Software Requirements Specification

for

Relay

Version 1.0 approved

Prepared by Nagi Obeid, Kevin Mitra

Advanced Software Engineering, CSUB

02/23/2021

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Intended Audience and Reading Suggestions 1

1.3 Product Scope 1

2. Overall Description 1

2.1 Product Perspective 1

2.2 Product Functions 1

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

3. External Interface Requirements 2

3.1 User Interfaces 2

3.2 Hardware Interfaces 2

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 3

4.1 System Feature 1 3

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 4

5.3 Security Requirements 4

5.4 Software Quality Attributes 4

5.5 Business Rules 4

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 5

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Relay | 02/23/2021 | Initial Submission | 1.0 |
| Relay | 02/25/2021 | Technical Modifications | 1.0 |

# Introduction

## Purpose

The purpose of Relay is to establish a communication path between two users using sockets. The path will allow any two users using the service to send and receive messages using the web, similar to other existing web messaging applications.

## Intended Audience and Reading Suggestions

This document is intended for any Developer that will be working on the Relay platform. Marketing staff and project managers are also highly encouraged to read the SRS document to gain a better understanding of the scope of the application. The SRS document can also be used by general users to get familiar with the different features and functionality of the platform. It is highly recommended that if anyone is to read the document to start by understand the purpose 1.1, then move on to the Overall Description.

## Product Scope

The General function of the platform is to deliver a method of communication between two users. The first objective is to build a seamless UI that will engage and captivate the user. The second objective is to develop a quick and responsive connection that will relay the messages between users. It will be essential to attempt in making bugs and connection lags as minimal as possible.

# Overall Description

## Product Perspective

The Relay platform is a completely individual entity and has no relation to other messaging platforms or any existing software applications on the market.

## Product Functions

Users Should be able to create an account and register to the Relay database.

Users with an existing account should be able to login to the application.

Logged in users will have the ability to request a user to become a Buddy

Users will be able to remove a Buddy from their Buddy list.

Users will be able to send a message to a Buddy

If a user no longer wants to have a Relay account, the user will have the option to deactivate the account.

## User Classes and Characteristics

There will be a class of User, which will give each Relay account holder the ability to Send Buddy Request, Delete Messages, Remove Buddy, and Deactivate Account.

There will be a class of ChatSession which will hold the data for each user within the chat. The data will include each users’ ID and Username.

## Operating Environment

The application will be hardware agnostic since it will function on any device with a web browser and internet connection. However older devices (any device manufactured before 2013) may not render intended UI.

## Design and Implementation Constraints

There are very few hardware limitations for the Relay application, essentially any device with a Web Browser should work as expected.

The applications core business logic should be programmed using Python and JavaScript. While the UI will be requiring HTML, CSS and jQuery, let it be noted that the Bootstrap framework will be used to render many of the styling of the application, attempt to keep raw CSS use at a minimum. PHP will be used on the server-side to establish a connection to the Relay database and register users. MySQL will be the required database.

## User Documentation

Online tutorials must be delivered with the application in the form of a webpage.

# External Interface Requirements

## User Interfaces

The goal is to have each function of the application be viewed in its own page or container. This will remove as much visual noise as possible and keep the UI very clean and fluid.

**Landing Page**

Should display two buttons to the user, one to Login and the other to Sign up.

**Home Page**

Should display a section with various tabs listing the different functions to perform. The home page will also display a list of users with a Relay account, each Users div should have a button that if pressed sends a Buddy request to that user.

**Send Message**

Should display all the Users that are currently Buddies of the current user, give the user the option to select which Buddy to send a message to. A styled text-field should be displayed allow the user to key in a maximum of 250 characters. A single button should be displayed on the page titled as SEND

**Deactivate**

Should prompt the user if they are sure that they would like to deactivate their account. Two buttons will be displayed YES and NO, if YES is selected destroy the account from the database and redirect the user to the landing page. If NO is selected redirect the user to the Home page.

## Hardware Interfaces

The only hardware interface is the keyboard on a device.

## Software Interfaces

The software interface will be the connection of the user to the appropriate user account on the database.

## Communications Interfaces

Since we will be building a messaging application there will be one primary communication method. We will be using Python to establish sockets as the two end points for communication between two devices or users. The server-side script will establish a socket and bind it to an IP address and port specified by the user. The script will then stay open and receive connection requests. A web browser will be the only form of communication, there will not be a native application for specific operating systems or devices.

# System Features

Send and receive messages

## System Feature 1

Create an Account

4.1.1 Description and Priority

When users create an account, it stores into the database for them to access it.

4.1.2 Stimulus/Response Sequences

Users with their accounts created may send a message while the receiver receives the messages from the sender.

## System Feature 2

Account Login

4.2.1 Description and Priority

Users may login into their account that is stored inside the database for access.

4.2.2 Stimulus/Response Sequences

After logging in, users may continue to send and receive messages with their user account.

# Other Nonfunctional Requirements

## Performance Requirements

Using Python and PHP will have the system become reliable, availability, and maintainability when users connect to their accounts.

## Safety Requirements

1. The User may send messages.

2. The User may receive messages.

## Security Requirements

- If a user inputs the password incorrectly with more than three attempts, their user account will be locked.

- If a user account is locked, contact the administrator to unlock it.

## Software Quality Attributes

The attributes will include:

- Reliability for users being able to send and receive.

- Maintainability: for users to keep using the server.

- Testability: to make sure the software communication functions well.

## Business Rules

- The server and the client have their responsive roles for sending and receiving messages with user accounts.

- Each side must follow the guidelines, otherwise, their account will be terminated.

- Any tolerance or profanity may receive their first warning.

- Second warning will have the user accounts terminated.