9530

St.MOTHER THERESA ENGINEERING COLLEGE

COMPUTER SCIENCE ENGINEERING

NM-ID: 1FB00D2D33C5AB9A5B542682601673AD

REG NO: 953023104009

DATE:29-09-2025

Completed the project named as

Phase 3

FRONT END TECHNOLOGY

CHAT APPLICATION UI

SUBMITTED BY,

ANDREW FERNANDO R

6381490340

Phase 3 – MVP Implementation:

1. Project Setup :

- **Development Environment**:
  - Set up the IDE (VS Code recommended).
  - Install Node.js and package manager (npm/yarn).
  - Choose a frontend framework (React, Vue, Angular). For MVP, React + Tailwind CSS works best.
- **Project Initialization**:
  - Run npx create-react-app chat-ui (or framework equivalent).
  - Create a GitHub repository for version control.
  - Set up .gitignore for node_modules, build files, and sensitive configs.
- **Dependencies**:
  - UI: Tailwind CSS / Material UI.
  - State management: Redux / Context API.
  - Realtime: Socket.io-client.
  - Testing: Jest + React Testing Library.

2. Core Features Implementation :

- **User Interface (UI)**:
  - **Login/Signup screen** (basic form).
  - **Chat Room screen** with:
    - Chat window (list of messages).
    - Input box for typing messages.
    - Send button (triggers message event).
    - Online user list (optional for MVP).
- **Messaging Flow**:
  - Display messages in a scrollable container.

- Differentiate between sent and received messages (different colors/bubbles).
- Auto-scroll to the latest message.

- **Realtime Communication**:
  - Connect frontend with backend using Socket.io-client.
  - Implement “send message” and “receive message” events.
- **Error Handling & Feedback**:
  - Show error messages if a message fails to send.
  - Loading indicators for network calls.

3. Data Storage (Local State / Database) :

- **Local State**:
  - Use React state/Context API for chat UI updates.
  - Maintain current user info, active room, and message history.
- **Temporary Storage**:
  - Store recent messages in local state for fast rendering.
  - Use localStorage/sessionStorage for user session persistence.
- **Database Integration (optional for UI MVP)**:
  - For a full MVP, connect to a backend with MongoDB/Firebase to store chats.
  - Save messages, user profiles, and timestamps.

4. Testing Core Features :

- **Unit Tests**:
  - Test UI components (e.g., does MessageBubble render correctly?).
  - Validate input field behavior.
- **Integration Tests**:
  - Test message flow (typing → sending → appearing in chat).
  - Ensure socket events trigger correctly.

- **End-to-End (E2E) Tests**:
  - Use Cypress/Playwright to simulate a user joining a chat room and exchanging messages.
- **Bug Tracking**:
  - Use GitHub Issues for logging UI/UX bugs (e.g., message duplication, alignment issues).

5. Version Control (GitHub) :

- Link : https://github.com/Andrew-57/NM-PROJECT-.git