***SOFTWARE PROJECT FINAL REPORT***

Prabesh Mishra, Shreejil Patel, and Jimmy Hopkins

12/5/2024

Table of Contents

List of Figures

List of Tables

**1. Introduction**

The General Chatting Site is a real-time messaging platform built to make communication easier for both individuals and teams. Whether for personal chats or professional collaboration, the platform provides a secure and reliable way to exchange messages. Key features, such as private and group chats, real-time notifications, and secure login, make it a great tool for fast and effective communication. The platform uses modern technologies to deliver quick message delivery, customizable user settings, and the ability to grow as more users join.

Key Features:

* Group and Private Chat Options: Support for team discussions and one-on-one conversations.
* Real-Time Messaging: Instant updates powered by Socket.IO for smooth communication.
* Notifications: Alerts for new messages and activities to keep users informed.
* Profile Customization: Options to personalize themes and settings for a better user experience.
* End-to-End Encryption: Protects private chats to keep them secure and confidential.
* Secure Data Storage: Uses MySQL to store user data and chat history safely, with regular backups to prevent data loss.

Use Cases:

* Team Collaborations: A shared space for group discussions and project coordination.
* Private Conversations: A safe and private platform for individual messaging.

Technologies Used:

* Socket.IO: A library for real-time, two-way communication.
* Node.js: A backend technology for building fast and scalable services.
* MySQL: A flexible and reliable database for managing users and messages.

The General Chatting Site focuses on creating a platform that can grow with its users. With an easy-to-use interface built using HTML, CSS, and JavaScript, and a powerful backend for performance and reliability, the platform is designed to handle modern communication needs while staying user-friendly.

**2. Project Management Plan**

The development of the General Chatting Site was a collaborative effort, with all team members contributing to every aspect of the project. Each team member took on specific roles and responsibilities based on their skills and interests, ensuring a balanced and efficient workflow.

Team Collaboration:

* Prabesh: Led presentation preparation and documentation, while contributing to development and testing.
* Shreejil: Focused on backend development with Node.js, MySQL, and Socket.IO, ensuring scalability and assisting with debugging.
* Jimmy: Designed and implemented the frontend using HTML, CSS, and JavaScript, ensuring responsiveness and integration with the backend.

The team used the Agile methodology, allowing them to build the project step by step and add features continuously. To handle potential risks, they included end-to-end encryption to protect data and performed load testing with horizontal scaling to manage heavy server traffic. The project used a cloud server for reliable storage and processing. Key technologies included Node.js, MySQL, Socket.IO, and HTML/CSS/JavaScript, ensuring the system is scalable, secure, and efficient.

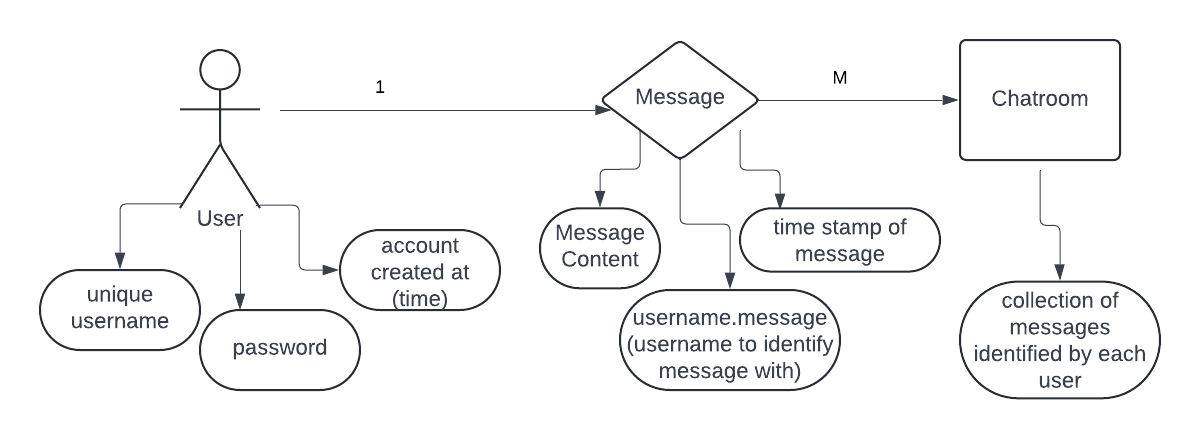
The project deliverables were completed according to the planned schedule, covering milestones such as backend and frontend integration, testing, and final deployment

3. Requirement Specifications

End Users: Individuals and teams who require real-time communication, both personal and professional.

System Administrators: Responsible for maintaining the platform, monitoring performance, and ensuring security.

Developers: Stakeholders involved in system upgrades and feature enhancements.



Caption

Login and Authentication:

* Actor: End User.
* Description: Users log in securely using credentials stored in the system.
* Precondition: The user must have a valid account.
* Postcondition: User gains access to their dashboard.

Send Message:

* Actor: End User.
* Description: The user sends a message in real-time to another user or group.
* Precondition: The user must be logged in.
* Postcondition: Message is delivered instantly.

Group Chat:

* Actor: End User.
* Description: The user participates in group discussions, sending and receiving messages.
* Precondition: The user must be part of a group.
* Postcondition: Messages are synchronized for all group members.

Search History:

* Actor: End User.
* Description: The user searches for messages or files in chat history.
* Precondition: The user is logged in and has existing messages.
* Postcondition: Relevant results are displayed.

The general Chatting Site is designed to meet critical non-functional requirements, ensuring reliability, security, and scalability.

**4. Architecture**

The General Chatting Site is built using a Client-Server Architecture with an MVC (Model-View-Controller) design. The backend, created with Node.js and Socket.IO, manages key tasks like user login, message handling, and database operations. The frontend, made with HTML, CSS, and JavaScript, provides an easy-to-use interface and connects to the backend using Socket.IO for real-time updates. Data is safely stored in MySQL database, which ensures fast access and the ability to handle growth.

This setup keeps the responsibilities of the system organized, making it easier to scale and maintain. The Client-Server model handles communication efficiently, Socket.IO powers instant messaging, and MySQL provides secure and flexible data storage. Cloud infrastructure ensures there’s enough processing power and storage to support real-time features.

**5. Design**

The UI is simple and responsive, making it easy for users to navigate. Key features include:

* Dashboard: Shows active group and private chats with real-time updates.
* Chat Window: Allows users to send and receive messages with timestamps.
* Profile Customization: Lets users update themes and settings to personalize their experience.
* Notifications: Alerts users about new messages and activities

Components Design:

* Frontend: Built with HTML, CSS, and JavaScript, it handles user interactions and communicates with the backend using REST APIs and Socket.IO for real-time updates.
* Backend: Built with Node.js and Socket.IO, it manages system logic, processes requests, and connects to the database.
* Database: MySQL stores user profiles, chat messages, and group details securely.  
  Dynamic processes include real-time messaging and secure login authentication.

Dynamic workflows include real-time message processing and secure login authentication.

Database Design:

* Users Collection: Stores user information (ID, credentials, profile, preferences).
* Messages Collection: Logs sender, receiver, content, and timestamps.
* Groups Collection: Tracks group details and members.

The modular design ensures scalability, performance, and security. Socket.IO enables real-time messaging. These choices align with the system goals for efficient communication and user management.

Traceability to Requirements:

* Real-time messaging → Socket.IO for instant updates.
* User authentication → Secure backend logic and encrypted database storage.
* Scalability → MySQL with flexible indexing.
* User customization → Frontend integrated with backend settings.

The design effectively meets all functional and non-functional requirements.

**6. Test Management**

The testing process for the General Chatting Site focused on verifying key functionalities. Test cases included checking successful login with valid credentials, error messages for invalid credentials, enforcing a time limit after sending a message, updating profile settings, applying customized themes, and retrieving accurate search results. Testing methods such as Boundary Testing (e.g., login credentials and search queries), Equivalence Partitioning (testing different user roles), and Scenario-Based Testing (simulating real-world actions like messaging and customization) ensured thorough coverage.

Test cases covered all critical use cases, confirming that the system met its functional and non-functional requirements. During testing, a few defects were found: profile settings not saving for specific themes and duplicate results in search history. These issues were resolved by fixing backend validation logic and improving database query indexing. Overall, the testing confirmed that the platform was functional, reliable, and ready for use.

**7. Conclusions**

The General Chatting Site was successfully developed as a functional platform for real-time communication, including features like private and group chats, real-time messaging, notifications, profile customization, and secure data management. While the core goals were achieved, there is room to make the platform better by improving its usability and overall quality.

The project taught us the value of teamwork, step-by-step development, and thorough testing. We also learned that better planning for user experience and design could have made the website more polished. Working on scalable and real-time features gave us valuable experience in handling complex system requirements.

Future improvements could include adding voice and video chat, smarter notifications, and file-sharing capabilities. The user interface can be made more attractive and seamless, and advanced features like AI chat suggestions or third-party tool integrations could be implemented. With these upgrades, the platform can become a more complete and powerful communication tool.

**References**

https://www.w3schools.com/howto/howto\_css\_login\_form.asp https://www.geeksforgeeks.org/how-to-handle-authentication-in-node/https://www.geeksforgeeks.org/web-socket-in-node-js/