SOFTWARE REQUIREMENTS SPECIFICATION

for

Wyrd

Version 1.0

Prepared by: Elijah Philip
December 27, 2024

Contents

1 Introduction

Wyrd is an application that allows users from around the globe to chat with each other individually, or with groups. The app is designed using rust for its performance so users can seamlessly communicate without any latency issues and its reliability due to its memory safety features. The app is available on Windows, macOS, and Linux, ensuring accessibility for a broad audience of PC users.

1.1 Purpose

This document serves as a comprehensive guide outlining the technical requirements, functional specifications, and performance benchmarks needed to construct the Wyrd chat application. The application aims to enable seamless, real-time communication across the globe, leveraging Rust's performance and memory safety features for a fast, reliable, and secure user experience.

1.2 Project Scope

The goal of this project is to develop Wyrd, a cross-platform chat application enabling seamless communication via text, voice, and file sharing. The key features include:

1. **Core Communication**
    - Real-time text and voice communication.
    - File sharing capabilities for various formats.
2. **User Management**
    - Account registration and login.
    - Account management (e.g., profile updates, password changes).
3. **Enhanced Messaging Features**
    - Group chat creation and management.
    - Search functionality for previous messages and contacts.
    - User presence indicators (online/offline status).
    - Typing indicators to show when users are typing.
    - Message formatting (e.g., bold, italics, custom fonts).
    - Notifications, even when the app is closed.
    - Read receipts to know what time the message was sent and if it was read.

1.3 Target Audience

This document is intended for:

- **Developers and Maintainers**: To outline the technical and functional goals of Wyrd and provide a roadmap for its development.
- **End Users**: To give an overview of the application's intended features and how it aims to improve communication.
- **System Administrators**: To guide the setup and maintenance of the application infrastructure.

The primary audience is the developer (myself), as this document will help focus on achievable goals and functionalities during the application's creation.

1.4 Definitions and Acronyms

2 Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

The application runs on Windows, Mac, and Linux. It is not supported by browsers.

2.1.2 User Interfaces

The application GUI provides menus, buttons, textboxes, scrollbar, panes, containders, and grids allowing for easy control by a keyboard and a mouse.

2.1.3 Hardware Interfaces

1. Supported Platforms
    -
2. Input Devices
    - Keyboard and mouse (required for interaction with the user interface).
    - Microphone (optional, required for voice chat functionality).
3. Output Devices
    - Monitor or screen (to display the application's graphical interface).
    - Speakers or headphones (optional, required for voice chat and notification sounds).

2.1.4 Software Interfaces

The Wyrd application interacts with various software systems to deliver its features efficiently and securely:

1. OS APIs
    - Windows, macOS, and Linux APIs are used for notifications, file handling, and window management to provide a seamless user experience across platforms
2. Frameworks and Libraries

- The user interface is developed using the Tauri framework for lightweight, cross-platform compatibility. Networking and real-time communication are handled via Tokio, while message and file encryption utilize the RustCrypto library.

3. Third-Party APIs
   - Sendgrid is used to handle user email verification and password resets.
   - Google, Microsoft, and Github Authentication are used as alternative ways for users to log in or sign up.
4. Database
   - Postgre SQL is used as the database to store past message, and personal information.

2.2 User Classes and Characteristics

Wyrd is designed for the following user classes:

- **End Users:** These users value ease of use and expect core features like text and voice chat, file sharing, and group management to work reliably across platforms.
- **System Administrators:** Technically skilled users responsible for maintaining the system in server-hosted environments. They require tools for monitoring, security, and deployment management.

2.3 Product Functions

2.4 Operating Environment

2.5 Design

2.6 Assumptions and Dependencies

2.7 Constraints

2.8 Risk Analysis

3 System Features and Requirements

3.1 Description and Priority

3.2 Functional Requirements

3.3 Nonfunctional Requirements

3.4 Data Flow Requirements

3.5 Interface Requirements

4 Other Nonfunctional Requirements

4.1 Performance Requirements

4.2 Security Requirements

4.3 Software Quality Attributes

4.4 Business Rules

5 Other Requirements

5.1 Legal and Regulatory Requirements

5.2 Hardware Requirements

5.3 External Interfaces

5.4 Deployment Requirements