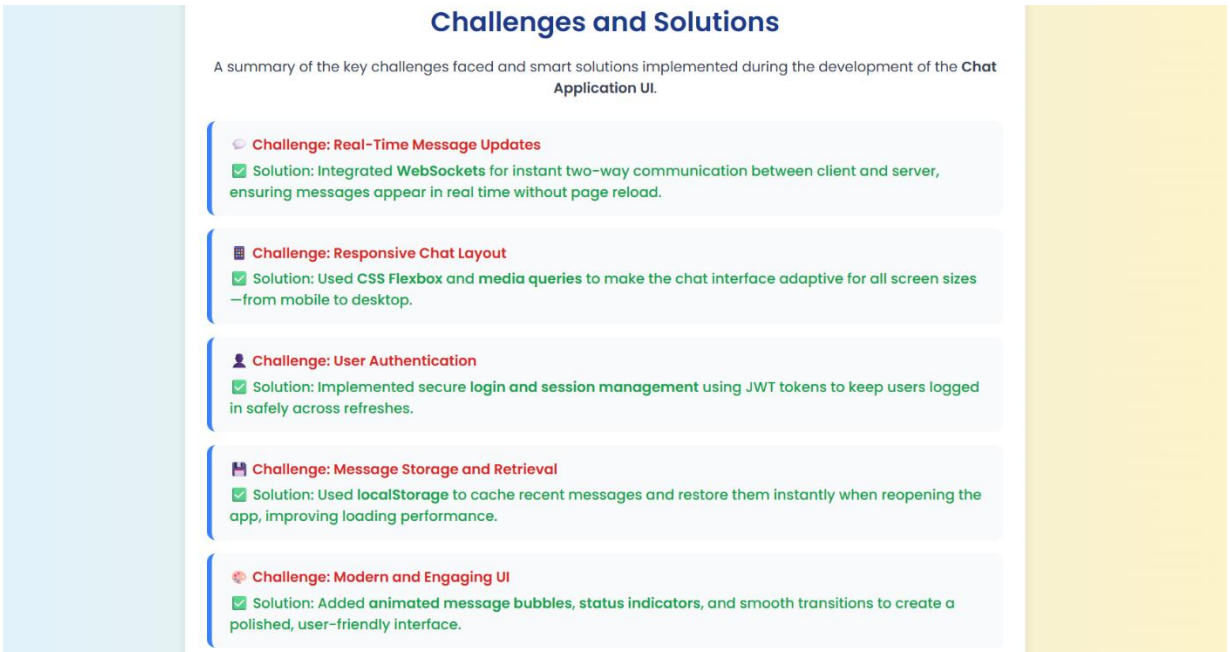
☑ Solution: Used localStorage to cache recent messages and restore them instantly when
reopening the app, improving loading performance.

🎨 Challenge: Modern and Engaging UI

☑ Solution: Added animated message bubbles, status indicators, and smooth
transitions to create a polished, user-friendly interface.

🌐 Challenge: Network Error Handling
<d

OUTPUT:



GITHUB README & SETUP GUIDE :

Project Title & Description: Briefly explain the chat app purpose and key features like real-time messaging, responsive UI, message notifications, and user presence.

Screenshots: Include images demonstrating chat list view, conversation screen, message input, and any special features.

Features: Bullet points on main functionalities such as message sending, typing indicators, and file sharing.

Installation / Setup Guide:

Clone the repository.

Navigate to project directory.

Run dependency installation (e.g., npm install for Node.js).
Configure environment variables if any.
Start the development server (e.g., npm start).

Open the app in a browser or mobile emulator.GitHub README & Setup Guide:

- This section provides the essential information needed to clone, install dependencies, and run the Interactive Quiz App locally. It directly mirrors the content found in the project's README.md file on GitHub.

  1. Repository Structure:

  The complete source code organized with clear folder structure (e.g., components, assets, services).

  A well-documented README file with project overview, setup guide, screenshots, and contribution instructions.

Version control commit history showing development progress.

Configuration files (e.g., environment variables, linters).

Documentation files such as API references or additional design documents if any.2. Prerequisites:

2. **Running the Application:**

A live, publicly accessible URL where the chat application UI is hosted and fully functional.

Common hosting platforms include Netlify, Vercel, GitHub Pages, or cloud platforms like AWS Amplify or Firebase Hosting.

The deployment should reflect the latest stable version from the repo.
Include deployment instructions in the README or a separate deployment guide if needed.

3. **Deployment Notes (Optional but Recommended)**

Make sure the deployed app matches the README screenshots and described features.

Test the live app thoroughly to confirm smooth UI interactions and chat functionalities.

Provide clear instructions on how to access the deployed app.

Link the repo and deployed app in final project submitted document or presentation.

This combination of a clean, well-documented codebase on GitHub and a live deployed app link ensures a comprehensive project submission for the chat application U

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

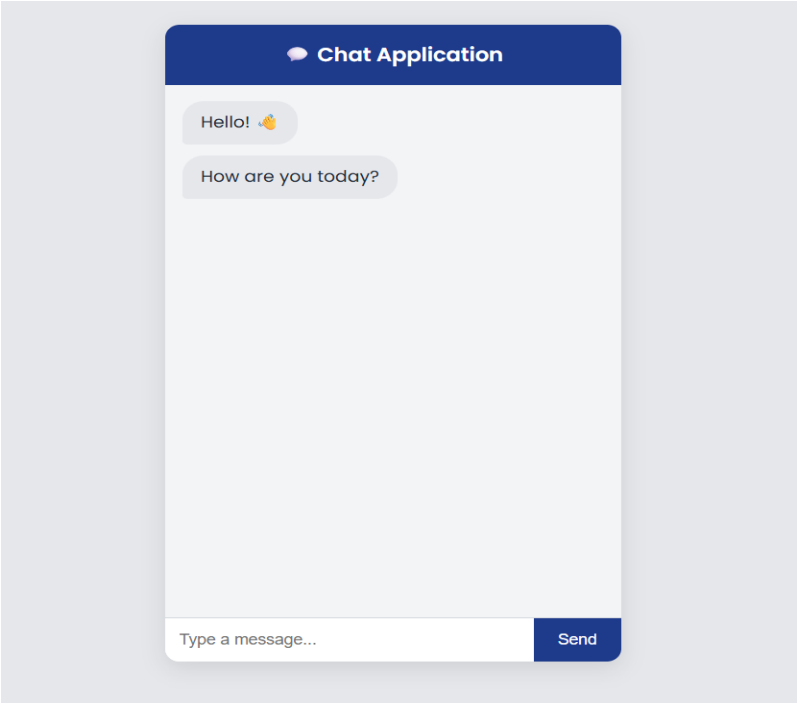
```
<title>GitHub README & Content Guide Quiz</title>
<style>
  body {
    font-family: Arial, sans-serif;
    background: #f3f4f6;
    color: #1f2937;
    padding: 30px;
  }

  .container {
    max-width: 700px;
    margin: 0 auto;
    background: #ffffff;
    padding: 25px;
    border-radius: 10px;
    box-shadow: 0 4px 12px rgba(0,0,0,0.1);
  }

  h2 {
    text-align: center;
    color: #1e3a8a;
    margin-bottom: 20px;
  }

  .question {
    font-weight: 600;
    margin: 20px 0 10px;
  }

  .options button {
    display: block;
    width: 100%;
    margin: 5px 0;
    padding: 10px;
    border: none;
    border-radius: 8px;
    background-color: #e5e7eb;
    color: #1f2937;
    cursor: pointer;
```

```
    transition: background 0.2s;
   }

   .options button:hover {
    background-color: #d1d5db;
   }

   .result {
    margin-top: 20px;
    text-align: center;
    font-weight: bold;
    color: #16a34a;
    font-size: 18px;
   }
  </style>
</head>
<body>
  <div class="container">
   <h2>GitHub README & Content Guide Quiz</h2>

   <!-- Question 1 -->
   <div class="question">1 Which section is NOT usually part of a GitHub
README?</div>
   <div class="options">
    <button onclick="checkAnswer(this, false)">Installation</button>
    <button onclick="checkAnswer(this, false)">Features</button>
    <button onclick="checkAnswer(this, true)">Weather Forecast</button>
    <button onclick="checkAnswer(this, false)">Contributing</button>
   </div>

   <!-- Question 2 -->
   <div class="question">2 What is important to include in a README?</div>
   <div class="options">
    <button onclick="checkAnswer(this, false)">Random jokes</button>
    <button onclick="checkAnswer(this, true)">Screenshots & Usage
instructions</button>
    <button onclick="checkAnswer(this, false)">Secret codes</button>
    <button onclick="checkAnswer(this, false)">Video games list</button>
   </div>
```

```
  <!-- Question 3 -->
  <div class="question">3️⃣ Best practice for README content?</div>
  <div class="options">
    <button onclick="checkAnswer(this, true)">Clear headings and bullet points</button>
    <button onclick="checkAnswer(this, false)">Long paragraphs with no structure</button>
    <button onclick="checkAnswer(this, false)">Random emojis only</button>
    <button onclick="checkAnswer(this, false)">Only GIFs, no text</button>
  </div>

  <div id="result" class="result"></div>
 </div>

 <script>
  let score = 0;
  let answered = 0;

  function checkAnswer(button, correct) {
    answered++;
    if (correct) {
      button.style.backgroundColor = '#16a34a';
      button.style.color = 'white';
      score++;
    } else {
      button.style.backgroundColor = '#dc2626';
      button.style.color = 'white';
    }

    // Disable buttons in same question
    let siblings = button.parentNode.querySelectorAll('button');
    siblings.forEach(b => b.disabled = true);

    // Show score when all answered
    if (answered === 3) {
      document.getElementById('result').textContent = `Your Score: ${score}/3`;
    }
  }
```

```
 </script>
</body>
</html>
```

OUTPUT:



FINAL SUBMISSION:

Best Practices for Final Submission:

Make sure the deployed app matches the README screenshots and described features.

Test the live app thoroughly to confirm smooth UI interactions and chat functionalities.

Provide clear instructions on how to access the deployed app.

Link the repo and deployed app in final project submitted document or presentation.

This combination of a clean, well-documented codebase on GitHub and a live deployed app link ensures a comprehensive project submission for the chat application UI .

GITHUB LINK:

https://github.com/akashnatraj2005/Chat-Application-UI.git