ChatWave – Real-Time Web Chat Application
Software Requirements Specification
Version <1.0>

Name : Md.Minhajul Islam
ID :2211022042
Section:09
Group :10

Revision History

Date	Version	Description	Author
<03/08/25>	<1.0>	SRS 1.0	Group-10

Table of Contents

Software Requirements Specification

1. Introduction

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire document, including its purpose, scope, definitions, acronyms, references, and an outline of the system requirements. The aim of this document is to analyze and define the ChatWave web chat application, ensuring a clear understanding of the system's features, capabilities, and requirements.This document will serve as a guide for developers, testers, and stakeholders to understand the ChatWave system, its technical stack, functional requirements, and non-functional constraints. Additionally, it will outline the key functionalities such as real-time messaging, authentication, error handling, and deployment strategies

1.1 Purpose

The purpose of this document is to define and document the software requirements for ChatWave, a real-time web chat application. This SRS outlines the system's functional and non-functional requirements, ensuring a clear understanding for developers, testers, and stakeholders. ChatWave aims to provide seamless and secure real-time communication using MERN (MongoDB, Express.js, React, Node.js), Socket.io, JWT authentication, Zustand for state management, and TailwindCSS for UI design. The document details the technical stack, core functionalities like messaging, user authentication, group chats, and online status tracking, as well as security, performance, and deployment considerations. By defining these aspects, the SRS serves as a structured guide throughout the Software Development Life Cycle (SDLC), ensuring a smooth development process and alignment with project goals.

1.2 Scope

Primarily, the scope pertains to the ChatWave application, which focuses on enabling real-time communication between users through a modern web-based chat platform. The project aims to provide seamless, secure, and scalable messaging with features such as one-on-one and group chats, real-time online status, media sharing, and secure authentication using JWT.

This SRS is designed to define the software requirements for the development of ChatWave, ensuring clarity in the system's functionalities, technical architecture, and security protocols. Additionally, it can serve as a reference for selecting suitable frameworks, tools, and deployment strategies. The document does not mandate a specific development methodology or framework but provides a structured guideline for implementation, ensuring consistency and maintainability.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
MERN	MongoDB, Express.js, React, Node.js – Full-stack JavaScript framework
JWT	JSON Web Token – Secure method for authentication & authorization
WebSockets	Protocol enabling real-time, bidirectional communication between clients and servers
Socket.io	JavaScript library for implementing WebSockets in real-time applications
Zustand	Lightweight state management library for React
TailwindCSS	Utility-first CSS framework for building responsive UIs
Daisy UI	Component library built on top of TailwindCSS
NoSQL	Database approach used in MongoDB for flexible data storage

1.4 References

The references for this project include:

- ✓ MERN Stack Documentation
- ✓ Socket.io Official Documentation
- ✓ MongoDB Atlas Security & Deployment Guide
- ✓ TailwindCSS & DaisyUI Official Documentation
- ✓ JWT Authentication Best Practices
- ✓ WebSockets Security & Implementation Guide

1.5 Overview

The remaining sections of this document provide a detailed description of the ChatWave system, including its users, functionalities, technical architecture, and constraints. Section 2 offers a general overview of the project, discussing the product's purpose, operating environment, user classes, and design constraints.Section 3 outlines the functional requirements, external interface requirements, data handling, and security considerations. It also details the specific system features, including authentication, real-time messaging, group chats, error handling, and deployment strategies. This section serves as a technical guide for development and implementation.Finally, Section 4 provides supporting information, such as database schema, system architecture diagrams, API documentation, and deployment strategies. These details ensure that ChatWave is developed efficiently, securely, and in line with the project's objectives.

2. Overall Description

This section provides a high-level description of ChatWave, including its purpose, functionality,

users, operating environment, constraints, and dependencies. The section outlines the problem the system aims to solve, the key stakeholders, and the essential features that define the platform.

ChatWave is designed as a real-time web chat application that enables seamless and secure communication between users. Built using the MERN stack, it integrates Socket.io for real-time messaging, JWT for authentication, Zustand for state management, and TailwindCSS for a modern UI. The system supports one-on-one and group chats, online/offline status tracking, multimedia sharing, and real-time notifications.

2.1 Product Perspective

ChatWave is an independent, web-based chat application designed to facilitate instant communication**.** It serves as a self-contained solution but can also be extended to integrate with other platforms via REST APIs and WebSockets.

The system follows a client-server architecture, where:

1. The frontend (React) communicates with the backend (Node.js + Express.js) via REST APIs and WebSockets.
2. The backend interacts with MongoDB Atlas to store user data, messages, and authentication credentials.
3. Socket.io enables real-time updates between users.
4. Zustand ensures efficient state management on the frontend.

This architecture ensures scalability, flexibility, and security**,** making ChatWave suitable for both small-scale and large-scale deployments.

2.2 Product Functions

ChatWave includes the following core functionalities:

1. User Authentication & Authorization – Secure JWT-based authentication for login/logout.
2. Real-Time Messaging – Instant message delivery using WebSockets (Socket.io)**.**
3. Group Chats **–** Users can **create** and participate in group conversations.
4. Online/Offline Status – Real-time presence tracking.
5. Media Sharing – Users can send images, videos, and documents.
6. Read Receipts & Message Status – Indicate when a message is delivered and read.

7. Profile Management – Users can update their profile picture, username, and status.
8. Error Handling & Notifications – Real-time error handling and notifications for new messages and events.
9. Secure Message Storage – Messages stored securely using MongoDB Atlas with encryption**.**

2.3 User Classes and Characteristics

User Class	Description
Registered Users	Users who create an account and use the chat functionalities, including messaging, profile updates, and media sharing.
Guest Users (Future Scope)	Users who can temporarily chat without full registration (optional feature).
Administrators	Users with elevated permissions to manage user accounts, monitor chats, and enforce community guidelines.

2.4 Operating Environment

ChatWave is a web-based application designed to run on:

1. Browsers: Chrome, Firefox, Edge, Safari
2. Operating Systems: Windows, macOS, Linux
3. Backend Server**:** Node.js + Express.js
4. Database: MongoDB Atlas (NoSQL)
5. Real-Time Communication: WebSockets via Socket.io
6. Deployment Platforms: Vercel, Render, Railway (for free hosting)

2.5 Design and Implementation Constraints

1. The application must be fully responsive for desktop and mobile users.
2. WebSockets must be supported by the user's browser.
3. Authentication must use JWT with secure token storage**.**
4. The backend must be scalable, handling high-concurrency messaging efficiently**.**
5. MongoDB must store messages securely with proper encryption and indexing for fast retrieval.

2.6 Assumptions and Dependencies

1. MongoDB Atla**s** is used as the primary cloud database.
2. A stable internet connection is required for real-time messaging.
3. The system assumes users have modern browsers that support WebSockets.
4. The backend server must be hosted on a cloud platform to ensure high availability.
5. Future enhancements might include mobile apps (React Native) **or** third-party API integrations**.**

3. Specific Requirements

The specific requirements are –

3.1 Functionality

This section describes the core functional requirements of the ChatWave real-time chat application. The functionalities are designed to ensure seamless communication, user security, and efficient system performance.

Each function is organized into subsections, covering key features such as authentication, real-time messaging, group chats, and profile management. These functionalities are implemented using MERN stack, Socket.io for WebSockets, JWT for authentication, Zustand for state management, and TailwindCSS for UI design.

3.1.1 User Authentication & Authorization

- The system shall allow users to register and log in securely using JWT authentication.
- The system shall hash passwords before storing them in the database.
- The system shall support OAuth authentication (Google, GitHub, etc.) (Future Scope).
- The system shall allow users to reset their passwords via email verification.
- The system shall automatically expire JWT tokens after a defined period to enhance security.

3.1.2 Real-Time Messaging

- The system shall use Socket.io to deliver messages instantly between users.
- The system shall allow users to send text messages, emojis, images, and files.
- The system shall enable message encryption for secure communication.
- The system shall store messages in MongoDB to allow chat history retrieval.

- The system shall mark messages as "delivered" and "seen".
- The system shall allow users to delete messages (for themselves or for everyone).

3.1.3 Group Chat

- The system shall allow users to create, join, and leave group chats.
- The system shall allow group admins to add or remove members.
- The system shall allow group members to send text, images, and files.
- The system shall display the list of active members in a group.
- The system shall support group mentions (@username) to notify specific users.
- The system shall allow users to mute group notifications.

3.1.4 Online & Offline User Status

- The system shall track and display user online/offline status in real-time.
- The system shall update the status automatically when a user logs in or out.
- The system shall allow users to set their status to "Do Not Disturb" (DND).
- The system shall indicate "last seen" timestamps for offline users.

3.1.5 Profile Management

- The system shall allow users to edit their username, profile picture, and status message.
- The system shall allow users to view other users' profiles.
- The system shall support uploading images for profile pictures.

3.1.6 Error Handling & Notifications

.

- The system shall log and handle errors on both frontend and backend.
- The system shall display real-time notifications for:
    - New messages
    - Online/offline status changes
    - System updates or errors
- The system shall show error messages when message delivery fails.

3.1.7 Media & File Sharing

- The system shall allow users to send images, videos, and documents.
- The system shall compress large media files for faster transmission.
- The system shall preview sent images and videos within the chat window.
- The system shall support cloud storage (e.g., Cloudinary, Firebase) for media files.

3.1.8 Message Search & History

- The system shall allow users to search for messages by keyword.
- The system shall allow users to filter messages by sender or date.
- The system shall load older messages using infinite scrolling.

3.1.9 Global State Management

- The system shall use Zustand for efficient state management.
- The system shall maintain chat state, user authentication status, and notifications globally.

3.1.10 Security & Data Protection

- The system shall encrypt all communication using SSL/TLS.
- The system shall store passwords securely using bcrypt hashing.
- The system shall restrict API access using role-based authentication.
- The system shall implement rate limiting to prevent spamming.
- The system shall support two-factor authentication (2FA) (Future Scope).

3.1.11 User Blocking & Reporting.

- The system shall allow users to block contacts to prevent further messages.
- The system shall provide an option to report inappropriate messages.
- The system shall allow administrators to review and take action on reported users.

3.1.12 Multi-Device Support

- The system shall allow users to log in from multiple devices simultaneously.
- The system shall sync messages across devices in real-time.
- The system shall allow users to log out remotely from other devices.

3.1.13 Free Deployment & Hosting

- The system shall be deployed on free hosting platforms like Vercel, Render, or Railway.
- The system shall support automatic deployment using GitHub Actions (CI/CD).
- The system shall allow developers to self-host the application using Docker.

3.2 Usability

This section describes the usability requirements of ChatWave, ensuring that the application is user-friendly, accessible, and provides a seamless experience across different devices. The system is designed with a modern and responsive UI, utilizing TailwindCSS and DaisyUI for a clean and intuitive interface.

3.2.1 Graphical User Interface (GUI)

- The system shall provide a visually appealing and consistent user interface using TailwindCSS and DaisyUI.
- The system shall have a responsive design, ensuring it works on desktop, tablet, and mobile devices.
- The system shall include a dark mode option for better user experience.(future scope)
- The system shall use icons and toolbars for improved navigation.
- The system shall provide an intuitive chat layout, displaying messages in a familiar bubble-style format.
- The system shall allow users to customize their chat themes.

3.2.2 Accessibility

- The system shall be keyboard navigable, allowing users to interact without a mouse.
- The system shall support screen reader compatibility, ensuring visually impaired users can navigate the platform.
- The system shall provide high contrast mode for users with visual impairments.

3.3 Reliability & Availability

This section defines the reliability and availability requirements for ChatWave, ensuring that the system remains operational, fault-tolerant, and resilient under various conditions. The platform is designed to handle high concurrency, prevent data loss, and provide real-time communication with minimal downtime**.**

3.3.1 Server Uptime & Failover Handling

- The system shall provide 99.9% uptime, ensuring high availability.
- The system shall implement automatic server restarts in case of unexpected crashes.
- The system shall use load balancing techniques to distribute traffic efficiently.
- The system shall have automatic reconnection mechanisms if a user loses internet connectivity.
- The system shall utilize WebSocket heartbeats to detect and recover lost connections.

3.3.2 Database Backup & Recovery

- The system shall perform automatic daily backups of MongoDB data.
- The system shall support point-in-time recovery to prevent permanent data loss.
- The system shall ensure that messages and user data are replicated across multiple database nodes.
- The system shall recover unsent messages if a user reconnects after going offline.

3.4 Performance

This section outlines the performance requirements for ChatWave, ensuring that the system delivers fast, efficient, and scalable real-time messaging. The system is designed to handle multiple concurrent users, maintain low latency, and optimize resource usage.

3.4.1 Response Time

- The system shall deliver messages within 100-300 milliseconds under normal network conditions.
- The system shall load chat history within 2 seconds for recent messages**.**
- The system shall ensure that user authentication (login/logout) completes within 1 second**.**
- The system shall display online/offline status updates within 500 milliseconds.

3.4.2 Scalability & Load Handling

- The system shall support 1000+ concurrent users without significant performance degradation.
- The system shall optimize database queries using indexes for faster data retrieval.
- The system shall automatically scale backend resources to handle high traffic.
- The system shall implement lazy loading for chat history to reduce initial load time.
- The system shall efficiently manage WebSocket connections to prevent server overload.

3.5 Security

This section defines the security requirements for ChatWave, ensuring that user data, messages, and authentication processes are protected against unauthorized access, breaches, and cyber threats. The system follows industry best practices for secure authentication, data encryption, and API protection.

3.5.1 Data Transfer Security

- The system shall use SSL/TLS encryption for all data transfers between the client and server.
- The system shall establish secure WebSocket connections to prevent unauthorized message interception.
- The system shall encrypt all messages in transit to ensure data privacy.

3.5.2 Data Storage Security

- The system shall encrypt sensitive user data (passwords, authentication tokens, etc.) before storing it in the database.
- The system shall hash passwords using bcrypt with a strong salting mechanism.
- The system shall store JWT tokens securely, preventing token leakage and replay attacks.
- The system shall implement role-based access control (RBAC) to restrict unauthorized actions.

3.5.3 Authentication Security

- The system shall use JWT-based authentication to verify user identities.
- The system shall enforce token expiration and renewal policies to prevent misuse.
- The system shall block multiple failed login attempts to prevent brute-force attacks.
- The system shall implement two-factor authentication (2FA) (Future Scope).

3.6 Supportability

This section describes the supportability requirements for ChatWave, ensuring that the system is maintainable, extensible, and easy to debug. It outlines the necessary tools, monitoring strategies, and documentation required for long-term support.

3.6.1 API Documentation

- The system shall provide comprehensive API documentation for developers.
- The documentation shall include REST API endpoints, WebSocket event descriptions, request/response formats, and authentication guidelines.
- The system shall support Swagger or Postman for interactive API testing.

3.6.2 Logging & Monitoring

- The system shall log all critical errors and warnings on both the client and server**.**
- The system shall store logs securely in a database or log management system (e.g., Logstash, Elasticsearch, or a cloud logging service).
- The system shall support real-time monitoring using tools like Prometheus, Grafana, or a cloud-based monitoring service**.**
- The system shall send alerts for critical errors to administrators via email or a notification system.

3.6.3 Extensibility & Maintainability

- The system shall follow modular code architecture to allow easy enhancements.
- The system shall support third-party API integrations for features like voice/video calls (Future Scope).
- The system shall ensure that frontend and backend components can be updated independently.

3.7 Design Constraints

This section outlines the design constraints for ChatWave, specifying the limitations and restrictions that must be considered during development. These constraints ensure that the system is efficient, scalable, and aligned with the chosen technology stack.

3.7.1 Technology Stack

- The system shall be developed using the MERN (MongoDB, Express.js, React, Node.js) stack.
- The system shall use Socket.io for real-time WebSocket communication.

- The frontend shall be built using React.js with TailwindCSS and DaisyUI.
- The backend shall be implemented using Node.js with Express.js.
- The system shall use MongoDB Atlas as the primary database.

3.7.2 Deployment Constraints

- The system shall be deployed on free hosting platforms such as Vercel, Render, or Railway.
- The system shall be optimized for low-cost cloud hosting with scalable options.
- The backend shall be containerized using Docker (Future Scope) to allow easy deployment across cloud services.
- The system shall support continuous integration and deployment (CI/CD) using GitHub Actions.

3.7.3 Web & Mobile Compatibility

- The system shall **be** optimized for modern web browsers (Chrome, Firefox, Edge, Safari).
- The system shall follow responsive design principles to ensure usability on desktop, tablets, and mobile devices**.**
- The system shall support progressive web application (PWA) features (Future Scope)**.**

3.7.4 Performance & Scalability Limitations

- The system shall limit the maximum file upload size to prevent server overload.
- The system shall use lazy loading for message history to improve performance.
- The system shall optimize WebSocket connections to handle high user concurrency efficiently.

3.8 On-line User Documentation and Help System Requirements

This section outlines the requirements for providing user documentation and support to ensure that users can effectively navigate and utilize ChatWave. The system shall offer comprehensive online help resources, FAQs, and troubleshooting guides to assist users in resolving common issues.

3.8.1 Online Help System

- The system shall provide an integrated help center accessible from the chat interface.

- The help center shall include step-by-step guides on user registration, messaging, group chats, and profile management**.**
- The system shall provide contextual tooltips and pop-ups to guide new users.

3.8.2 Frequently Asked Questions (FAQs)

The system shall include an FAQ section covering common queries, including:

- How to create an account and log in?
- How to send messages and create group chats?
- How to reset passwords and update profile information?
- How to report or block users?
- How to troubleshoot connection issues?

3.8.3 User Feedback and Support

- The system shall allow users to submit feedback or report bugs via a feedback form.
- The system shall provide an automated chatbot or support email option for additional assistance.
- The system shall maintain a troubleshooting section for resolving login, messaging, and connectivity issues**.**

3.8.4 Multi-Language Support (Future Scope)

- The system shall support multiple languages to cater to a global audience (Future Scope).
- The system shall allow users to select their preferred language in the settings menu.

3.9 Purchased Components

This section lists the third-party libraries, frameworks, and services that ChatWave depends on for development, security, and deployment. While ChatWave primarily uses open-source technologies, some external services and APIs are integrated to enhance its functionality.

3.9.1 Open-Source Libraries & Frameworks

The following open-source libraries and frameworks are used in the development of ChatWave**:**

Frontend:

- React.js – JavaScript framework for building user interfaces (MIT License**).**
- Zustand – Lightweight state management library for React **(**MIT License**).**
- TailwindCSS – Utility-first CSS framework for styling **(**MIT License**).**
- DaisyUI – UI component library based on TailwindCSS (MIT License**).**

Backend:

- Express.js – Web framework for Node.js **(**MIT License**).**
- Mongoose – ODM for MongoDB to handle database operations **(**MIT License**).**
- Socket.io – Enables real-time WebSocket communication **(**MIT License**).**
- jsonwebtoken (JWT) – Used for secure authentication **(**MIT License**).**
- bcrypt – Library for password hashing **(**MIT License**).**

3.9.2 Cloud Services & APIs

ChatWave relies on the following external cloud services and APIs for hosting, storage, and security**:**

- MongoDB Atlas – Cloud-based NoSQL database service (Free tier available).
- Cloudinary (Optional) – For storing and serving media files (Freemium model).
- Vercel / Render / Railway – Cloud platforms used for free frontend and backend deployment**.**
- GitHub Actions (Optional) – For automated CI/CD pipelines.

3.9.3 Future Third-Party Integrations (Future Scope)

In future updates, ChatWave may integrate:

- Twilio / Vonage – For adding voice and video calling features**.**
- Firebase / AWS S3 – For enhanced media file storage**.**
- Google OAuth / GitHub OAuth – For social login options.

3.10 Interfaces

This section describes the interfaces required for ChatWave, including user interfaces, hardware interfaces, software interfaces, and communication interfaces. These interfaces ensure smooth interaction between users, the system, and external services.

3.10.1 User Interfaces

- The system shall provide a modern and responsive web interface built with React.js, TailwindCSS, and DaisyUI.
- The chat interface shall display real-time messages, user status, and notifications.
- The system shall include a user profile page where users can edit their name, profile picture, and status.
- The system shall provide a login and registration page with JWT-based authentication.
- The system shall support dark mode and accessibility settings.
- The interface shall be optimized for both desktop and mobile devices.

3.10.2 Hardware Interfaces

- The system shall run on any device with an internet connection and a modern web browser.
- The system shall support desktop, laptop, tablet, and mobile devices.
- The system shall require a stable internet connection for real-time communication.

3.10.3 Software Interfaces

- Frontend (React.js) shall interact with Backend (Node.js + Express.js) via REST APIs and WebSockets.
- The system shall use MongoDB Atlas for storing user data, messages, and authentication credentials.
- The backend shall use Mongoose as an ODM (Object Data Modeling) tool for database operations.
- The system shall integrate with Cloudinary or Firebase (Optional) for storing images and files.
- The system shall use GitHub Actions (Optional) for CI/CD automation.

3.10.4 Communications Interfaces

- The system shall use WebSockets (via Socket.io) for real-time messaging and status updates.
- The system shall use HTTPS for secure API communication.
- The system shall use JWT authentication for secure login and authorization.
- The system shall send email notifications (Future Scope) for password resets and account verification.

3.11 Licensing Requirements

This section defines the licensing requirements for ChatWave, ensuring compliance with open-source licenses and any third-party software usage. Since ChatWave is built using open-source technologies, all dependencies must follow their respective license agreements.

3.11.1 Open-Source Licenses Used

ChatWave incorporates several open-source libraries under permissive licenses such as MIT, Apache 2.0, and BSD. The following components are used:

Frontend Libraries

- React.js – MIT License
- Zustand – MIT License
- TailwindCSS – MIT License
- DaisyUI – MIT License

Backend Libraries

- Node.js & Express.js – MIT License
- Socket.io – MIT License
- Mongoose (MongoDB ODM) – MIT License
- jsonwebtoken (JWT for authentication) – MIT License
- bcrypt (Password hashing) – MIT License

Deployment & Cloud Services

- MongoDB Atlas – Free-tier available, subject to MongoDB Inc. terms.
- Vercel / Render / Railway – Free hosting, subject to provider terms.
- Cloudinary (Optional, for media storage) – Free-tier available, subject to Cloudinary terms.

3.11.2 Compliance Requirements

- The ChatWave project shall respect all third-party licens**es** and properly acknowledge open-source contributions.
- Any modifications to open-source libraries shall be documented and comply with their respective licenses.

- If ChatWave integrates paid third-party services in the future, the licensing terms must be reviewed before implementation.

3.12 Legal, Copyright, and Other Notices

This section defines the legal, copyright, and other notices related to the ChatWave application, ensuring that intellectual property rights, trademarks, and legal disclaimers are properly addressed.

3.12.1 Copyright and Ownership

- The ChatWave application, including its source code, UI design, and documentation, is the intellectual property of the development team or organization responsible for its creation.
- Any unauthorized reproduction, modification, or distribution of ChatWave's code or assets without proper authorization is prohibited.

3.12.2 Open-Source Compliance

- ChatWave utilizes several open-source libraries (React, Express, Socket.io, TailwindCSS, etc.), which are governed by their respective licenses (e.g., MIT License).
- The project shall not violate the terms of any third-party software license agreements**.**

3.12.3 Trademarks and Branding

- "ChatWave" and its logo (if applicable) are trademarks of the project and cannot be used without proper attribution.
- Any third-party logos or branding used in the application (e.g., OAuth providers like Google, GitHub) must follow their brand usage guidelines.

3.12.4 Disclaimer

- ChatWave is provided "as is" without any warranties, whether express or implied.
- The development team is not responsible for any data loss, security breaches, or downtime caused by external factors.
- Users are responsible for complying with data protection laws (such as GDPR) when using the application**.**

3.13 Applicable Standards

This section outlines the industry standards and best practices that ChatWave follows to ensure security, performance, and compatibility with modern web technologies. Adhering to these standards helps maintain high-quality software development, data protection, and regulatory compliance.

3.13.1 Security Standards

- OWASP Top 10 Security Guidelines – The system shall follow OWASP (Open Web Application Security Project) best practices to mitigate common security vulnerabilities such as SQL injection, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), and authentication flaws**.**
- JWT Authentication Best Practices **–** The system shall implement secure token-based authentication with short-lived tokens and refresh tokens**.**
- SSL/TLS Encryption **–** All data transmissions between the client and server shall be encrypted using TLS 1.2 or higher**.**
- GDPR Compliance (Future Scope) **–** If ChatWave stores user data for extended periods, it must comply with data protection laws such as GDPR and CCPA**.**

3.13.2 Web Development Standards

- HTML5, CSS3, ECMAScript 6+ (ES6+) – The frontend shall be developed using modern web technologies for better performance and compatibility.
- WebSockets Standard (RFC 6455) **–** The system shall implement WebSockets using Socket.io, following th**e** IETF WebSocket Protocol (RFC 6455) for real-time communication.
- REST API Standards **–** The system shall follow RESTful principles for structuring API endpoints, handling HTTP requests, and maintaining stateless communication**.**
- Accessibility Standards (WCAG 2.1) **–** The UI shall follow Web Content Accessibility Guidelines (WCAG 2.1**)** to ensure usability for all users, including those with disabilities.

3.13.3 Performance & Scalability Standards

- MongoDB Indexing & Query Optimization **–** The system shall use indexes to optimize database queries and reduce response times.
- Web Performance Optimization (WPO) Best Practices **–** The frontend shall use lazy loading, caching, and code splitting to improve page load speed**.**
- Scalability Best Practices **–** The backend shall be stateless and support horizontal scaling for handling increased traffic.

3.13.4 Code Quality & Version Control

- Code Formatting & Linting **–** The project shall follow ESLint and Prettier standards to maintain consistent code style**.**
- Git Version Control – The system shall use GitHub for source code management with proper branching strategies (e.g., main, dev, feature branches).
- CI/CD Best Practices – The system shall follow Continuous Integration and Deployment (CI/CD**)** methodologies using GitHub Actions or similar tools**.**

4. Supporting Information

This section provides additional technical details, including the database schema, system architecture, API documentation, and deployment strategy.

4.1 **Database Schema**

An overview of the MongoDB database structure, including collections for users, messages, and groups**.**

4.**2 System Architecture Diagram**

A high-level representation of the client-server architecture, including frontend, backend, database, and WebSocket communication.

4.3 **API Documentation**

A list of REST API endpoints for authentication, messaging, and user management.

4.4 **Deployment Strategy**

A description of the hosting platforms (Vercel, Render, Railway) and database management (MongoDB Atlas).