Software Requirements Specification

for

Harmony

Version 1.0 approved

Prepared by Khalid Qureshi, Harsh Mishra, Pankaj Parihar, Pratham Patil

Thadomal Shahani Engineering College

August 11, 2025

Table of Contents

Table of Contents ii
Revision History ii
1. Introduction 2
1.1 Purpose 2
1.2 Document Conventions 2
1.3 Project Scope 2
1.4 References 2
2. Overall Description 2
2.1 Product Perspective 2
2.2 User Classes and Characteristics 2
2.3 Operating Environment 2
2.4 Design and Implementation Constraints 2
2.5 Assumptions and Dependencies 2
3. System Features 2
3.1 System Feature 1 2
3.2 System Feature 2 (and so on) 2
4. Data Requirements 2
4.1 Logical Data Model 2
4.2 Data Dictionary 2
4.3 Reports 2
4.4 Data Acquisition, Integrity, Retention, and Disposal 2
5. External Interface Requirements 2
5.1 User Interfaces 2
5.2 Software Interfaces 2
5.3 Hardware Interfaces 2
5.4 Communications Interfaces 2
6. Quality Attributes 2
6.1 Usability 2
6.2 Performance 2
6.3 Security 2
6.4 Safety 2
6.5 [Others as relevant] 2

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

This introduction provides an overview of the Software Requirements Specification (SRS) for the Harmony project. It outlines the purpose, scope, and organization of the document.

1.1 Purpose

The basic intention of this document is to provide a clear, comprehensive, and unambiguous specification for the "Harmony" real-time chat application. It serves as a foundational agreement between the developer and the project evaluator, detailing exactly what the system will do. This document is intended for multiple readers: for the **project instructor**, it is a basis for evaluation and grading; for the **developer**, it is a technical blueprint for implementation; and for any other interested party, it provides a complete understanding of the project's goals, features, and constraints.

1.2 Document Conventions

- **Bold** headings are used for section titles.
- *Italics* is used whenever a definition is given.
- Requirements are expressed in clear, concise sentences, and critical terms are defined in the glossary.
- All requirements are uniquely identifiable for reference.

1.3 Project Scope

From a management perspective, Harmony is an initiative to develop a modern, web-based communication platform that enables users to connect and converse in real time. In layman's terms, Harmony is a private chat application, similar to a simplified version of Discord or Slack, where users can create accounts, set up topic-based chatrooms, and exchange instant messages. The primary goal is to create a seamless and engaging user experience built entirely on open-source technologies.

The major features included in this version are:

- Secure User Account Creation and Login
- Real-Time Instant Messaging
- Dynamic Chatroom Creation and Management
- Message Editing and Deletion
- Role-Based Moderation Controls (User vs. Administrator)

1.4 References

- [1] IEEE Std 830-1998, *Recommended Practice for Software Requirements Specifications*. [Online]. Available: https://ieeexplore.ieee.org/document/720574

- [2] PubNub, "How Do Chat Apps Work?: An overview of chat application architecture," [Online]. Available: https://www.pubnub.com/blog/how-do-chat-apps-work/

- [3] Microsoft, "Microsoft Teams: An existing commercial product for team collaboration and chat," [Online]. Available: https://www.microsoft.com/en-in/microsoft-teams/group-chat-software

- [4] Slack, "Slack: A popular platform for channel-based messaging," [Online]. Available: https://slack.com/

- [5] Discord, "Discord: A VoIP and instant messaging social platform," [Online]. Available: https://discord.com/

2. Overall Description

2.1 Product Perspective

Harmony is designed as a standalone, feature-rich messaging application that provides users with a fluid and immediate communication channel. The system facilitates instantaneous, low-latency conversations through dynamically managed chatrooms, giving users control over their content with message editing and deletion capabilities. It also incorporates a necessary layer of moderation through distinct user and administrator roles, ensuring that communities can be managed effectively. The entire platform is a showcase of modern web development practices using a full-stack open-source technology set.
(Same features but worded better)

2.2 User Classes and Characteristics

This application has two primary classes of users who interact with it differently:

1. **Standard User:** This is the most common user of the application. After creating an account and logging in, their primary interaction is joining chatrooms and communicating. They can view the list of existing rooms, join any of them, and start sending messages. They have control over their own contributions, with the ability to edit a message to fix a typo or delete it entirely. A standard user can also create new chatrooms, inviting others to join a

conversation on a new topic.

2. **Administrator:** This user has all the capabilities of a Standard User, but with additional privileges for platform moderation. An administrator's main interaction, beyond standard chatting, is to maintain the health and order of the community. If a message violates community guidelines, the administrator can delete it, regardless of who sent it. If a chatroom becomes redundant or problematic, an administrator has the authority to remove the entire room and all its associated messages from the system.

2.3 Operating Environment

Users access the Harmony application through a standard web browser on any internet-connected device, such as a laptop, desktop, or mobile phone. The application is entirely cloud-based. The frontend, a responsive web application, is served to the user's browser from Vercel's global edge network. The backend logic and APIs run as serverless functions on the same Vercel platform, which communicates with a cloud-hosted MongoDB Atlas database cluster. This architecture ensures high availability and low latency for users regardless of their geographical location.

2.4 Design and Implementation Constraints

Client-Side Minimum Requirements:

- **Hardware:** Any device with at least 2GB of RAM and a modern processor capable of running a current web browser.
- **Browser:** The latest stable version of Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari. The browser must have JavaScript and WebSockets enabled.

Developer Environment Versioning:

- **Node.js:** Version 18.x or later.
- **npm:** Version 9.x or later.
- **React:** Version 18.x or later.
- **TypeScript:** Version 5.x or later.

3. System Features

3.1 System Feature 1 (User Registration)

This feature handles all aspects of user identity. New users can create an account by providing a unique username and a secure password. The system ensures the username is not already taken. Returning users can log in using their credentials. Upon successful login, the system provides the user with a secure session token (JWT) that is used to authenticate all subsequent actions, ensuring that only logged-in users can access the chat functionalities. A logout feature is also provided to securely end the session.

3.2 System Feature 2 (Login)

This is the core feature of Harmony. Within a chatroom, users can send messages that appear instantly for all other participants in the same room. This is achieved through a persistent WebSocket connection managed by Socket.IO, which allows for bidirectional, low-latency data flow. Users are also given control over their own messages after sending; they can edit a message to correct information or delete a message they no longer wish to be visible.

3.3 System Feature 3 (Chatroom Management)

To organize conversations, Harmony allows any authenticated user to create new chatrooms. A user simply provides a name for the new room, and it immediately becomes available for others to join. The application will display a list of all available chatrooms, allowing users to easily browse and enter different conversations.

3.4 System Feature 3 (Administrative Moderation)

This feature provides essential community management tools to privileged users. An Administrator has the power to oversee the entire platform. They can delete any message from any user in any chatroom, which is critical for removing inappropriate or harmful content. Furthermore, an Administrator can permanently delete an entire chatroom if it is no longer needed or violates platform rules.

4. Data Requirements

4.1 Logical Data Model

The system's data will be stored in a MongoDB database using three primary collections:

- **Users:** This collection stores a document for each user, containing their unique ID, username, a securely hashed password, and their assigned role (e.g., 'user' or 'admin').
- **Rooms:** This collection maintains a list of all available chatrooms. Each document includes the room's unique ID, its name, and a timestamp of its creation.
- **Messages:** This collection stores every message sent on the platform. Each message document contains a unique ID, the message content, references to the user who sent it and the room it belongs to, and a timestamp.

5. External Interface Requirements

5.1 User Interfaces

The system will provide a single Graphical User Interface (GUI) rendered in a web browser. The interface will be designed to be clean, responsive, and intuitive, with clear visual cues for navigating between rooms, sending messages, and accessing user controls. It will adapt its layout for optimal viewing on both desktop and mobile screens.

6. Quality Attributes

6.1 Usability

The application shall be highly intuitive and easy to use. A new user should be able to understand the full functionality—from registration to chatting—without requiring a tutorial or help document. The user flow shall be logical, and all interactive elements will be clearly labeled.

6.2 Performance

The system must feel responsive and fast to the user. Real-time message delivery latency between clients in the same region should be under 500 milliseconds. The server architecture must be capable of supporting up to 500 concurrent users without noticeable degradation in performance. API response times for actions like login should be under 1 second.

6.3 Security

Security is a critical requirement. User passwords must be salted and hashed using a strong algorithm like bcrypt. All data transmitted between the client and server must be encrypted using TLS (via HTTPS and WSS). The system must also incorporate server-side validation and sanitization of all user inputs to protect against common web vulnerabilities like Cross-Site Scripting (XSS).