**Software Engineering**

**Software Requirements Specification**

**(SRS) Document**

**<Puzzle>**

**<9/19/2022>**

**<Version 1.0>**

**<By: Sofie Muska, Stuart Bridges, and Benjamin Woods>**

**<We have abided by the UNCG Academic Integrity Policy on this assignment.>**

|  |
| --- |
| **Table of Contents** |

[1. Introduction 2](#_Toc113453891)

[2. General Description 3](#_Toc113453892)

[3. Functional Requirements 3](#_Toc113453893)

[4. Technical Requirements 4](#_Toc113453894)

[4.1 Operating System & Compatibility 4](#_Toc113453895)

[4.2 Interface requirements 4](#_Toc113453896)

[4.2.1 User Interfaces 4](#_Toc113453897)

[4.2.2 Hardware Interfaces 4](#_Toc113453898)

[4.2.3 Communications Interfaces 4](#_Toc113453899)

[4.2.4 Software Interfaces 4](#_Toc113453900)

[5. Non-Functional Requirements 4](#_Toc113453901)

1. Introduction
   1. **Purpose:** The goal of the Puzzle application is to enable rock climbers to effortlessly keep track of their climbing progress and to help them find the motivation they need to achieve their rock climbing goals by connecting them to the rock-climbing community through climbing gyms.
   2. **Document Conventions:** The purpose of this Software Requirements Document (SRD) is to describe the climber view, the climbing gym view, and the administrator view for the Puzzle web application as well as the developer-oriented requirements. The climber-oriented requirements frame the web app from the climber’s perspective, and the same applies, respectively, to the climbing gym requirements, to the administrator requirements, and to the developer requirements. The requirements describe the different types of climber, climbing gym, and administrative scenarios served by the application, and they include a detailed description of functional, data, performance, and other important requirements.
   3. **Definitions, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| **Term** | **Definition. Acronym, Abbreviation** |
| SQL | A domain-specific language for managing relational databases. |
| API | An application programming interface is a software intermediary that allows two applications to talk to each other. |
| Java | The programming language we will use to build the backend of this web application. |
| JPA | A Java Persistence API is a specification of Java that is used to access, manage, and persist data between Java objects and a relational database. |
| JavaScript | JavaScript is a scripting or programming language that allows the implementation of complex features on web pages. |
| HTML | The web development language we will use to build the frontend of this web application. |
| CSS | Cascading Style Sheets is a stylesheet language used to describe the presentation of a document written in HTML. |
| Spring Boot | A Java-based framework used to help create the standalone web application. |
| React | A JavaScript library for building user interfaces. |

* 1. **Intended audience:** The Introduction, the General Description, and the Functional Requirements sections of this documentation are meant for all users to view, whereas the Technical and Non-Functional Requirements sections are meant for developers.

Project Stakeholders: developers, administrators, climbing gyms, and climbers.

* 1. **Project Scope:** Our goal is to make the tracking of climbing progress easier for climbers and to make it easier for climbing gyms and climbers to connect. This aligns with our overall business goals: to make everyday actions simpler and the path to connections between people clearer. This project will cement Caffeinated Foxes in the minds of climbers as individuals who seek to inspire them to become better versions of themselves through determination and consistent effort.
  2. **Technology Challenges:** TBD
  3. **References:** TBD

## General Description

* 1. **Product Perspective:** Puzzle found its origins in a climber’s desire for a simpler way to track their climbing progress. The idea was originated by a climber for climbers.
  2. **Product Features:** The product features include the ability for individual climbers and climbing gyms to create accounts and the ability for administrators to manipulate those accounts. Climbers can also add climbing routes to their profiles, where they can track their climbing progress, with different tracking options based on climbing style. For gyms, the functionality also includes the ability to create climbing routes, which they can share with their subscribers, and to update their subscribers on any events taking place at the gym. For administrators, the functionality also includes the possibility to delete and view accounts as well as to create climbing gym accounts.
  3. **User class and characteristics:** Our web application requires basic knowledge of how to interact with traditional web pages and a general understanding of climbing terminology.
  4. **Operating environment:** The application is designed to operate on the web across many different devices.
  5. **Constraints:** Due to our introductory knowledge of project development and limited resources, we had to develop with a small selection of web environments in mind, which may have an undesirable effect on the user interface.
  6. **Assumptions and dependencies:** We assume that only climbers and those with climbing knowledge will be using the application. We also assume that climbers will likely be using the web application on a mobile device in an indoor rock-climbing gym while climbing gyms and administrators will likely be accessing the web application via laptop or desktop. The reliability of Spring Boot may affect the project.

## Functional Requirements

* 1. **Primary**

FR0: The system will allow the user to create a “climber” account.

FR1: The system will allow the user to create a “gym” account.

FR2: The system will allow the user to log into their account.

FR3: The system will allow a climber account to add a climbing route to their profile. The route can either be added manually or from the route catalog of a gym the climber is subscribed to.

FR4: The system will allow the climber to track their climbing progress on a specific route. Progress tracking will be done either by number of falls or by time, depending on the climbing style associated with the route.

FR5: The system will store progress information in a database.

FR6: The system will display progress information to the climber both in table format and as a “number of falls over date” or a “time over date” graph.

FR7: The system will allow the climber to input their number of falls or climbing time into a table and to select the date on which they attempted the route in the table’s date column.

FR8: The system will allow the climber to subscribe to a climbing gym.

FR9: The system will allow the climber to unsubscribe from a climbing gym.

FR10: The system will maintain a list of the climber’s active gym subscriptions.

FR11: The system will allow the climbing gym to update their homepage with event information.

FR12: The system will allow the climbing gym to add climbing routes to their catalog.

FR13: The system will allow the climbing gym to remove climbing routes from their catalog.

FR14: The system will allow administrators to view a list of user accounts.

FR15: The system will allow administrators to delete user accounts.

* 1. **Secondary:** TBD

## Technical Requirements

* 1. Operating System & Compatibility

The application will be compatible with any operating system that is able to view and to interact with traditional web pages.

* 1. Interface requirements
     1. User Interfaces

The first screen will be the login/register home screen, where there will be two buttons—for the user to either register or to log in—and a short description of the web app.

If the user hits the “Register” button, they will be taken to a page where they will be asked to either “Register as a Gym” or to “Register as a Climber.” If they choose to “Register as a Gym,” they will be taken to a page where they will fill out their gym information, which will create their account. If they choose to “Register as a Climber,” they will be taken to a page where they will fill out their personal information—name, surname, email address, and password. The email address will function as their username. Once they fill out their information, they will click on the “Create Account” button to create their account.

If the user hits the “Log in” button, they will be taken to a screen where they will see a box with two fields to fill out, “Username” and “Password.” Once they fill out the fields, they will hit the “Log in” button under the “Password” field, which will take them to their homepage. The layout of their homepage will depend on the type of user they are.

The “administrator” homepage will include a list of user accounts. Next to each account, there will be a trash can symbol. Clicking this symbol will allow the administrator to delete the specific user account.

The “climbing gym” homepage will display a history of the gym’s announcements to its subscribers. It will also include an “Events and Updates” field, where the gym can inform climbers of upcoming events and other happenings. Clicking the “Post” button will post the message. The “climbing gym” homepage will also include a “Climbing Routes” tab. Here, the climbing gym will see a list of its current routes. Beneath each route name will be relevant route information, and next to each route name will be a trash can symbol and a pen symbol. Clicking the trash can symbol will allow the gym to delete the route and clicking the pen symbol will allow the gym to edit the route. Above the list of routes, there will be a “Create New Route” button, which will allow the climbing gym to add a new route to their catalog. The gym will be able to name the route and to add relevant information below. When it is done creating the route, the gym will hit the “Post” button to post the new route.

The “climber” homepage will display a history of the climber’s attempted routes. Clicking on the name of the route will take the climber to a new page, where they will see the name of the route. Beneath the name of the route will be the route information. Finally, there will be a table. This table will have two columns: “Date” and either “Number of Falls” or “Time.” Above the table will be a button with the option to “Add New Attempt.” Clicking this button will add a new row to the table, where the climber will be able to input the relevant information. Next to the table, there will be a graph, which will automatically change as the climber updates their progress. If the first entry in the table has 0 listed under “Number of Falls,” the user will be notified that they “flashed” the route. In any other case, once the climber inputs 0 under “Number of Falls,” they will be notified that they “cleaned” the route. These messages will be displayed above the graph.

The “climber” homepage will also include another tab, called “Climbing Gyms.” Here, the climber will see a list of their active gym subscriptions. Next to each gym name will be a trash can symbol, which will allow the user to unsubscribe from the gym. Clicking on the gym name will take them to the gym’s homepage. On the gym’s homepage, they will see the gym’s messages to the public and a tab called “Climbing Routes.” Clicking on this tab will take them to the gym’s route catalog. There, clicking a route will expand the route to show the route’s information. Next to the route name, there will be a plus button, which will allow the climber to add the route to their progress monitor if they have climbed it. The button will not function if the climber has already added the route to their progress monitor.

On the climber’s “Climbing Gym” page, there will also be a search box, which will allow them to search for climbing gyms to subscribe to. Hitting enter in the search box will show a dialogue box, which will ask them: “Would you like to subscribe to [gym name]?” The options below this question will be “Yes” and “No.” Hitting “Yes” will take the climber to the gym’s homepage.

The style guides used will be IDE dictated, specifically the Intellij and Netbeans style guides.

* + 1. Hardware Interfaces

The web application will run on any hardware device that has access to the internet, the ability to display webpages, and the ability to interact with web pages. This includes, but is not limited to, smartphones, tablets, desktop computers, and laptops.

* + 1. Communications Interfaces

We will use necessary internet protocols built into Spring Boot to connect users to our web applications.

* + 1. Software Interfaces

We will use React and Spring Boot to help build the frontend, as well as JPA for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend.

## Non-Functional Requirements

* 1. **Performance requirements**

NFR0(R): The user will be able to create a “climber” account in under 3 minutes.

NFR1(R): The user will be able to create a “gym” account in under 3 minutes.

NFR2(R): The user will be able to log into their account in under 1 minute.

NFR3(R): A “climber” user will be able to add a route to their profile in under 3 minutes.

NFR4(R): A “climber” user will be able to view their climbing progress in under 1 minute.

NFR5(R): The local copy of the climbing progress database will consume less than 20 MB of memory.

NFR6(R): The table and graph will consume less than 20 MB of memory.

NFR7(R): A “climber” user will be able to input their progress data into the route progress table in under 1 minute.

NFR8(R): A “climber” user will be able to subscribe to a climbing gym in under 2 minutes.

NFR9(R): A “climber” user will be able to unsubscribe from a climbing gym in under 2 minutes.

NFR10(R): The local copy of a climber’s list of gym subscriptions will consume less than 20 MB of memory.

NFR11(R): A “climbing gym” user will be able to update its subscribers on events and other news in under 5 minutes.

NFR12(R): A “climbing gym” user will be able to add a new route to their route catalog in under 5 minutes.

NFR13(R): A “climbing gym” user will be able to remove a route from their route catalog in under 1 minute.

NFR14(R): The local copy of the list of user accounts will take up less than 50 MB of memory.

NFR15(R): An administrator will be able to delete a user account in under 1 minute.

* 1. **Safety requirements**

Do not allow any user to create an administrator account, as they could perform unauthorized deletions of accounts.

* 1. **Security requirements**

NFR16(R): The system will only be usable by users that have an account.

NFR17(R): All passwords will be hashed for greater security.

* 1. **Software quality attributes**
     1. Availability

Should be available to any web users with access to a web browser.

* + 1. Correctness

The code should adhere to our specified style guides and pass the number of unit and system tests we create.

* + 1. Maintainability

The code and system should be written and developed with developers in mind, who may or may not be a part of the original team. The functions and web app behavior should leave clear space for updates.

* + 1. Reusability

Users should have all their data stored when they log in and should not have to worry about any of that data being lost. The code should also be readable to developers who have never seen it.

* + 1. Portability

The project being built on the web should make it extremely portable and accessible to nearly anyone who wants to use it. Using Java for the backend should also increase the portability of the software.

Detailing on the additional qualities that need to be incorporated within the software:

1. Dependency Inversion (part of the SOLID principles)
2. Adherence to the Style Guide
3. Proper use of Enumeration
4. MVC Architecture
5. Separation of Concerns
6. Loose Coupling
   1. **Process Requirements**
      1. Development Process Used: The project is built on the incremental development model with specification, development, and validation activities interleaved (with feedback).
      2. Time Constraints: The overall time constraint is the UNCG Fall Semester 2022, and implementation/development of the final project is constrained to November 22, 2022.
      3. Cost and Delivery Date: There is no associated cost with development of Puzzle.
   2. **Other requirements**

NFR18(R): UNCG Honor Code adherence.

All SRS/SRD should be:

* **Correct:** A method of analysis that ensures that the software meets the requirements identified.
* **Unambiguous:** There is only one interpretation of what the software will be used for and it is communicated in a common language.
* **Complete:** There is a representation for all requirements for functionality, performance, design constraints, attributes, or external interfaces.
* **Consistent:** Must be in agreement with other documentation, including a systems requirements specification and other documents.
* **Ranked for Importance and/or Stability:** Since all requirements are not of equal weight, you should employ a method to appropriately rank requirements.
* **Verifiable:** Use measurable elements and defined terminology to avoid ambiguity.
* **Modifiable:** A well-defined organizational structure of the SRS document that avoids redundancies can allow easy adaptation.
* **Traceable:** Ability to trace back to the origin of development and move forward to the documents produced from the SRS.
* **Legible and Professionally Presented**: Must use a consistent font and style. Must have proper formatting of tables and charts. Must be grammatically correct. Use active tense and concise sentences.

Diagram

Description automatically generated