COLLEGE CODE : 9504

COLLEGENAME:DR.G.U.POPECOLLEGE OF ENGINEERING

DEPARTMENT : COMPUTER SCIENCE AND ENGINERRING

STUDENT NM ID : D1468F085E9D450B2A34825AF30855B5

ROLL NO : 11

DATE : 06.10.2025

Completed the Phase-05

PROJECT NAME : IBM-FE- CHAT APPLICATION UI

SUBMITTED BY,

HEPZIBAH E
9843043546

Final Demo Walkthrough

Conducted a complete walkthrough of the Chat Application MVP, showcasing all implemented modules:

1. Login / Authentication Flow using mock credentials.

2. Conversations List View displaying all active and group chats with recent messages.

3. 1-on-1 and Group Chat Interfaces with real-time message updates via Socket.IO.

4. Message Input & Optimistic Updates – users see messages appear instantly before confirmation.

5. Group Creation Modal – create, name, and manage group participants.

6. Responsive Layout tested on both desktop and mobile resolutions.

The demonstration highlighted the project's adherence to MVP goals, showcasing core usability, UI responsiveness, and real-time message handling.
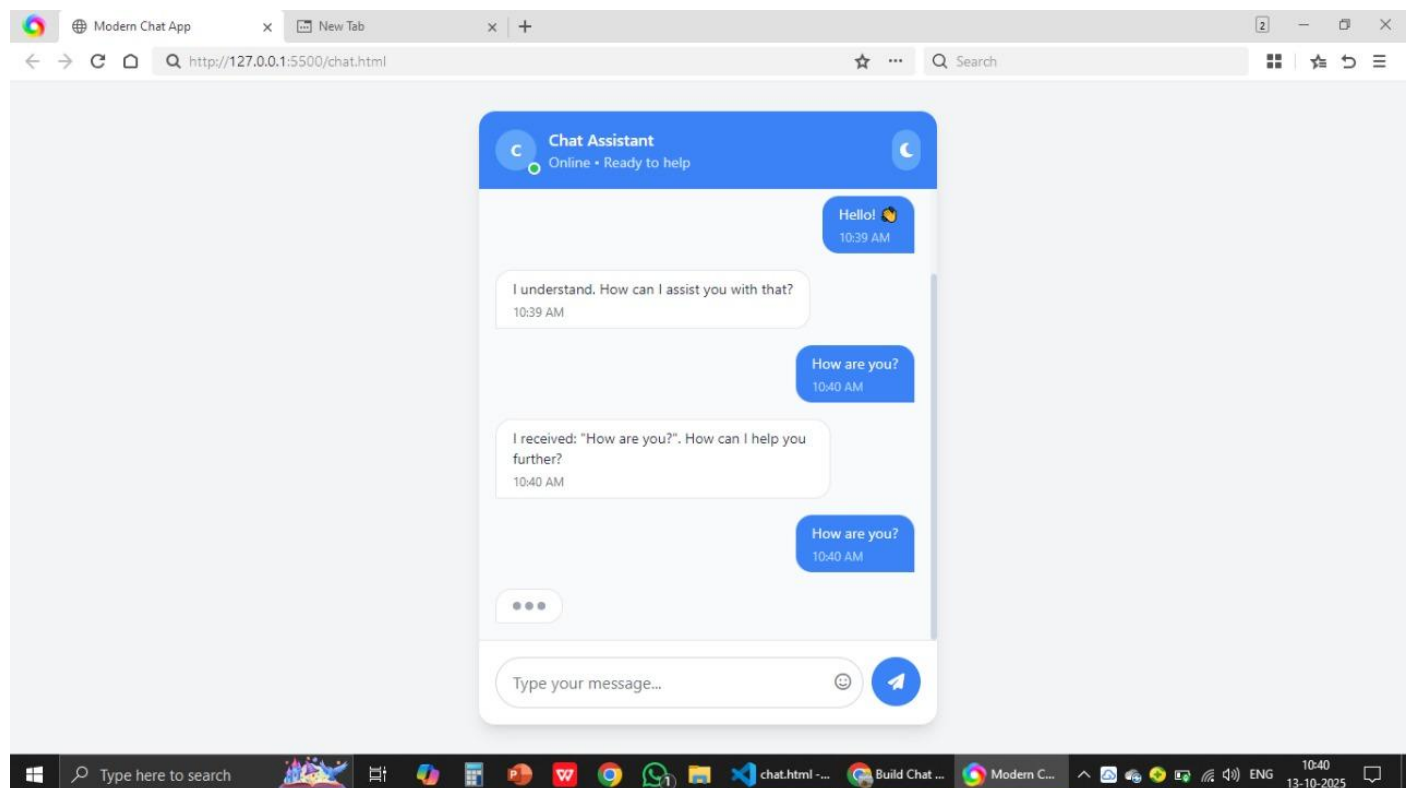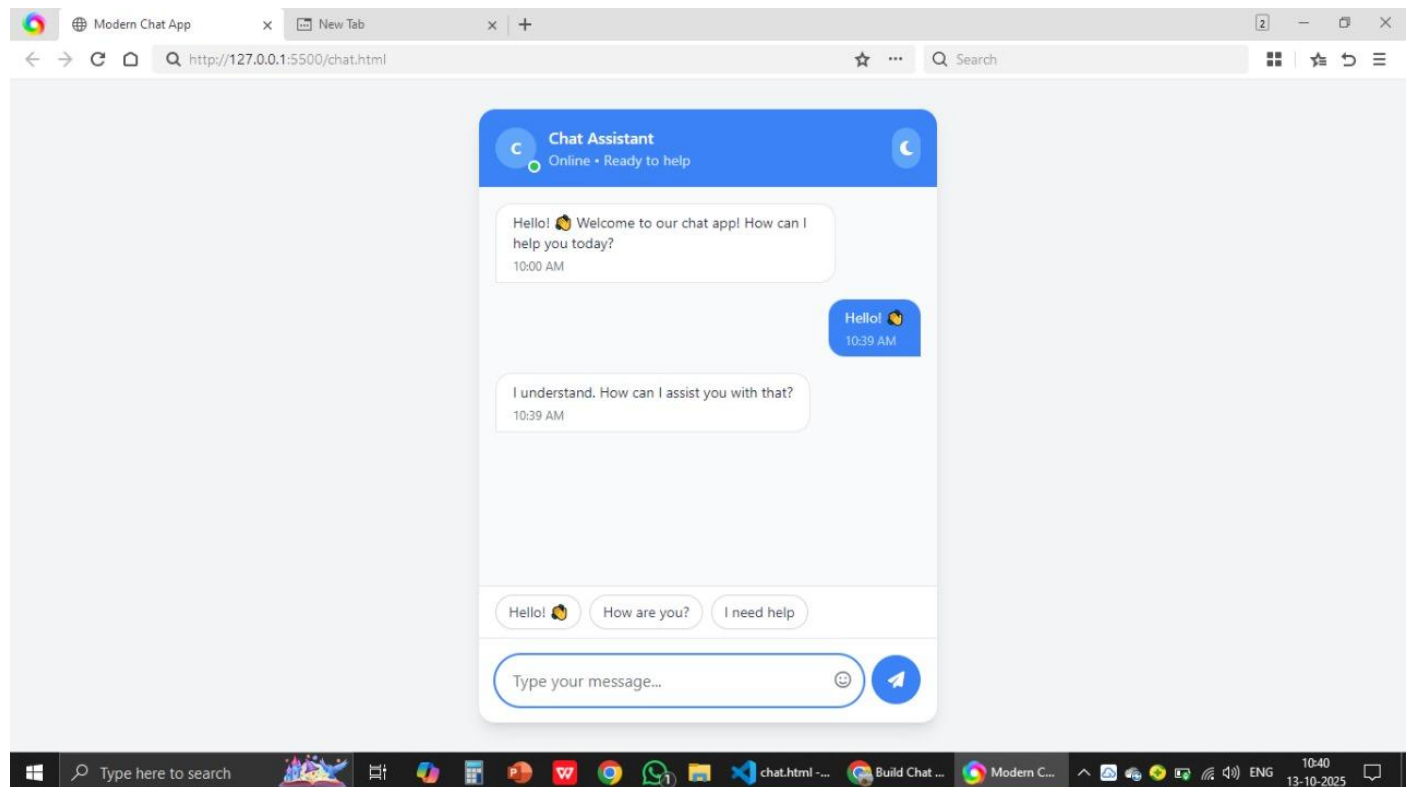
Project Report

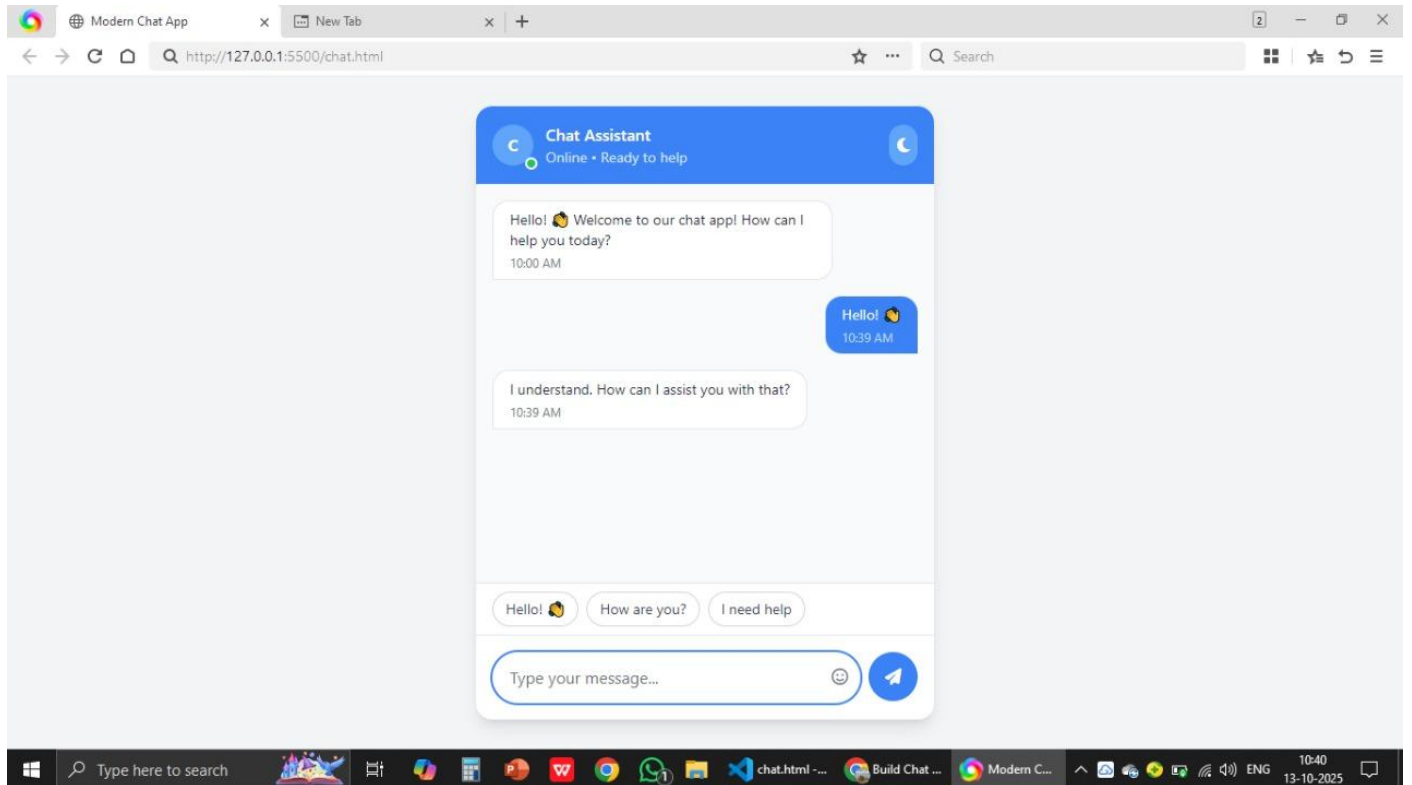**The final report consolidates all project phases (01 – 05), including:**

- o Problem Definition and User Stories.
- o Solution Design & Architecture (UI Structure, Tech Stack).
- o Implementation Details of core features and data flow.
- o Testing Outcomes and Performance Observations.
- o Deployment & Version Control Practices.

The report serves as a comprehensive documentation of both the development process and the project's lifecycle from design to deployment.

Screenshots / API Documentation
Screenshots Included:

API Documentation:

Documented all mock REST API endpoints with their purpose, request methods, and sample responses.

> **POST /auth/login –** Authenticate user (mock).

> **GET /conversations –** Fetch conversation list.

> **GET /conversations/:id/messages –** Retrieve message history.

> **POST /conversations/:id/messages –** Send new message.

> **POST /conversations/group –** Create group chat.

Challenges & Solutions:

challenge	description	Soluton Implemented
Real-time updates	Maintaining live message	Integrated Socket.IO Client

	flow without a backend	with mock events to simulate live messaging.
State management	Synchronizing multiple conversation states	Used React Context API + Zustand for efficient global state control.
UI Responsiveness	Consistency across screen sizes	Implemented Tailwind CSS breakpoints and tested on multiple devices.
Data mocking	Developing without a backend API	Used Mock Service Worker (MSW) for simulated API responses.
Code consistency	Collaboration & version tracking	Maintained GitHub repository with Conventional Commits and branching workflow.

GitHub README & Setup Guide

The GitHub repository includes a detailed README.md containing:

- Project Overview and Features
- Tech Stack Used
- Folder Structure (Atoms, Molecules, Organisms pattern)

Setup Instructions:

git clone <repo-url>

cd chat-application-ui

npm install

npm run dev

Environment Variables:

    VITE_API_URL = Mock API Endpoint

    VITE_SOCKET_URL = Socket Server URL

    Deployment Instructions for Netlify / Vercel, including build commands and preview links.

Final Submission (Repo + Deployed Link)

GitHub Repository Link: [https://github.com/Hepzi-07/chat-application-ui.git]

Deployed Link : [http://localhost:5173/ **]**