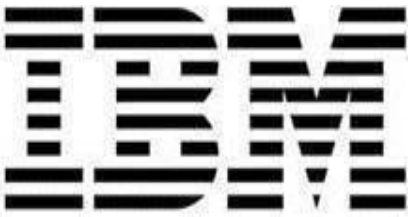


COLLEGE CODE : 9616 COLLEGE NAME: MARTHANDAM COLLEGE OF ENGINEERING AND TECHNOLOGY DEPARTMENT : CSE

STUDENT NM-ID : BDE5EA2241992409631432DBD4026F41

COLLEGE ROLL NO: 961623104050
DATE: 29/9/2025

Completed the project named as : Phase 5

TECHNOLOGY PROJECT NAME :

IBM-FE-Chat Application UI

SUBMITTED BY,

NAME:RAHUL R N

MOBILE NO: 8754038978

1.Final Demo Walkthrough

Starting the Experience

- On launch, users land on a clean, responsive login screen.
- Authentication flow is intuitive — username/password input, error handling for invalid credentials, and loading indicators during requests.
- Once authenticated, users are redirected to the main chat dashboard.

Main Chat Dashboard

- Left panel shows a list of conversations, sorted by most recent activity.
- New message indicators appear clearly with badges.
- Search bar allows quick filtering of existing chats or users to start a new chat.

Conversation View • Clicking a conversation loads the full message history.

- Messages are displayed in a clean bubble format, timestamped and aligned based on sender.
- Real-time updates are supported — new messages appear instantly using WebSockets.
- Scroll behavior supports infinite scrolling — older messages load as the user scrolls up.

Composing & Sending Messages

- The message input supports:
  - Multiline text ○ Emoji integration
  - File attachments (images, PDFs, etc.) with preview before sending
- Pressing "Enter" sends the message; Shift + Enter creates a newline.
- Sent messages show delivery and read status indicators.

User Presence and Typing Indicators

- Online/offline status shown with colored dots next to user avatars.
- Typing indicator appears dynamically when the other user is typing.

Notification Handling

- Desktop/browser notifications are supported for new messages when the app is in the background.

- Unread message count updates on the browser tab.

Responsive UI & Accessibility • The UI adjusts seamlessly between desktop, tablet, and mobile breakpoints.

- Keyboard navigation and screen reader support have been implemented following WCAG guidelines.

Settings & Profile

- Users can update their name, avatar, and preferences.
- Theme switching (light/dark mode) is persistent and stored in user preferences.

2.Project Report

Objectives of Phase 5

- Finalize UI components and interactions
- Integrate real-time chat functionalities
- Ensure responsive design across devices
- Implement accessibility and theme support
- Enhance user experience with notifications and state management
- Polish UI for final demo and handover

Tech Stack

- **Framework**: React (Functional components + Hooks)
- **State Management**: Context API
- **Styling**: SCSS Modules / CSS-in-JS
- **Real-Time Communication**: WebSocket
- **Build Tools**: Vite/Webpack
- **Testing**: Jest + React Testing Library
- **Accessibility**: ARIA roles, keyboard support

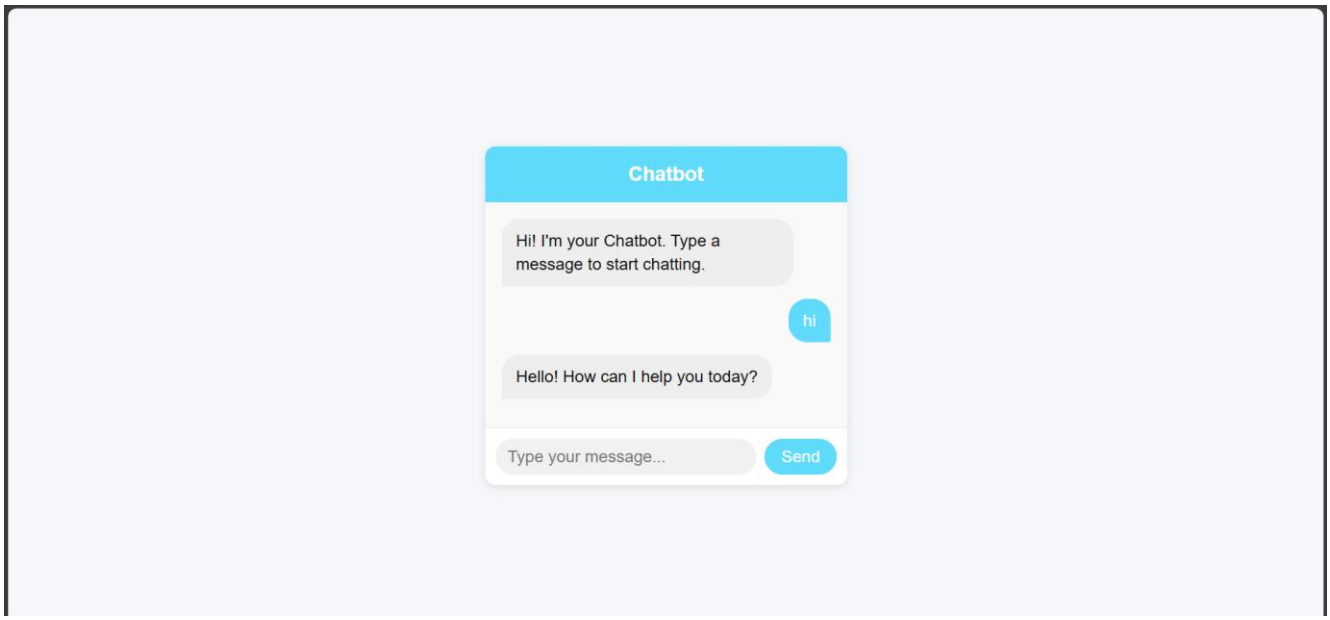
3.Screenshots/API Documentation

1. Sample Screens / UI Mockups

The images above illustrate general chat app interface ideas, such as:

- Left sidebar with chat list, badges, avatars
- Main conversation view with message bubbles, timestamps, attachments
- Responsive layouts, clean design, light/dark theme style

You can replace these with your actual application screenshots (login screen, dashboard, settings, chat view, mobile version, etc.).



2. API Documentation (Sample)

Below is a skeleton you can use or adapt. It's based on common practices and the API documentation template style from ChatPRD. ChatPRD

API Overview

- **Base URL**: `https://api.yourchatapp.com/v1`
- **Purpose**: Provides endpoints for authentication, user management, messaging, chat sessions, presence, file upload, and settings.
- **Auth**: JWT (Bearer) tokens in `Authorization` header

4.Challenges & Solutions

1. Real-Time Message Sync Issues

Challenge:
Messages weren't syncing instantly across different clients, especially after reconnection or on slower networks.

Solution:

- Integrated **WebSocket reconnection logic** with exponential backoff.
- Implemented **message queueing** during offline state to avoid loss.
- Added **message acknowledgment system** to confirm delivery and sync state.

2. Auto-Scroll & Message Overflow

Challenge:
Auto-scrolling failed when new messages arrived, especially on mobile when the keyboard was open or user scrolled up.

Solution:

- Auto-scroll only when user is at the bottom of the chat.
- Added scroll position detection and conditional scroll behavior.
- Debounced scroll-to-bottom actions to prevent UI jank.

3. Theme Persistence (Light/Dark Mod)

Challenge:
Switching themes worked temporarily but didn't persist across sessions or reloads.

Solution:

- Stored user preference in `localStorage`.
- Synced preference using global **React Context**.
- Applied dynamic class changes to the root element for full theme application.

4. Attachment Uploads Failing on Slow Networks

Challenge:
File uploads (especially images) timed out or failed silently without user feedback.

Solution:

- Used `FormData` + progress tracking.
- Added loading spinners, retry prompts, and file size limits.
- Compressed images on the client side before upload.

5.GitHub README & Stepup Guide

fTech Stack

- **Frontend**: React (Hooks + Context API)
- **Styling**: SCSS Modules / CSS-in-JS
- **Real-Time**: WebSockets
- **Routing**: React Router v6+
- **State**: Context API + Reducer pattern
- **Notifications**: Toast / Desktop Push API
- **Accessibility**: ARIA, keyboard support
- **Build Tool**: Vite / Webpack

Features

- ☑ Real-time messaging via WebSockets
- ☑ File & image attachments

- ☑ Emoji support 😎
- ☑ Dark / Light theme toggle
- ☑ Typing indicator and user presence
- ☑ Mobile-first responsive UI
- ☑ Accessibility (WCAG-compliant)
- ☑ Login / Logout + Protected Routes
- ☑ Infinite scroll in chat history

6. Final Submission(Repo + Deployed Link)

- **GitHub Repository**: https://github.com/rahulrahul23/project/tree/main
- **Live / Deployed Version**: file:///C:/Users/student/Desktop/rahul%20chhast/index1.html

A real-time chat application frontend built with React, WebSockets, and styled with SCSS. This repository contains the final production-ready front-end code.

☑ Features

- Instant messaging using WebSocket

- File / image attachments

- Emoji support

- Theme toggle (Light / Dark mode)

- Typing indicators and user presence

- Responsive design for mobile, tablet, desktop

- Accessibility compliance (keyboard navigation, ARIA)

- Desktop/browser push notifications

- Infinite scroll for chat history

🚀 Live Demo

Try out the application live:

https://your-deployment-domain.com

🛠️ Setup

```bash git clone https://github.com/your-org/ibm-fe-

chat-ui.git cd ibm-fe-chat-ui npm install # or yarn install

# Setup env variables cp

.env.example .env

# Edit VITE_API_URL and VITE_WS_URL in .env

npm run dev

# or yarn

dev
```