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

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>