***Software Engineering***

***Software Requirements Specification***

***(SRS) Document***

**Gaming Library API**

**02/20/24**

**Version 0.1**

**By: Jacob Echeverry Espinosa, Elijah Carpenter, Chris Nieves**

**We have abided by the academic integrity policy on this assignment.**

**Table of Contents**

[1. Introduction 3](#_Toc156255120)

[1.1. Purpose 3](#_Toc156255121)

[1.2. Document Conventions 3](#_Toc156255122)

[1.3. Definitions, Acronyms, and Abbreviations 3](#_Toc156255123)

[1.4. Intended Audience 4](#_Toc156255124)

[1.5. Project Scope 4](#_Toc156255125)

[1.6. Technology Challenges 4](#_Toc156255126)

[1.7. References 4](#_Toc156255127)

[2. General Description 5](#_Toc156255128)

[2.1. Product Features 5](#_Toc156255129)

[2.2. User Class and Characteristics 5](#_Toc156255130)

[2.3. Operating Environment 5](#_Toc156255131)

[2.4. Constraints 5](#_Toc156255132)

[2.5. Assumptions and Dependencies 5](#_Toc156255133)

[3. Functional Requirements 6](#_Toc156255134)

[3.1. Primary 6](#_Toc156255135)

[3.2. Secondary 6](#_Toc156255136)

[3.3. Use-Case Model 6](#_Toc156255137)

[3.3.1. Use-Case Model Diagram 6](#_Toc156255138)

[3.3.2. Use-Case Model Descriptions 7](#_Toc156255139)

[3.3.2.1. Actor: Manager (Alice) 7](#_Toc156255140)

[3.3.2.2. Actor: Actor Name (Responsible Team Member) 7](#_Toc156255141)

[3.3.2.3. Actor: Actor Name (Responsible Team Member) 7](#_Toc156255142)

[3.3.3. Use-Case Model Scenarios 7](#_Toc156255143)

[3.3.3.1. Actor: Manager (Alice) 7](#_Toc156255144)

[3.3.3.2. Actor: Actor Name (Responsible Team Member) 7](#_Toc156255145)

[3.3.3.3. Actor: Actor Name (Responsible Team Member) 8](#_Toc156255146)

[4. Technical Requirements 9](#_Toc156255147)

[8.1. Interface Requirements 9](#_Toc156255152)

[8.1.1. User Interfaces 9](#_Toc156255153)

[8.1.2. Hardware Interfaces 9](#_Toc156255154)

[8.1.3. Communications Interfaces 9](#_Toc156255155)

[8.1.4. Software Interfaces 9](#_Toc156255156)

[9. Non-Functional Requirements 10](#_Toc156255157)

[9.1. Performance Requirements 10](#_Toc156255158)

[9.2. Safety Requirements 10](#_Toc156255159)

[9.3. Security Requirements 10](#_Toc156255160)

[9.4. Software Quality Attributes 10](#_Toc156255161)

[9.4.1. Availability 10](#_Toc156255162)

[9.4.2. Correctness 10](#_Toc156255163)

[9.4.3. Maintainability 10](#_Toc156255164)

[9.4.4. Reusability 10](#_Toc156255165)

[9.4.5. Portability 10](#_Toc156255166)

[9.5. Process Requirements 10](#_Toc156255167)

[9.5.1. Development Process Used 10](#_Toc156255168)

[9.5.2. Time Constraints 10](#_Toc156255169)

[9.5.3. Cost and Delivery Date 10](#_Toc156255170)

[9.6. Other Requirements 10](#_Toc156255171)

[10. Design Documents 11](#_Toc156255172)

[10.1. Software Architecture 11](#_Toc156255173)

[10.2. High-Level Database Schema 11](#_Toc156255174)

[10.3. Software Design 11](#_Toc156255175)

[10.3.1. State Machine Diagram: Actor Name (Responsible Team Member) 11](#_Toc156255176)

[10.3.2. State Machine Diagram: Actor Name (Responsible Team Member) 11](#_Toc156255177)

[10.3.3. State Machine Diagram: Actor Name (Responsible Team Member) 11](#_Toc156255178)

[10.4. UML Class Diagram 11](#_Toc156255179)

[11. Scenario 11](#_Toc156255180)

[11.1. Brief Written Scenario with Screenshots 11](#_Toc156255181)

# Introduction

## Purpose

The goal of the Gaming Library API is to connect gamers with new interesting titles and empower them to

give their valuable feedback on the games they play, both to game developers and to other gamers.

## Document Conventions

The purpose of this Software Requirements Document (SRD) is to outline the Game Library API (GLA) both from a developer-side perspective and user-side perspective.

From the developer-side, this document will outline the overall software architecture, features to be implemented, technologies to be used, potential hazards to be avoided, and classes of client to be served.

From the user-side, this document will outline the high-level functionalities of the software to different user classes.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Java | A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the Restaurant Manager. |
| MySQL | Open-source relational database management system. |
| .HTML | Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content. |
| SpringBoot | An open-source Java-based framework used to create a micro Service. This will be used to create and run our application. |
| MVC | Model-View-Controller. This is the architectural pattern that will be used to implement our system. |
| Spring Web | Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system. |
| Thymeleaf | A modern server-side Java template engine for our web environment. This is one of the dependencies of our system. |
| NetBeans | An integrated development environment (IDE) for Java. This is where our system will be created. |

## Intended Audience

Developers:

* Jacob Echeverry Espinosa
* Eijah Carpenter
* Chris Nieves

Clients:

* Users
* Developers
* Administrators

The Introduction section is intended for both developers and clients.

The General Description section is intended for both developers and clients.

The remainder of the document describes technical specifics, and therefore is intended for developers.

## Project Scope

The goal of the software is to provide a dynamic, visually appealing web platform which recommends high-quality games to users and empowers users to give honest feedback on the games they play.

This will benefit both users and game developers. Users will receive high quality recommendations according to their specific preferences. Developers will receive vast datasets of user feedback on their games, which will allow them to reinforce features which are popular and rework features which are not.

The ultimate, aggregate effect will be users playing better games and developers making better games.

## Technology Challenges

To be completed…

## References

To be completed…

# General Description

## Product Features

As a web application, the primary capacity of the software will be a website

Users will be able to access the database of listed games, and their reviews, without registering an account.

Upon registering an account, the user will input their category preferences for recommendations, which are mutable at any time, and the user will gain the functionality of leaving and removing reviews for games.

Categories in this sense refers to a variety of qualities games can have, including but not limited to genre,

playstyle, and art-style.

The recommendation system will respond both to a user’s preferences and to the given review scores.

Games will be added by interactions from publishers. Upon registering a publisher account, the functionality of adding/removing games from the database is gained,

## User Class and Characteristics

The application is designed around three user classes:

* Users:

Gamers visiting the website, who are recommended games and leave reviews.

* Publishers

Game developers, who register their games on the website.

* Administrators

The professionals controlling the website and managing its functionality.

## Operating Environment

The application is designed to operate on the web and to be compatible with all consumer devices commonly used for web-browsing, including desktops, laptops, smartphones, and tablets.

## Constraints

To be completed…

## Assumptions and Dependencies

The software will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within NetBeans.

# Functional Requirements

## Primary

The system will allow users to search for games and leave reviews of game.

The system will allow users to record their preferences for games.

The system will recommend games to users based on the aggregate reviews and their personal preferences.

The system will allow publishers to add/remove their games, and leave/modify descriptions of them.

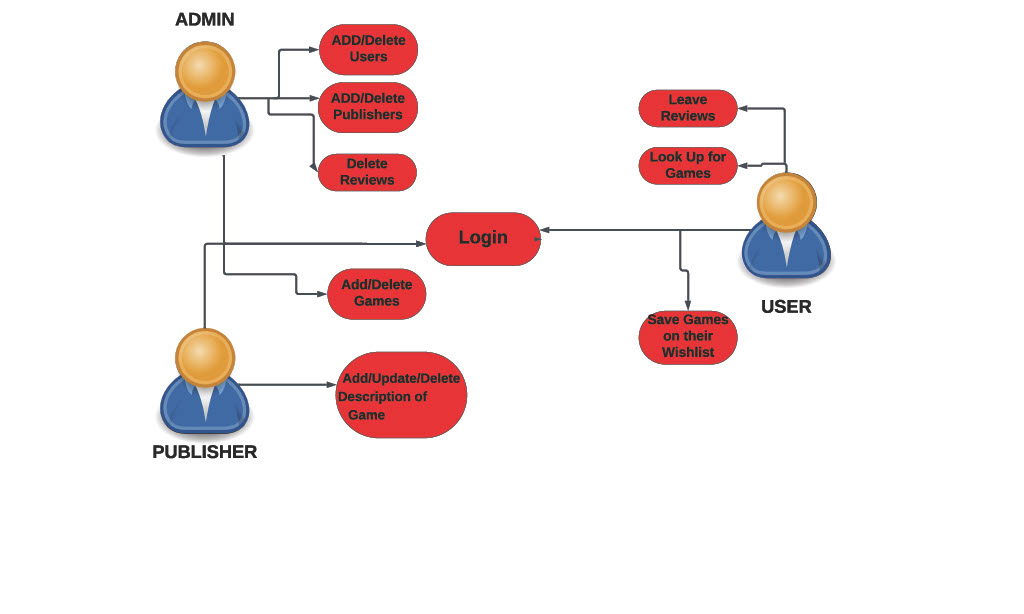
The system will allow administrators total control over the applications functionalities, including the deleting publishers and users, and deleting/modifying reviews.

## Secondary

* All client classes will have password protected accounts.
* Each class will have access to its functionalities only.
* Additionally, a user will only be ale to modify their own reviews, and a publisher will only be able to modify their own game descriptions.

## Use-Case Model

## Use-Case Model Diagram



## Use-Case Model Descriptions

### Actor: Publisher (Jacob Echeverry Espinosa)

* **Register games:** Upon registering a publisher account, a publisher will be able to register their games in the game database.
* **Change game description:** Given a game already registered and described, the description is mutable. This will be useful for games receiving updates or downloadable content.

### Actor: User (Elijah Carpenter)

* **Search for Games:** The user inputs preferences and receives recommendations for game titles. Additionally, they can freely search the database for titles based on name and qualities.
* **Leave Reviews:** For each individual game title, a user can leave a numerical ranking on a 1 to 10 scale, as well as a text description of their experience with the game.

### 3.3.2.3 Actor: Administrator (Chris Nieves)

* **Add/Remove Users/Publishers:** Adding manually may be necessary to aid with registration problems on the client side, and removing may be necessary to punish bad actions by clients.
* **Delete Reviews:** Likewise, deleting reviews is a basic moderation function that must be performed to control bad actors.

## Use-Case Model Scenarios

### Publisher: (Jacob Echeverry Espinosa)

* **Use-Case Name**: Register games
  + **Initial Assumption**: The publisher has access to the web app and has publisher status.
  + **Normal**: The user will enter a detailed description of their game, along with sample gameplay footage, concept art, and will categorize their game
  + **What Can Go Wrong:** Bad information could be erroneously inputted. The description is mutable for this reason.
  + **Other Activities**: Visual attachments can also be added/mutated, as with the text description.
  + **System State on Completion**: The game is added to the database and is visible to users.
* **Use-Case Name**: Change game description
  + **Initial Assumption**: The publisher has access to the web app and has publisher status.
  + **Normal**: The publisher alters the existing text description with new text.
  + **What Can Go Wrong**: n/a
  + **Other Activities**: Visual attachments can also be added/mutated, as with the text description.
  + **System State on Completion**: The changes made go to the database and are visible to users.

### User: (Elijah Carpenter)

* **Use-Case Name**: Search for Games
  + **Initial Assumption**: The user has a registered account with user status and has inputted game preferences.
  + **Normal**: Based on the user’s preferences and the numerical review rankings, the web app displays recommended games.
  + **What Can Go Wrong**: n/a
  + **Other Activities**: The user may also freely search through the database by titles, categories, and ranking.
  + **System State on Completion**: A selection of games is displayed to the user.
* **Use-Case Name**: Leave Reviews
  + **Initial Assumption**: The user has a registered account with user status.
  + **Normal**: The user writes and submits a text review and numerical ranking of a game.
  + **What Can Go Wrong**: Malicious ‘review-bombing’ is possible.
  + **Other Activities**: A user may also modify reviews they’ve already made.
  + **System State on Completion**: The review is stored in the database and visible to all users as attached to the game.

### Administrator: (Chris Nieves)

* **Use-Case Name**: Add/remove publishers/users
  + **Initial Assumption**: The administrator has a registered account with administrator privileges.
  + **Normal**: The administrator find a user to perform the action on, and selects the action in the user interface.
  + **What Can Go Wrong**: n/a
  + **Other Activities**: The administrator can also temporarily restrict functionalities from an account.
  + **System State on Completion**: The selection action is taken, and an account is added or removed from the database, along with its associated data.
* **Use-Case Name**: Remove reviews
  + **Initial Assumption**: The administrator has a registered account with administrator privileges.
  + **Normal**: A review is selected to be deleted, and a final deletion interface is confirmed.
  + **What Can Go Wrong**: Its possible an administrator could maliciously remove negative reviews from having a conflict of interest.
  + **Other Activities**: Administrators also have the ability to moderate reviews, for example, to remove foul language.
  + **System State on Completion**: The review is removed from the database and is no longer visible.

# Technical Requirements

## Interface Requirements

### User Interfaces

[The logic behind the interactions between the users and the software. This includes the sample screen layout, buttons and functions that would appear on every screen, messages to be displayed on each screen and the style guides to be used.]

### Hardware Interfaces

The web application will support all consumer products commonly used for web-browsing, including laptops, desktops, smartphones and tablets.

### Communications Interfaces

It must be able to connect to the internet as well as the local database on phpMyAdmin.

### Software Interfaces

We will use React and Spring Boot ThymeLeaf 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

[Constraints on the services or functions offered by the system (e.g., timing constraints, constraints on the development process, standards, etc.). Often apply to the system as a whole rather than individual features or services.]

## Performance Requirements

[The performance requirements need to be specified for all the functional requirements.]

## Safety Requirements

[List out any safeguards that need to be incorporated as a measure against any possible harm the use of the software application may cause.]

## Security Requirements

[Privacy and data protection regulations that need to be adhered to while designing of the product.]

## Software Quality Attributes

[Detailing on the additional qualities that need to be incorporated within the software like maintainability, adaptability, flexibility, usability, reliability, portability etc.]

### Availability

[Details]

### Correctness

[Details]

### Maintainability

[Details]

### Reusability

[Details]

### Portability

[Details]

## Process Requirements

### Development Process Used

[Software Process Model]

### Time Constraints

### Cost and Delivery Date

## Other Requirements

# Design Documents

## Software Architecture

## High-Level Database Schema

## Software Design

### State Machine Diagram: Actor Name (Responsible Team Member)

### State Machine Diagram: Actor Name (Responsible Team Member)

### State Machine Diagram: Actor Name (Responsible Team Member)

## UML Class Diagram

# Scenario

## Brief Written Scenario with Screenshots