Project Plan

Group-10 ,Section-9

ChatWave – Real-Time Web Chat Application

Name : Md.Minhajul Islam

ID :2211022042

Section:09

Group :10

Client/Customer: Any person, Friends Group, ECE Department, North South University

Team Members:

	Name	ID
1	Md.Minhajul Islam	2211022042
2	Tabassum Tasnim	2211451042
3	Shuvra Saha Rimjim	2222483042
4	Md. Jalal Abedin	2211158642

Outline Description:

ChatWave is a real-time chat web application designed for fast, secure, and efficient messaging. It offers both private and group chat features, user presence status, and support for media sharing. The platform is tailored to work across devices and browsers using modern technologies like React, Node.js, and Socket.io.

The objective is to create a reliable communication platform suitable for:

- Personal messaging
- Team collaboration in corporate settings
- Educational use (student-teacher communication)
- Real-time customer support systems

Current Status:

- Proposal & SRS completed
- Initial GitHub repository created
- Preliminary system diagrams (ER, Use Case, Class, Sequence) in progress
- Basic tech stack setup completed (React frontend + Express backend)

Plan:Major Stages & Assigned Tasks

Stage	Task Description	Assigned To	Timeline
1. Setup & Planning	Requirement analysis, SRS creation	Md.Minhajul Islam	Week 1
2. UI/UX Design	Wireframes and component planning	Md.Minhajul Islam	Week 2
3. Backend Development	Auth system, user model, DB setup	Md.Minhajul Islam	Week 3,4
4. Frontend Development	Login/signup pages, message UI	Md.Minhajul Islam	Week 3,4
5. Real-Time Messaging	Implementing Socket.io for chat	Md.Minhajul Islam	Week 5
6. Media & File Sharing	File upload and retrieval	Md. Jalal Abedin	Week 6
7. Notifications	In-app and push notifications	Tabassum Tasnim	Week 7
8. Testing & Debugging	Functional and security testing	Shuvra Saha Rimjim	Week 8
9. Deployment	Host frontend/backend + connect MongoDB Atlas	Md.Minhajul Islam	Week 9

Technical Environment:

- **Frontend:** React.js, TailwindCSS, Daisy UI
- **Backend:** Node.js, Express.js
- **Real-Time Communication:** Socket.io
- **Database:** MongoDB (Atlas)
- **Auth & Security:** JWT, Bcrypt
- **Version Control:** Git + GitHub
- **Deployment Platforms:** Vercel (Frontend), Render or Railway (Backend)

Architectural Design (Overview)

The architecture of ChatWave follows a modern web-based client-server model using the MERN stack and additional libraries for real-time communication and state management. The high-level design is structured as follows:

- **Client Side (Frontend):**
  - ➢ Built using React.js with TailwindCSS and Daisy UI for a responsive and modern UI.
  - ➢ Connects to the server via Socket.io for real-time messaging and via REST APIs for CRUD operations (like user registration, login, and message history).
  - ➢ Uses Zustand for efficient global state management (e.g., storing authenticated user info, active chats, and UI states).

- **Server Side (Backend):**
  - ➢ Developed using Express.js (Node.js framework) to handle HTTP requests and Socket.io connections.
  - ➢ Follows a modular MVC (Model-View-Controller) structure for better scalability and maintainability.
  - ➢ Implements JWT-based stateless authentication for secure and token-based access control.

- **Database Layer:**
  - ➢ Uses MongoDB Atlas, a cloud-hosted NoSQL database.
  - ➢ Stores user credentials, chat messages, timestamps, and real-time session data.

- **Real-time Communication:**
  - ➢ Socket.io is used to implement bi-directional communication between users for instant chat messages, typing indicators, and online/offline status.

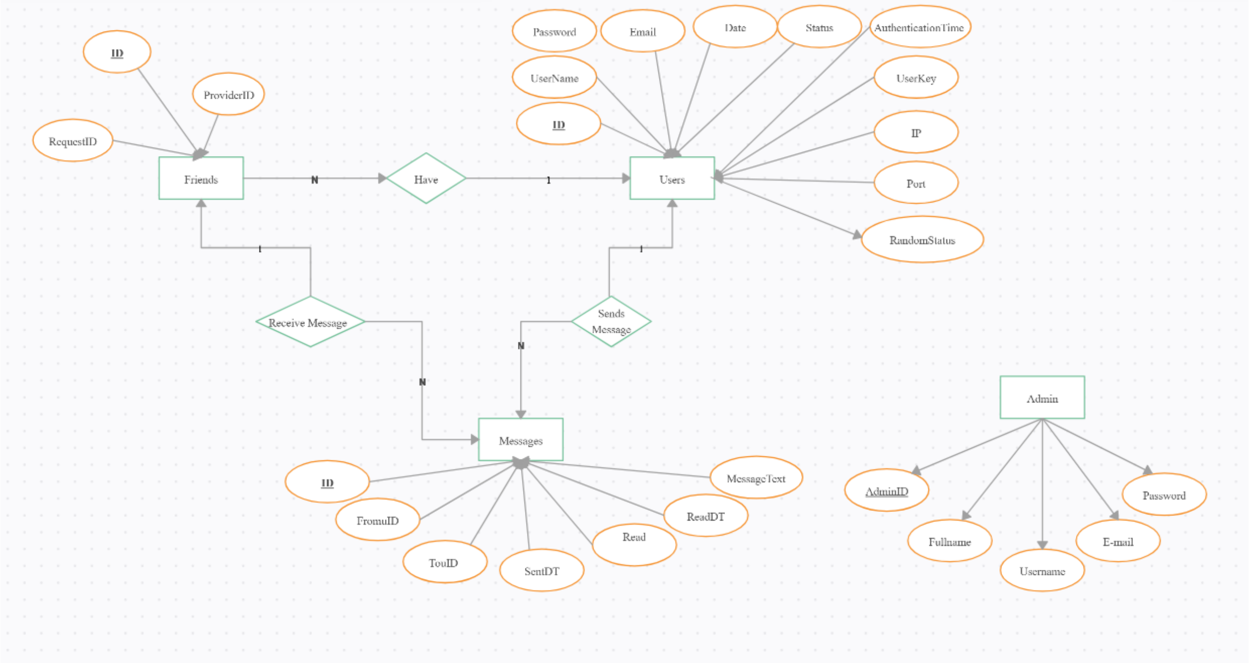
Scheduling Snapshot:

Week	Focus Area
1	SRS, project planning
2	UI/UX design
3-4	Backend setup & auth
5	Real-time communication
6	File sharing features
7	Notifications

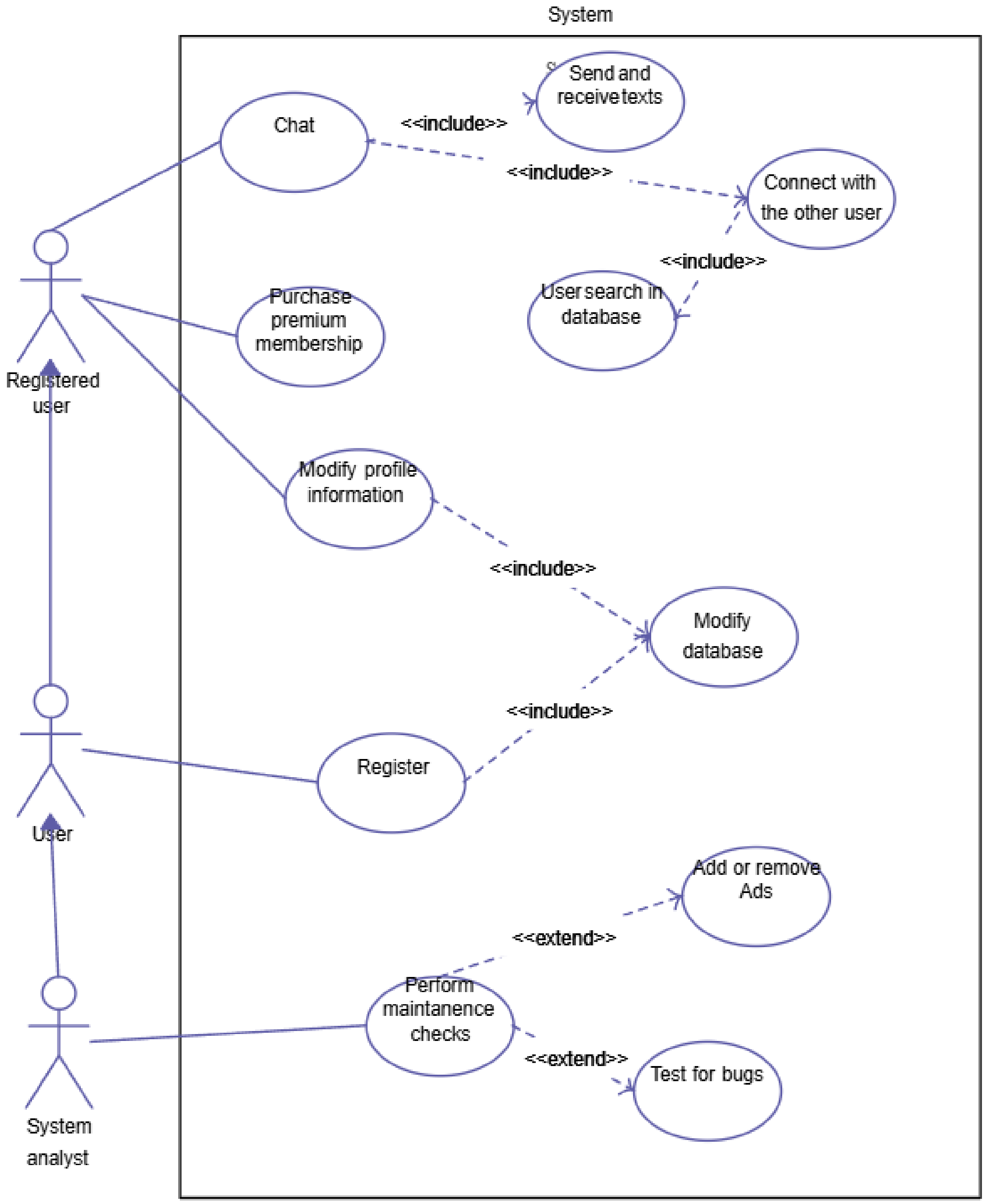
Week	Focus Area
8	Testing & optimization
9	Final deployment & wrap-up

Diagrams

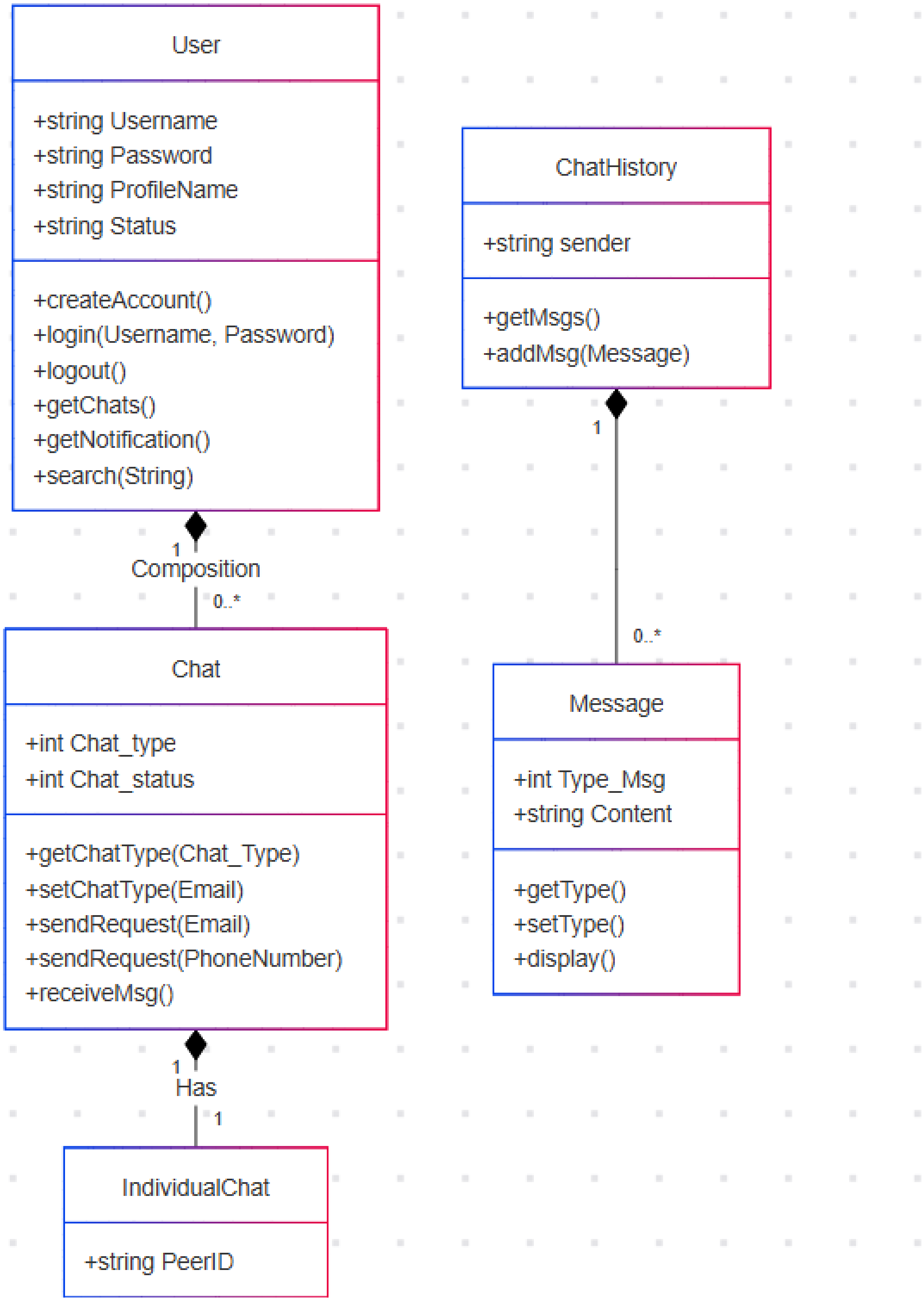
Entity Relationship (ER) Diagram:



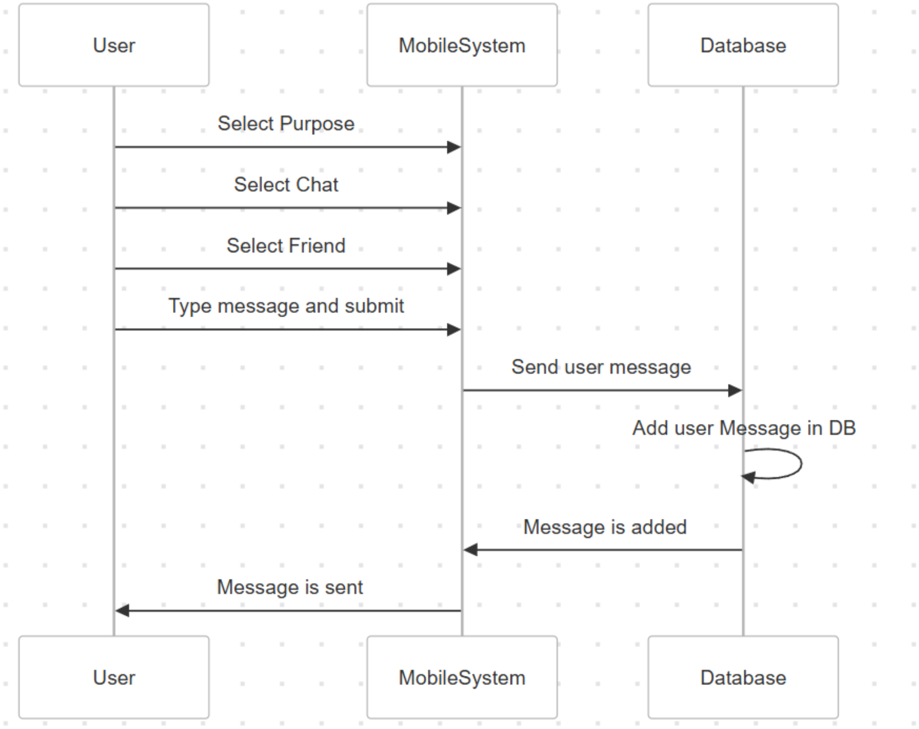
Use-Case Diagram:



Class Diagram:



Sequence Diagram:


User
MobileSystem
Database
Select Purpose
Select Chat
Select Friend
Type message and submit
Send user message
Add user Message in DB
Message is added
Message is sent
User
MobileSystem
Database