A Minor Project Mid Defense Report on

Ek-Kaan - A Chat Web App

Submitted in Partial Fulfillment of the Requirement for

the Degree of **Bachelors in Software Engineering**

Under Pokhara University

Submitted by:

Shahil Dhakal, 201631

Sandilya Silwal, 201636

Drishya Neupane, 201648

Under the supervision of:

Er. Aruna Chhatkuli

Date:

30th July, 2024

Department of Software Engineering

NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur, Nepal

ABSTRACT

The chat application is a modern web platform built using the MERN stack (MongoDB Atlas, Express.js, React.js, Node.js) to ensure a robust backend and frontend architecture. It utilizes Tailwind CSS and the DaisyUI Library to deliver a responsive and visually appealing user interface.

The application provides a seamless chatting experience with real-time messaging supported by WebSocket technology. It features user authentication and message encryption to maintain security. Users can create profiles, participate in private chats, and support sending and receiving messages and attachments.

MongoDB Atlas integration helps support data storage, while Express.js and Node.js enable efficient external processing. React.js combined with Tailwind CSS and DaisyUI enhances front-end development with reusable components and design elements.

This chat application targets users who seek a modern, responsive, and secure platform for communication.

Keywords – MERN stack, MongoDB Atlas, Express.js, React.js, Node.js, WebSocket, real-time messaging, user authentication, message encryption, file attachments, chat rooms, private conversations, scalable data storage, frontend development

Table Of Contents

List of Figures

List of Tables

1. Introduction

Effective and reliable messaging is essential in today's digital environment, where instant messaging plays an important role in personal and professional interactions. To meet this growing need, we plan to create a web interface built using the MERN stack, which includes MongoDB, Express.js, React.js, and Node.js. This technology option provides a robust, scalable, and maintainable architecture for both the backend and frontend components of the application.

Our chat application will offer users a comprehensive platform for real-time messaging, aiming to deliver a highly responsive and dynamic user experience. Key features include user authentication, message history, file sharing, and notifications.

WebSocket technology will facilitate instant communication, make messages instant and up-to-date, and improve communication by reducing latency.

To create an intuitive and user-friendly experience, we will use Tailwind CSS and DaisyUI library. These tools will allow us to create clean, responsive, and beautiful interfaces. The component-based architecture of React.js combined with the customizable styling options of Tailwind CSS will support the development of a modular and maintainable front-end.

The application will also prioritize performance, security, and data integrity. MongoDB, hosted in the cloud through MongoDB Atlas, will provide scalable data storage solutions. Express.js and Node.js will facilitate efficient server-side operations, supporting real-time updates and secure data management.

Overall, this chat web application is envisioned as a versatile tool for individuals and organizations, facilitating smooth and efficient communication across various contexts.

2. Problem Statement

Effective communication tools are essential in today's digital world, yet many existing messaging platforms face significant issues. Users often experience delays in message delivery, which disrupts conversations and reduces engagement. Ensuring real-time communication is crucial for a smooth user experience.

Security and privacy are major concerns as well. Many chat applications lack robust encryption and authentication measures, leaving user data vulnerable to breaches. There is a pressing need for secure messaging platforms that protect user information.

Another challenge is the user interface. Outdated or non-responsive designs hinder user satisfaction across different devices. Modern, intuitive interfaces are necessary to enhance the overall user experience.

Scalability also poses a problem. As user numbers grow, many applications struggle to manage increased data and traffic effectively, leading to performance issues.

This project seeks to resolve these issues by developing a chat web application with the MERN stack. The application will utilize WebSocket technology for real-time messaging, implement strong security measures, and feature a responsive design with Tailwind CSS and DaisyUI. These solutions will ensure a secure, scalable, and user-friendly communication platform.

3. Project Objectives

The objective of the “Chat Web Application” project is to develop a modern chat platform that addresses common issues in current messaging systems. The application aims to offer a robust solution for users who need a reliable, secure, and intuitive communication tool for personal and professional use.

Specifically, the main objective of the Chat Web Application is:

- To create a web application that facilitates real-time messaging with minimal latency while implementing robust security features, including secure user authentication and data encryption.

4. Significance of the Study

The Chat Web Application project holds significant importance in enhancing communication efficiency, security, and user experience, offering a range of benefits and potential applications.

- **Enhanced Communication Efficiency**: The application offers real-time messaging capabilities that facilitate instant communication and collaboration among users, crucial for improving productivity and responsiveness.
- **Heightened Security Standards**: Strong security measures, including robust authentication and encrypted data transmission, are implemented to ensure user privacy and data integrity.
- **Improved Message Management**: Comprehensive message management features, including reliable history management and efficient search capabilities, enhance the efficiency of information retrieval and review processes.
- **Accessibility and User-Friendliness**: Designed with a user-centric approach, the application provides a user-friendly interface accessible to users of varying technical expertise, promoting widespread adoption and usability.

5. Scope and Limitations

This section offers a clear overview of what the project aims to achieve and the potential challenges it may face.

5.1. Scope:

The Chat Web Application holds significant importance and can benefit individuals and organizations in the following ways:

1. **Professional Communication**: Employees and teams can utilize the application to facilitate real-time communication and collaboration, enhancing productivity and coordination within organizations.
2. **Educational Settings**: Students and educators can use the platform for instant messaging and collaboration on academic projects, improving the learning experience and engagement.
3. **Personal Use**: Individuals can enjoy seamless and secure messaging for personal communication, ensuring a reliable and enjoyable chatting experience.

5.2. Limitations:

While the Chat Web Application offers a range of useful features, it also has certain limitations that users should be aware of:

1. **Scalability Constraints**: Although designed to handle a growing number of users, performance may be affected by extreme traffic spikes or very high volumes of data.
2. **Browser Compatibility**: The application's performance and appearance may vary depending on the browser used, potentially leading to inconsistencies across different platforms.
3. **Feature Limitations**: As the application focuses on real-time messaging and security, some advanced features like file sharing or multimedia integration may be limited in the initial

6. Literature Study/Review

To get ready for this project, we looked into several topics to learn more about what we need to do next. Studying these materials helped us understand more about the project.

6.1. Similar Works

Several existing projects and open-source solutions offer functionalities similar to those of the Chat Web Application, focusing on real-time messaging and secure communication:

Rocket.Chat: An open-source team communication platform that provides real-time chat, file sharing, and integrations with various tools. It supports private and public channels, similar to the features of the Chat Web Application. [1]

Mattermost: An open-source messaging platform for team communication and collaboration. It offers real-time messaging, customizable notifications, and integrations with other services, focusing on security and scalability.[2]

Zulip: An open-source team chat application that supports real-time messaging and threaded conversations. It focuses on efficient message organization and integration, offering features relevant to the Chat Web Application.[3]

Chatwoot: An open-source customer support and engagement tool that offers real-time messaging and multi-channel support. It includes features for conversation history and user management, aligning with the objectives of the Chat Web Application.[4]

Matrix (Element): An open-source, decentralized chat protocol with real-time messaging and Teastrong focus on security and interoperability. It offers scalable and secure communication features akin to those of the Chat Web Application.[5]

6.2. Similar Concepts

Telegram: Provides real-time messaging with a focus on security and encrypted communications. Its features for instant message delivery and secure chats reflect the core concepts of your application.

Signal: An app emphasizing strong privacy and security with end-to-end encrypted real-time messaging. Its approach to secure communication and user privacy mirrors our objectives.

Chime: Provides real-time messaging and user authentication, built using the MERN stack. It shares similarities in technology stack and real-time communication features with the Chat Web Application.

Wire: A secure messaging app offering end-to-end encryption and real-time communication. Its emphasis on user privacy and secure messaging aligns with the security features of your application.

Flock: A team communication tool with real-time messaging and collaboration features. It offers secure communication and integrates with various productivity tools, reflecting the collaborative aspects of your chat application.

7. Methodology

This part explains how we plan to develop our project, covering both the steps we'll follow to create the software and the technical setup we'll use.

7.1. Software Development Life Cycle

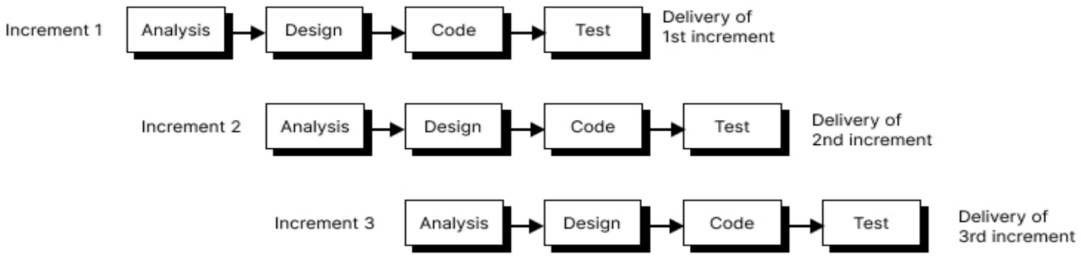

Fig 1: Incremental Model

We follow an increment model for this project and work on the core features first then create the rest around it.The development of the chat web application follows an incremental model, structured into three key phases:

- **First Increment: UI/UX, Analysis and Design**

  UI/UX Design**:** Focus on designing an intuitive and responsive user interface using Tailwind CSS and the DaisyUI Library. This includes creating wireframes and mockups for the chat interface and user interactions.

  Analysis and Design**:** Define the requirements and architecture of the application. This includes setting up the project structure, selecting libraries and tools, and designing the data schema.

- **Second Increment: Feature Implementation**

  Implementation of Core Features**:** Develop the main functionalities of the application including user authentication, real-time messaging, message display, and file sharing. This involves coding the backend with Express.js and Node.js, and the frontend with React.js.

  Integration of Real-time Communication**:** Implement WebSocket technology for real-time messaging capabilities to ensure instant communication between users.

- **Third Increment: Deployment & Integration**

  Deployment**:** Prepare the application for deployment using cloud services such as MongoDB Atlas for data storage. Configure the server and frontend for production use.

  Integration and Testing**:** Integrate all components and conduct thorough testing to identify and fix any issues. This includes end-to-end testing of user flows and real-time features.

  This model also emphasizes the importance of documentation throughout the development lifecycle, ensuring clarity, maintainability, and supportability of the web-based chat application from planning through deployment and beyond.

7.2. Software Specifications

The web application aims to facilitate real-time messaging and collaboration among users. The software components are as follows:

Backend Technology Stack:

- **Programming Language:** JavaScript (Node.js) - Utilized for server-side scripting and backend development.
- **Framework:** Express.js - A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
- **Database:** MongoDB (via MongoDB Atlas) - A scalable NoSQL database service designed for modern applications, offering flexibility and performance.

Frontend Technology Stack:

- **Framework/Library:** React.js - A JavaScript library for building user interfaces, enabling efficient rendering and seamless updates.
- **Styling:** Tailwind CSS - A utility-first CSS framework for rapid UI development, providing a customizable and responsive design approach.
- **UI/UX Design:** Figma - A collaborative interface design tool used for creating and prototyping user interfaces with intuitive design capabilities.
- **Build Tool:** Vite - A build tool that provides a faster and leaner development experience by serving source files over native ES modules for optimized deployment.

Development Tools:

- **IDE:** Visual Studio Code - A lightweight and powerful code editor with built-in support for debugging, syntax highlighting, and Git integration.
- **Version Control:** Git - A distributed version control system for tracking changes in source code during software development.
- **Diagramming:** PlantUML - A tool that allows users to create UML diagrams from plain text descriptions, enhancing documentation and system design clarity.
- **Additional Diagramming Tool:** draw.io - An online diagramming tool used for creating flowcharts, wireframes, and other visual representations to aid in project design and documentation.

Real-time Communication:

- **WebSocket Technology:** Socket.io - Used for real-time messaging capabilities, ensuring instant communication between users.

Additional Tools and Libraries:

- **Authentication:** Custom authentication solutions to ensure secure user access and privacy.
- **File Handling:** Integrated support for file sharing, allowing users to send and receive files within the chat interface.

8. System Model And UML Diagram

This section introduces the project's system model and UML diagrams. The system model shows how the components interact with each other, and the UML diagrams use standardized symbols to give a visual overview. These tools help explain the project's structure, function, and how its parts are connected, making it easier to understand and communicate about the project.

8.1. Use Case Diagram

A use case diagram depicts the interactions between users (actors) and a system to achieve specific goals. It shows the relationships between actors and use cases, highlighting the functionalities provided by the system from the user's perspective.

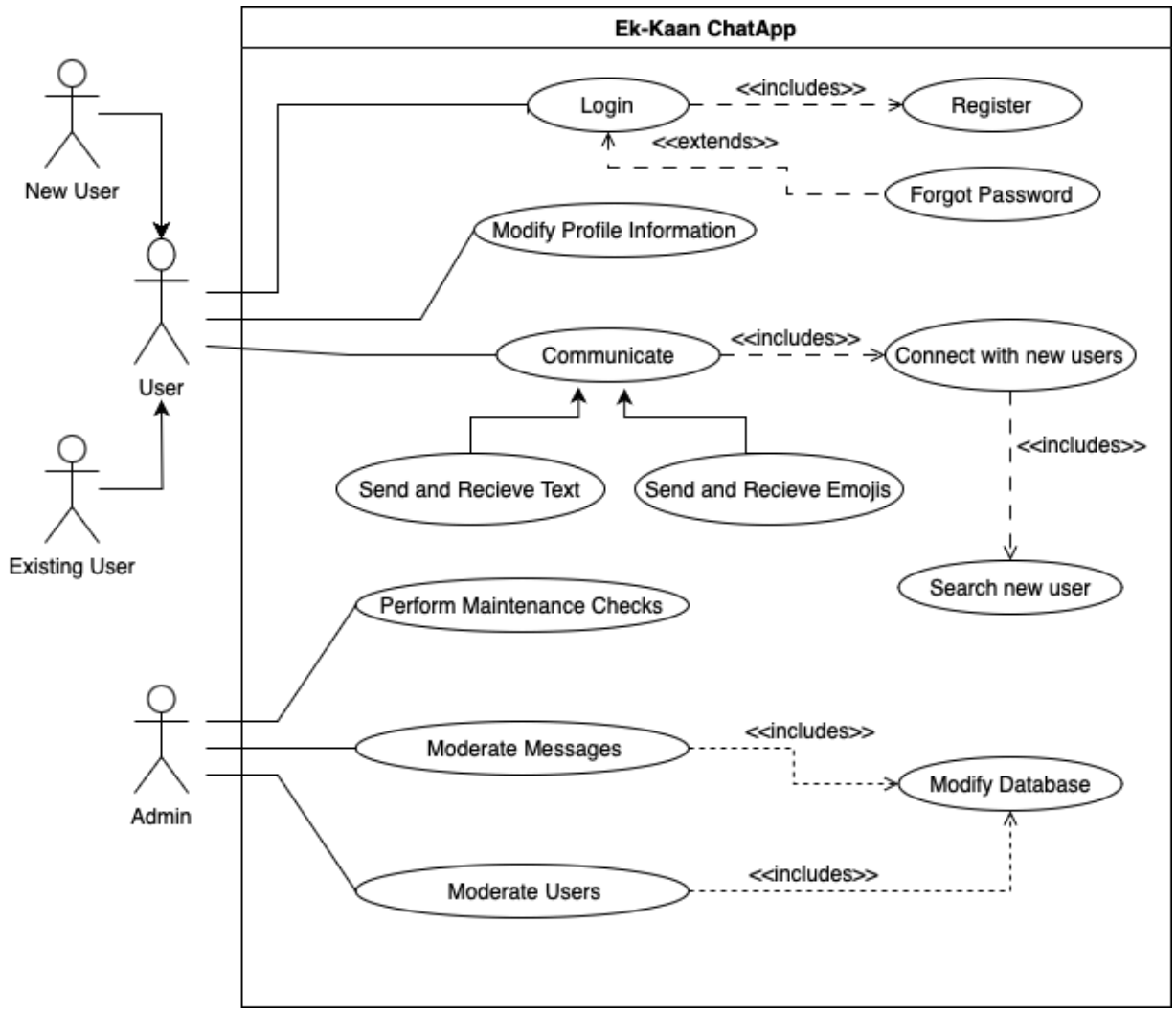

Fig 2: Use Case Diagram

8.2. Activity Diagram

An activity diagram illustrates the flow of control within a system or a specific process. It shows the sequence of activities, actions, and decisions involved in completing a task or achieving a goal. It's useful for modeling business processes or the logic of algorithms.

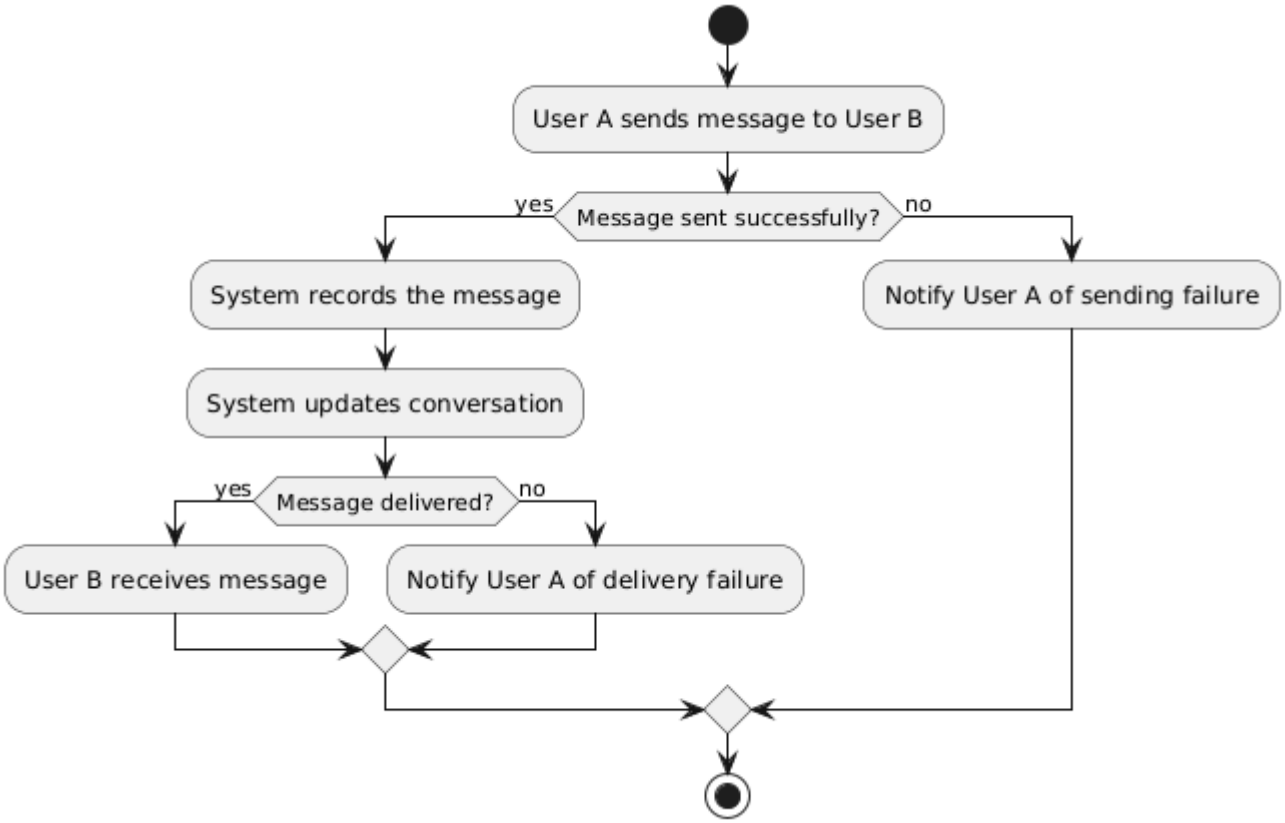

Fig 3: Activity Diagram

8.3. Entity Relationship (ER) Diagram

An ER diagram is used in database design to illustrate the logical structure of a database, focusing on the entities (objects or concepts) within the system and their relationships. It helps visualize how data entities relate to each other.

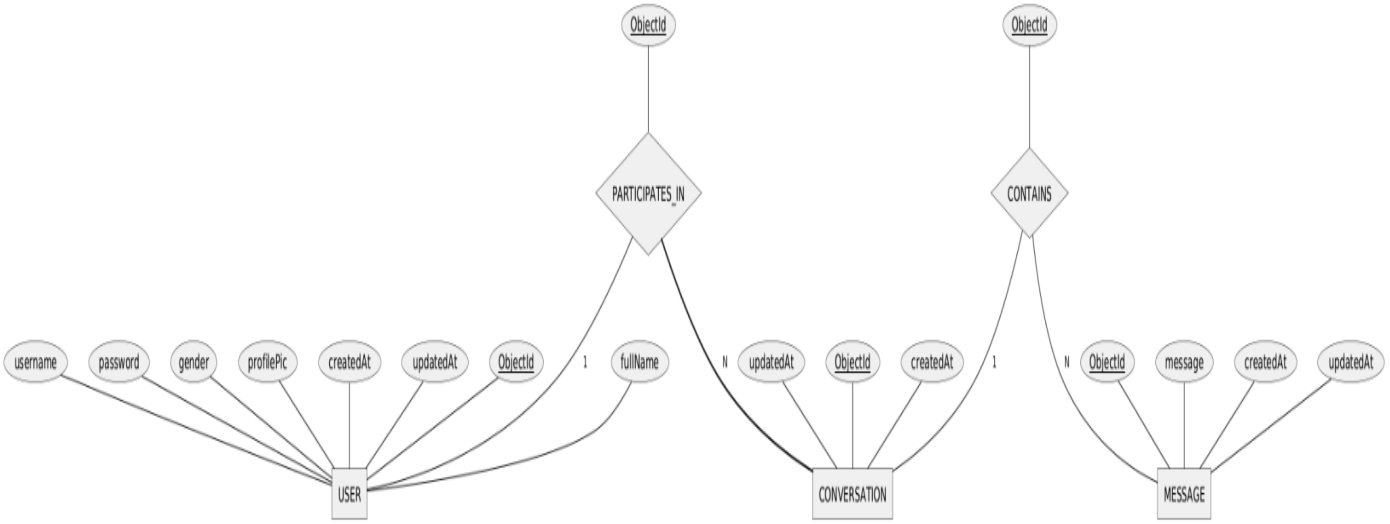

Fig 4: ER Diagram

8.4. Component Diagram

A component diagram shows the organization and dependencies among software components in a system. It depicts the physical components (like executable files, libraries, etc.) and their relation- ships, providing a high-level view of the architecture.

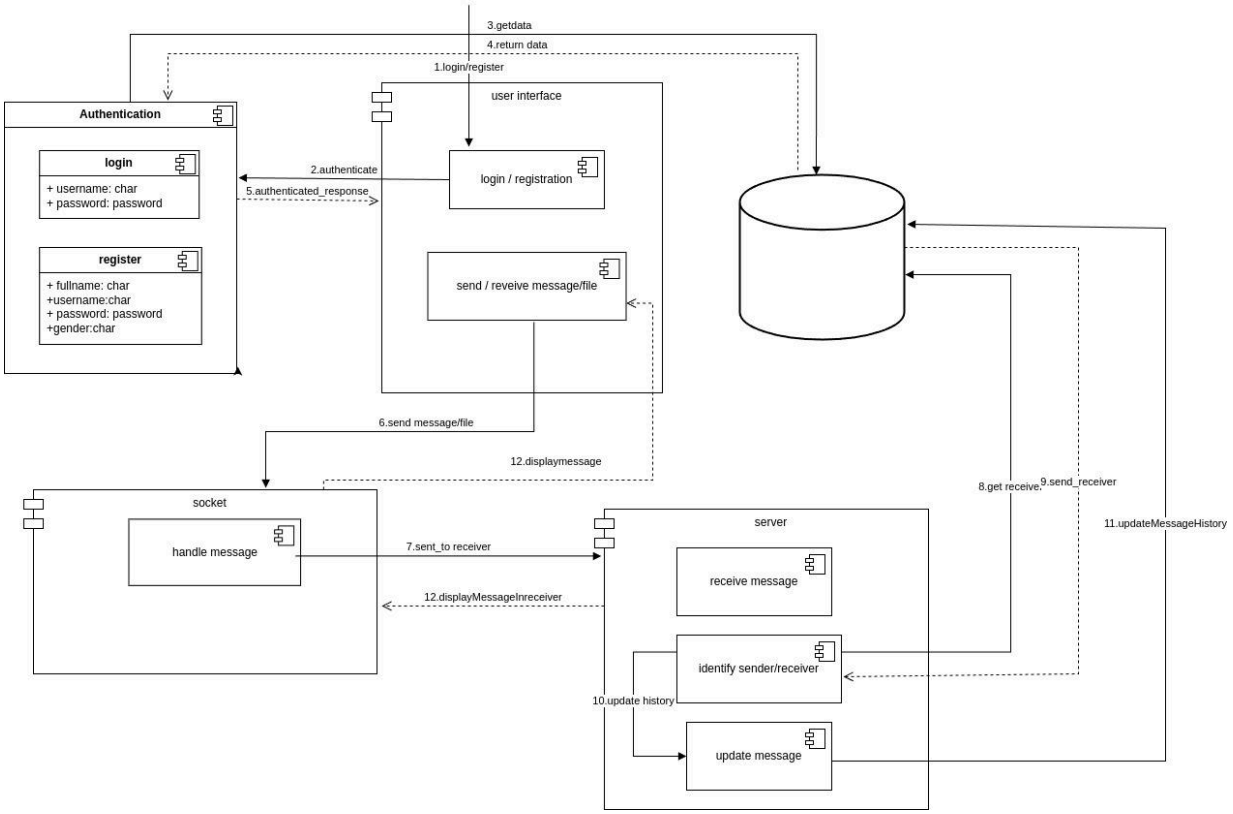

Fig 5: Component Diagram

8.5. System Sequence Diagram

A system sequence diagram (SSD) represents the sequence of interactions between external actors (users or other systems) and a system to accomplish a specific functionality. It emphasizes the mes- sage flows and their order between objects over time.

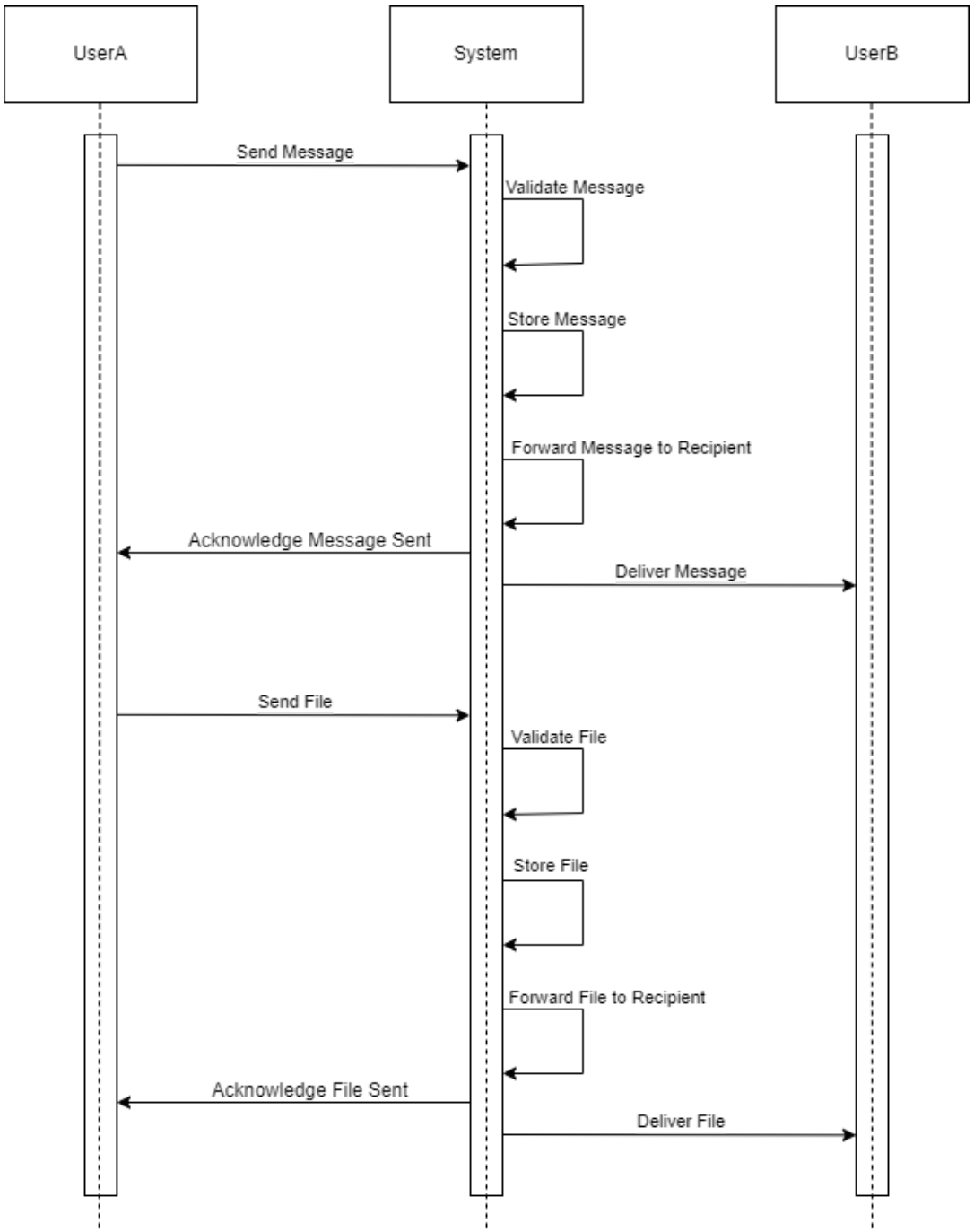

Fig 6: System Sequence Diagram

8.6. Class Diagram

A class diagram illustrates the structure and relationships of classes within a system. It shows the static design of a system, including classes, attributes, methods, and their associations. Class dia- grams are fundamental for object-oriented modeling.

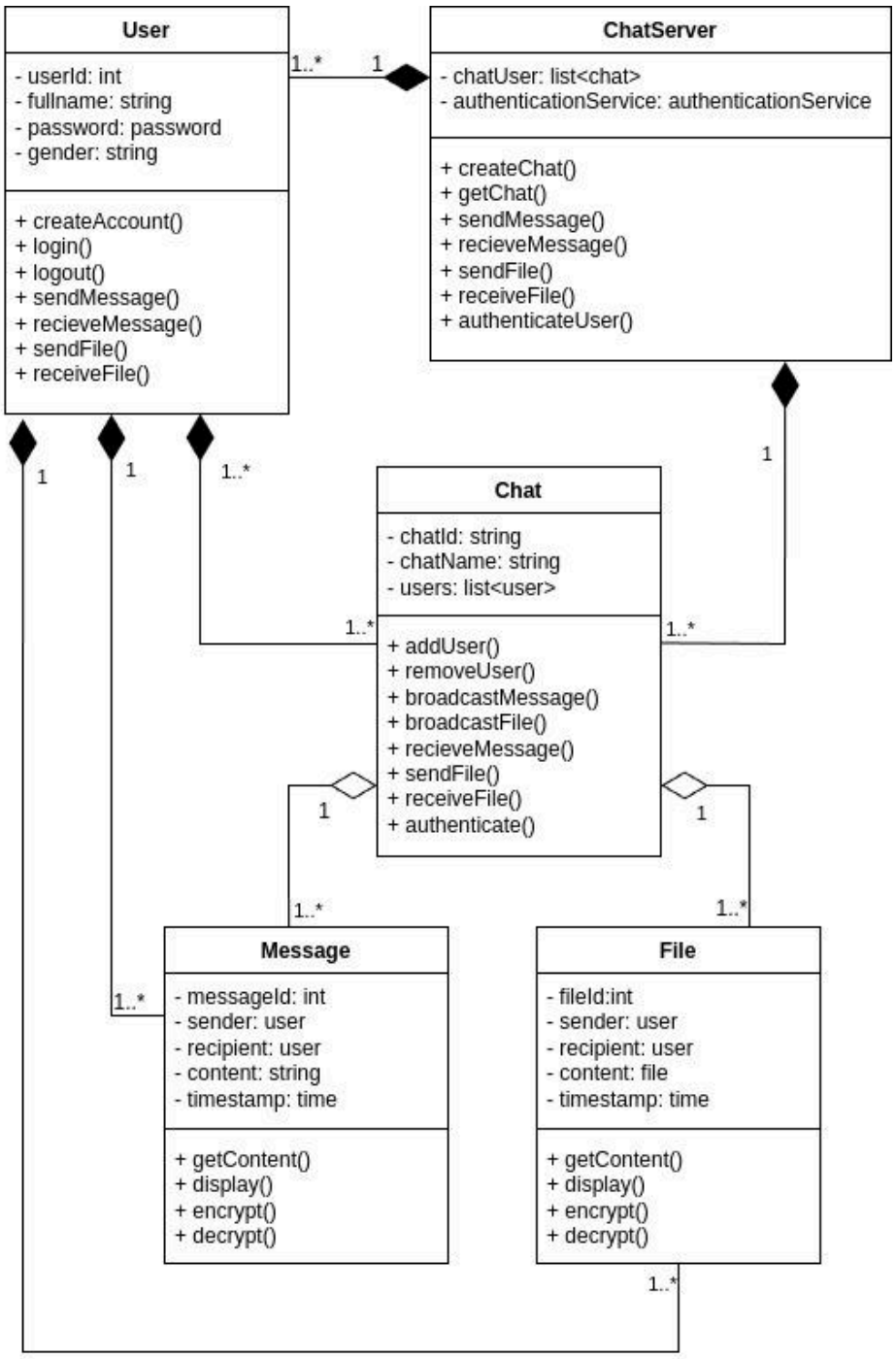

Fig 7: Class Diagram

9. Tasks Done So Far

These are the tasks we've finished, showing what we've achieved in our project.

1. **Frontend Design**
   - Built responsive, visually appealing chat app interfaces with Tailwind CSS and DaisyUI.
2. **User Authentication**
   - Developed secure user registration and login functionality.
   - Implemented server-side and client-side validation.
3. **Real-time Messaging**
   - Implemented WebSocket technology for real-time user communication.
   - Ensured instant message delivery and updates without page reloads.
4. **Timestamp Implementation**
   - Incorporated timestamps for each message.
5. **File Sharing Implementation**
   - Enabled file upload and sharing.
6. **Message Persistence**
   - Stored messages in MongoDB for retrieval and display in conversations.
   - Ensured messages persist across user sessions and device restarts.
7. **Error Handling**
   - Integrated robust error handling for both frontend and backend.
   - Displayed user-friendly error messages for common issues.

10. Task Remaining

Here are the tasks yet to be completed, outlining what remains to be accomplished in our project.

1. **Message Sorting:** Sorting messages for maintaining an organized and user-friendly conversation history. This involves ensuring that messages are displayed in a logical sequence, typically based on their timestamps, to facilitate coherent and chronological communication.

11. Result and Discussion

11.1. Result

- Successfully developed a responsive web chat application featuring a user-friendly interface and secure registration and login functionalities.
- Implemented real-time messaging utilizing WebSocket technology, ensuring accurate timestamps are displayed for each message.
- Integrated comprehensive file-sharing functionalities, enabling users to upload, share, and retrieve files securely within the chat application.

11.2. Discussion

- The application architecture employs WebSocket technology for real-time message passing and utilizes MongoDB for message storage. This setup ensures that messages are persisted across user sessions and device restarts, enabling seamless retrieval and display in conversations.
- Mongoose, integrated with Node.js, is utilized to set a default value for a timestamp field, ensuring that each document is automatically assigned the current date and time upon creation.
- Future enhancements could focus on implementing group chat functionality and developing an administrative profile panel.

12. Proposed Deliverable

The final deliverables for the Simple Chat Web App project will encompass the following:

- **Chat Functionality:** Complete implementation of real-time messaging with WebSocket integration.

- **File-Sharing Feature:** Addition of a robust file upload and sharing capability, with secure storage solutions.

- **User Interface Enhancements:** Development of a responsive and intuitive user interface utilizing Tailwind CSS and DaisyUI.

- **Security Measures:** Implementation of secure user authentication, message encryption, and protection against web vulnerabilities.

- **Documentation and Testing:** Provision of comprehensive documentation, thorough testing to ensure reliability, and preparation for deployment.

13. Project Task and Time Schedule

The following table and Gantt chart shows how the project tasks are distributed, scheduled, and assigned to team members:

13.1. Team Responsibility

Shahil Dhakal	API, Database and Documentation
Drishya Neupane	UI/UX Design, Backend and Documentation
Sandilya Silwal	Frontend, Graphics Designing and Documentation

13.2. Project Increments

	Increment 1	Increment 2	Increment 3
Frontend	Set up basic layout, routing, and static pages.	Added dynamic components, integrate with backend APIs	Implemented advanced features, finalize styling.
Backend	Initial Setup, API Routes, Authentication	Implemented core functionality and real-time features.	Optimized performance, handled deployment.
Testing & QA	-	Tested individual components and features.	Conduct comprehensive testing and address issues.
Deployment & Maintenance	-	Prepare for deployment (staging environment).	Deploy to production, monitor, and maintain.

The incremental process of the project is as follows:

Increment 1: Frontend Development

- Design and implement UI components
- Integrate core chat features with authentication and routing
- Set up real-time messaging UI and connect with backend services

Increment 2: Feature Implementation

- Develop and deploy API endpoints for authentication, messaging, and user management
- Set up WebSocket server for real-time communication
- Implement file handling, storage, and database integration

Increment 3: Deployment and Integration

- Perform unit and integration testing of frontend and backend
- Conduct end-to-end testing for complete user flows
- Optimize performance, security, and finalize deployment configurations

13.3. Gantt Charts

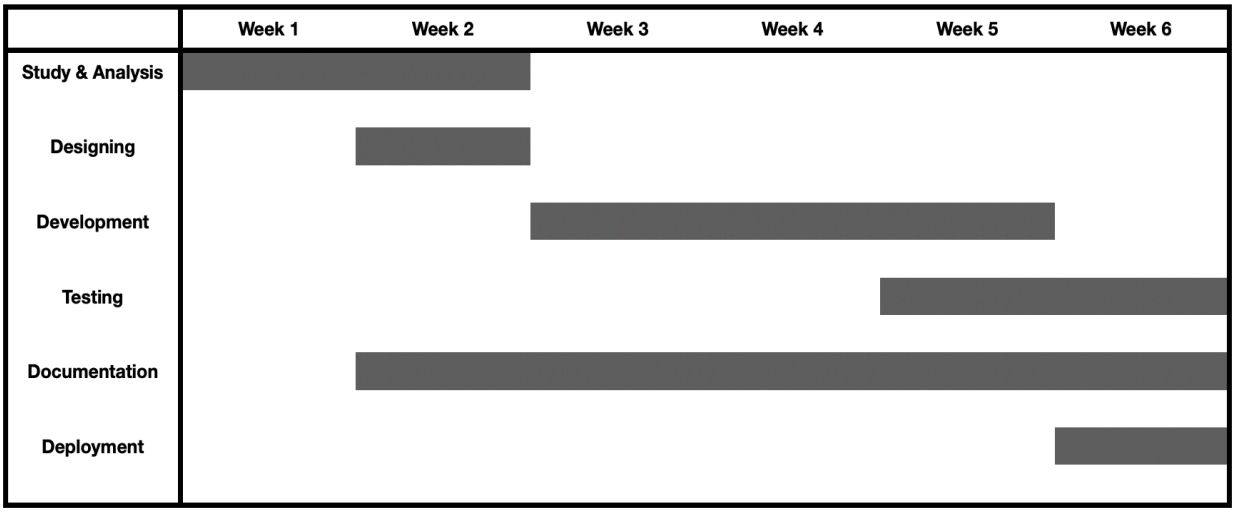

Fig 8: Increment 1 Gantt Chart

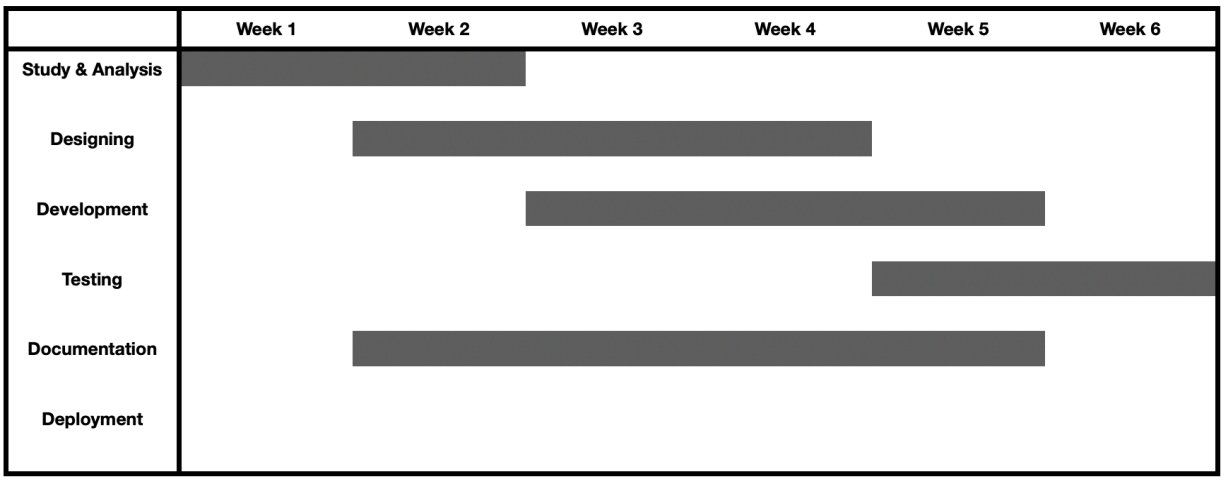

Fig 9: Increment 2 Gantt Chart

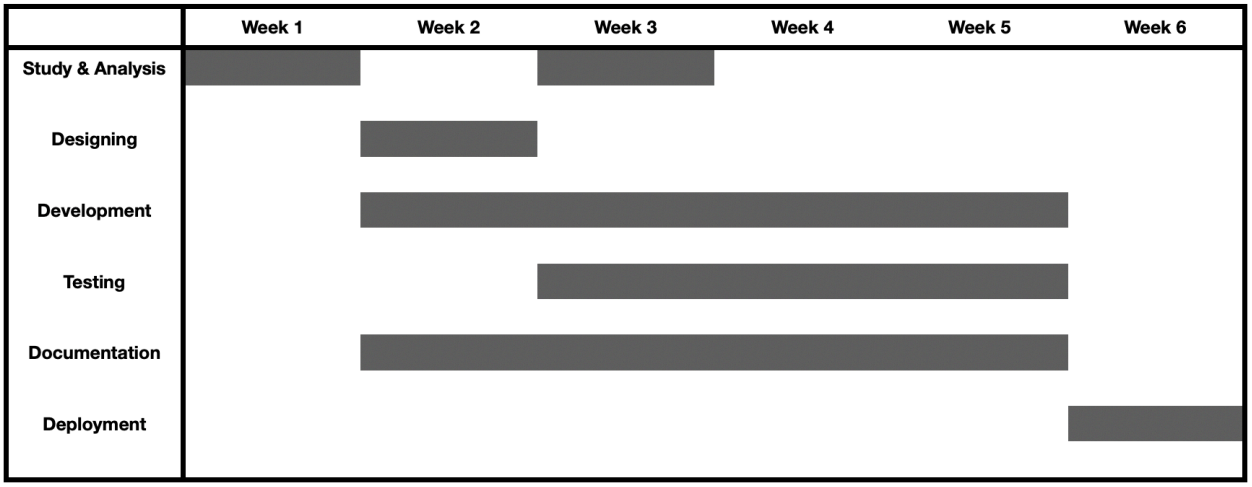

Fig 10: Increment 3 Gantt Chart

References

[1] Massochin, M. (2022). Why choose Rocket.Chat for your open source chat tool.

[2] Mattermost Team. (2019). *Developer's Alternative to Slack*.

[3] R. Barde and A. A. Raouf, "Evaluating and Improving Accessibility of Web Applications: Zulip Chat Case Study," *2021 17th International Computer Engineering Conference (ICENCO)*, Cairo, Egypt, 2021, pp. 112-117, doi: 10.1109/ICENCO49852.2021.9698944.

[4] Shandily, H. (n.d.). *What if we could use Email Signatures across all channels?*

[5] Branscombe, M. (2021, Jan 21). Element: Secure messaging for tech-savvy organizations.