

***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
1.1. Purpose	3
1.2. Document Conventions	3
1.3. Definitions, Acronyms, and Abbreviations	3
1.4. Intended Audience	3
1.5. Project Scope	4
1.6. Technology Challenges	4
1.7. References	4
2. General Description	4
2.1. Product Features	4
2.2. User Class and Characteristics	5
2.3. Operating Environment	5
2.4. Constraints	5
2.5. Assumptions and Dependencies	5
3. Functional Requirements	6
3.1. Primary	6
3.2. Secondary	6
3.3. Use-Case Model	6
3.3.1. Use-Case Model Diagram	6
3.3.2. Use-Case Model Descriptions	6
3.3.2.1. Actor: Manager (Alice)	6
3.3.2.2. Actor: Actor Name (Responsible Team Member)	7
3.3.2.3. Actor: Actor Name (Responsible Team Member)	7
3.3.3. Use-Case Model Scenarios	7
3.3.3.1. Actor: Manager (Alice)	7
3.3.3.2. Actor: Actor Name (Responsible Team Member)	8
3.3.3.3. Actor: Actor Name (Responsible Team Member)	8
4. Technical Requirements	9
8.1. Interface Requirements	9
8.1.1. User Interfaces	9
8.1.2. Hardware Interfaces	9
8.1.3. Communications Interfaces	9
8.1.4. Software Interfaces	9
9. Non-Functional Requirements	10

9.1.	Performance Requirements	10
9.2.	Safety Requirements	10
9.3.	Security Requirements	10
9.4.	Software Quality Attributes	10
9.4.1.	Availability	10
9.4.2.	Correctness	10
9.4.3.	Maintainability	10
9.4.4.	Reusability	10
9.4.5.	Portability	10
9.5.	Process Requirements	10
9.5.1.	Development Process Used	10
9.5.2.	Time Constraints	10
9.5.3.	Cost and Delivery Date	10
9.6.	Other Requirements	10
10.	Design Documents	10
10.1.	Software Architecture	10
10.2.	High-Level Database Schema	11
10.3.	Software Design	11
10.3.1.	State Machine Diagram: Actor Name (Responsible Team Member)	11
10.3.2.	State Machine Diagram: Actor Name (Responsible Team Member)	13
10.3.3.	State Machine Diagram: Actor Name (Responsible Team Member)	13
10.4.	UML Class Diagram	14
11.	Scenario	15
11.1.	Brief Written Scenario with Screenshots	15

1. Introduction

1.1. 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.

1.2. 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.

1.3. 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.

1.4. 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.

1.5. 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.

1.6. Technology Challenges

The app is to be ran using MacOS or Windows OS using XAMPP or AMPPS database on phpadmin

1.7. References

Tom Lodge, Andy Crabtree, and Anthony Brown. 2018. IoT App Development: Supporting Data Protection by Design and Default. In Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers (UbiComp '18). Association for Computing Machinery, New York, NY, USA, 901-910. <https://doi-org.libproxy.uncg.edu/10.1145/3267305.3274151>

Erik Wilde and Mike Amundsen. 2019. The Challenge of API Management: API Strategies for Decentralized API Landscapes. In Companion Proceedings of The 2019 World Wide Web Conference (WWW '19). Association for Computing Machinery, New York, NY, USA, 1327-1328. <https://doi-org.libproxy.uncg.edu/10.1145/3308560.3320089>

2. General Description

2.1. 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,

2.2. 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.

2.3. 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.

2.4. Constraints

The use of an API that will provide all the necessary information of games may not provide actual accurate information if it may not be up to date.

2.5. 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.

3. Functional Requirements

3.1. 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.

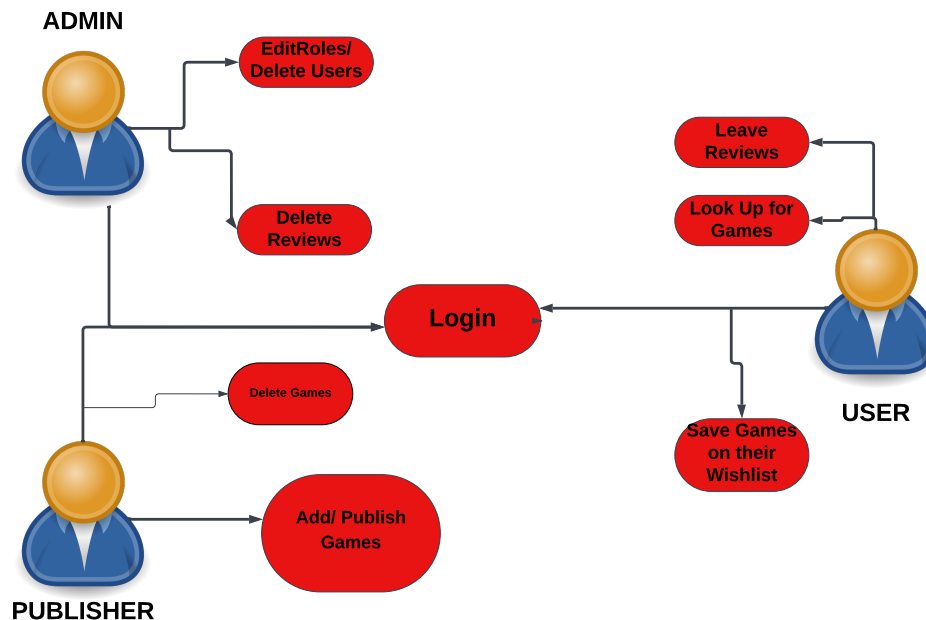
The system will allow administrators total control over the applications functionalities, including the deleting publishers and users, and deleting/modifying reviews.

3.2. Secondary

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

3.3. Use-Case Model

3.3.1. Use-Case Model Diagram



3.3.2. Use-Case Model Descriptions

3.3.2.1. 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 or Delete game or game description:** Given a game already registered and described, the description is mutable. This will be useful for games receiving updates or downloadable content.

3.3.2.2. 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.
- **Wishlist:** The user will be able to save the games. A database Wishlist which would then recommend other games based on it

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.

3.3.3. Use-Case Model Scenarios

3.3.3.1. 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 or Delete game or 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:** The publisher deletes a wrong gamer
 - **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.

3.3.3.2. 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.

3.3.3.3. 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.

4. Technical Requirements

4.1. Interface Requirements

4.1.1. User Interfaces

There will be a navbar on the top where the user can go through the home page, publisher page, admin page, games lists, and Wishlist concerning their role authority. By clicking on the admin page or logging in on the admin log, the user will be redirected to the admin section where he has the privileges to remove games by ID, remove users by ID, and edit users' roles. If the users go to the publisher section, the publisher will be able to publish games and edit games according to preference. By clicking the Wishlist or Browse games the User can look up games, leave reviews, and add them to their Wishlist which they can later check out.

4.1.2. Hardware Interfaces

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

4.1.3. Communications Interfaces

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

4.1.4. Software Interfaces

We will use React and Spring Boot Thyme Leaf 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 front end to the backend.

5. Non-Functional Requirements

5.1. Performance Requirements

- The redirecting from all pages should not take more than 2-3 seconds.
- Recommended games should change based on the user's Wishlist
- Deleting games should be able to be completed using an ID
- RAM space shouldn't go above 200MB

5.2. Safety Requirements

5. Prevent users from adding the same game twice in the Wishlist. Cannot have games that has been removed by the admin on the Wishlist
6. Publisher cannot publish the same game twice
7. Admin cannot demote another admin

5.3. Security Requirements

8. Spring boot security dependency
9. Passwords in database

5.4. Software Quality Attributes

5.4.1. Availability

10. As long as the API/Database and application is running, the app will be available

5.4.2. Correctness

11. Depends on the API information Accuracy. Methods in Java should help the API accuracy show.

5.4.3. Maintainability

12. CRUD classes will make the code straightforward to maintain.

5.4.4. Reusability

13. Good implementation code using CRUD can help make the code reusable.

5.4.5. Portability

14. Will be pushed to GitHub as a repository and can be accessed anywhere as long there is internet and you are able to download the necessary local files

5.5. Process Requirements

5.5.1. Development Process Used

15. We developed a process using html and Java using the MVC model to make sure our website works fast and efficiently.

5.5.2. Time Constraints

16. Finished by the end of April

5.5.3. Cost and Delivery Date

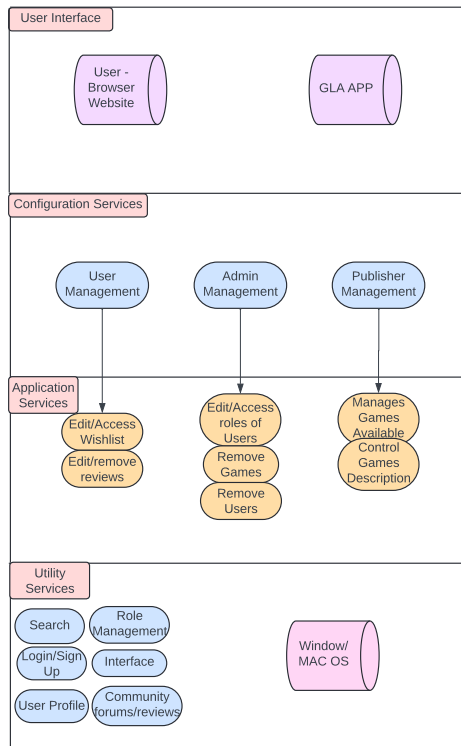
17. So far the application is free and the cost of our API is also presumed to be free

5.6. Other Requirements

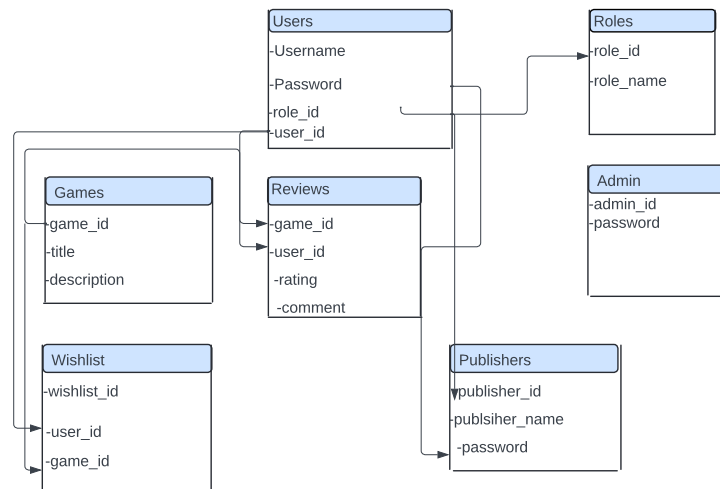
-N/A

6. Design Documents

6.1. Software Architecture

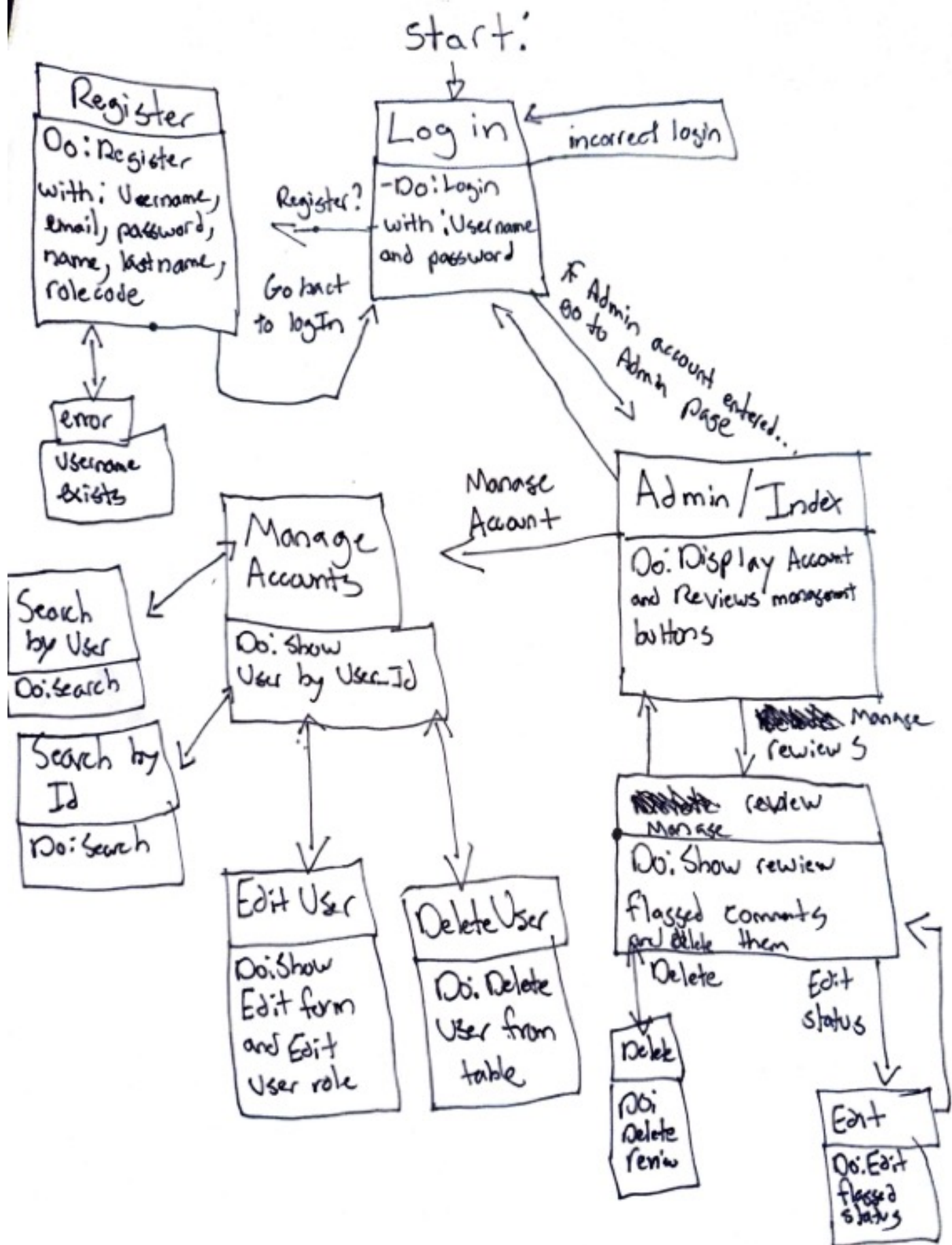


6.2. High-Level Database Schema

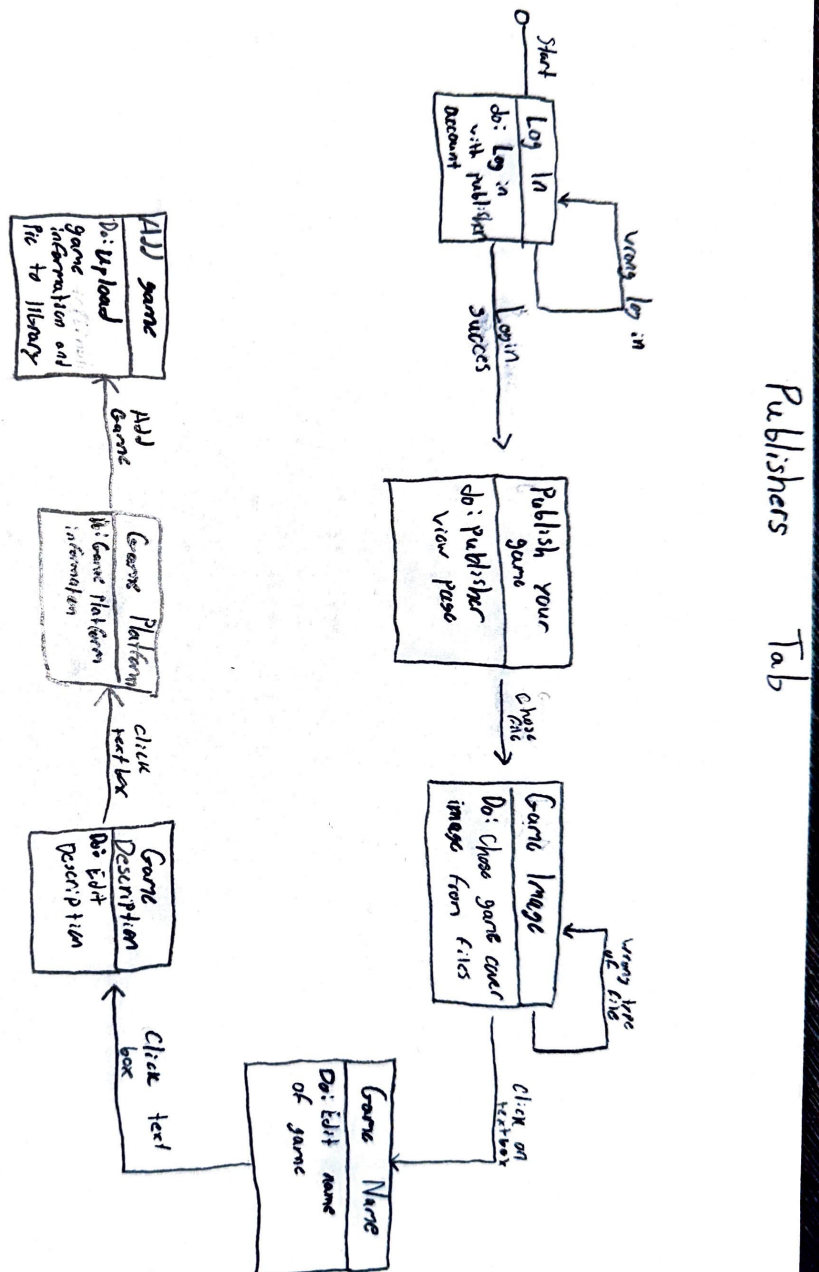


6.3. Software Design

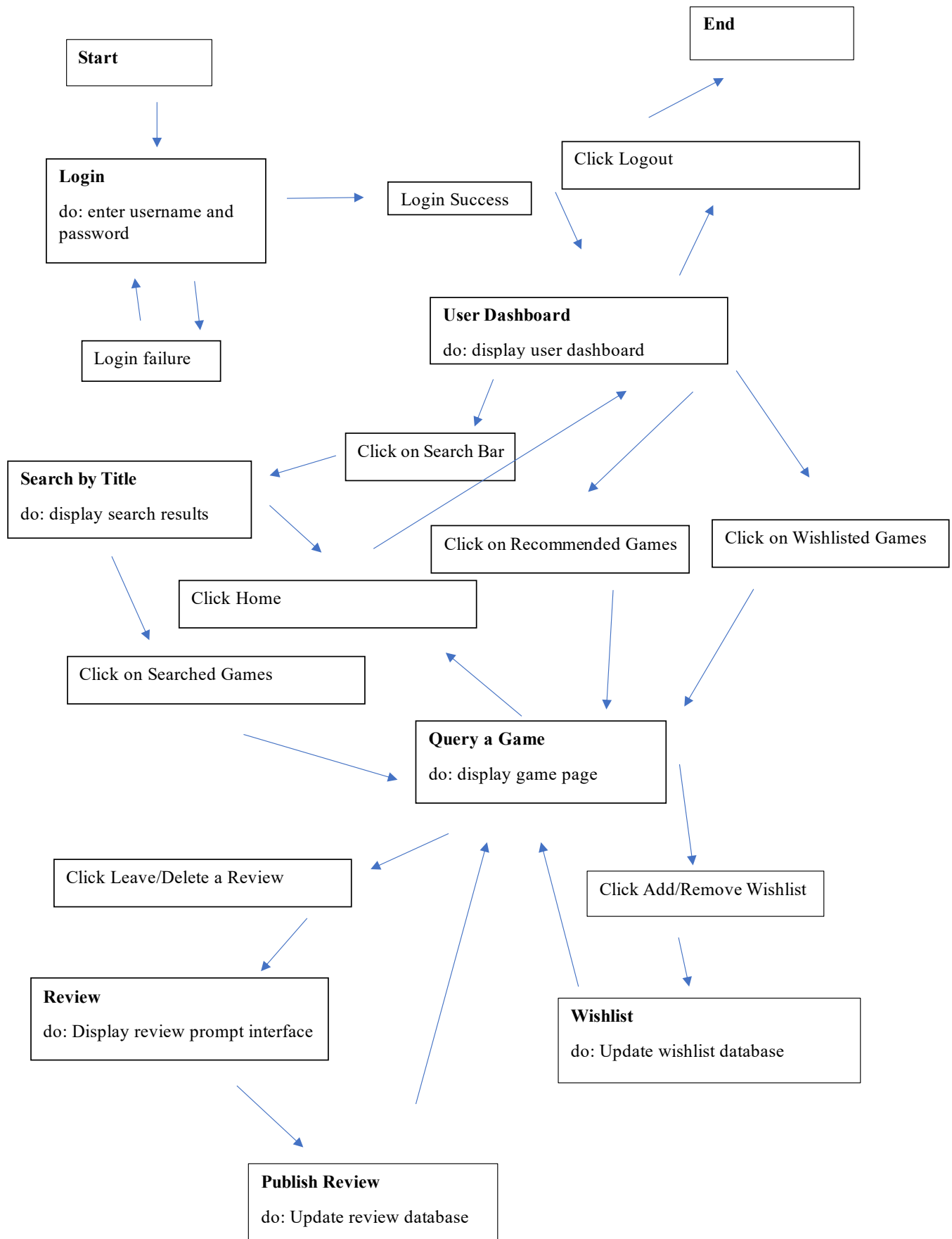
6.3.1. State Machine Diagram: Admin (Chris Nieves)



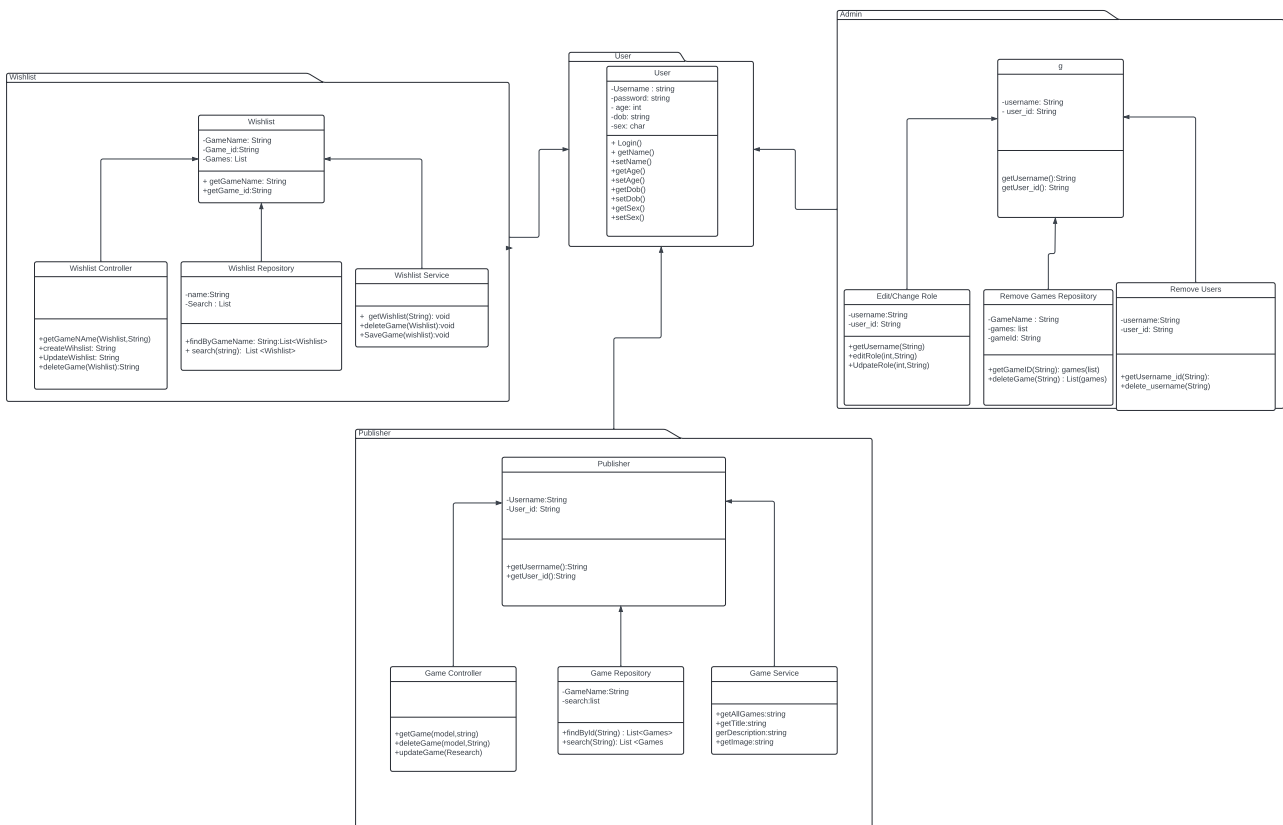
6.3.2. State Machine Diagram: Publisher (Jacob Echevarria)



6.3.3. State Machine Diagram: User (Elijah Carpenter)



6.4. UML Class Diagram

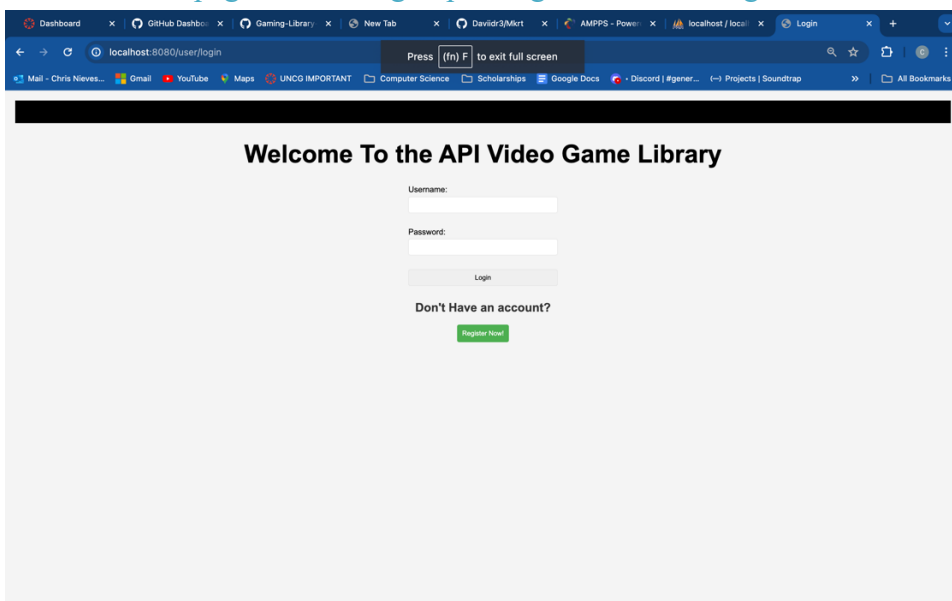


7. Scenario

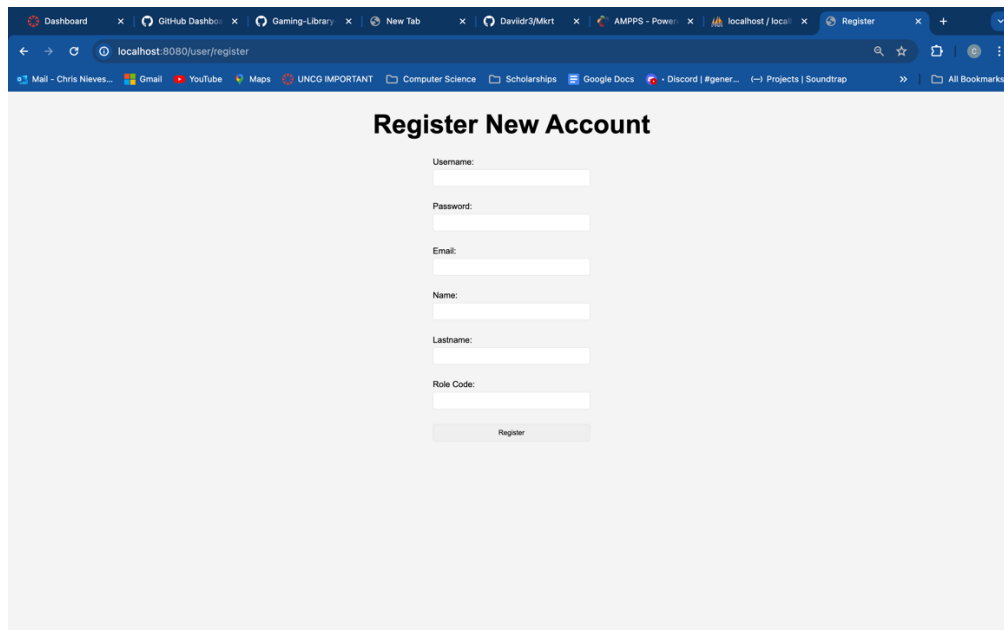
7.1. Brief Written Scenario with Screenshots

Written Scenario (Admin/ Login):

Users load the page and can sign up or log in to an existing account from the database.\

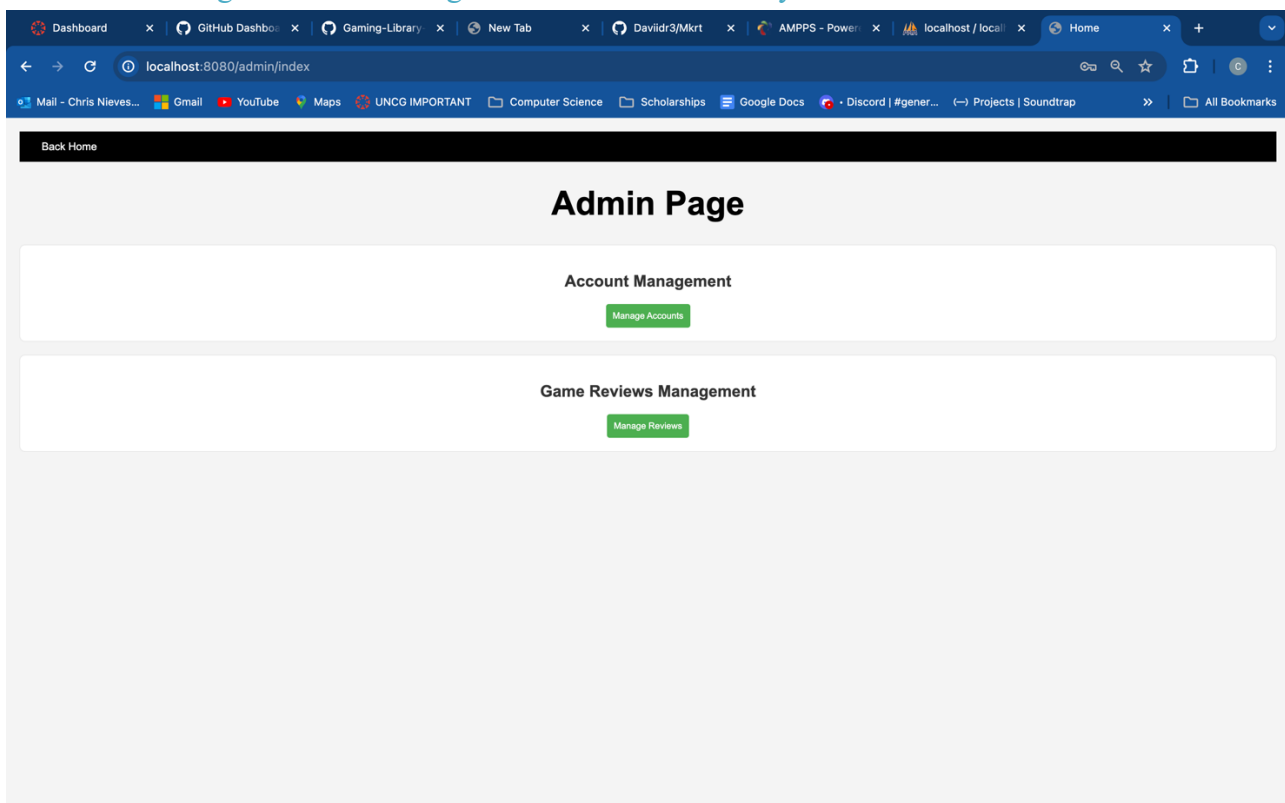


-When registering, the user must write in all fields except the last one. The code field determines whether or not the user can be a publisher or admin based on the code they put. The user cannot place a Username that already exists.



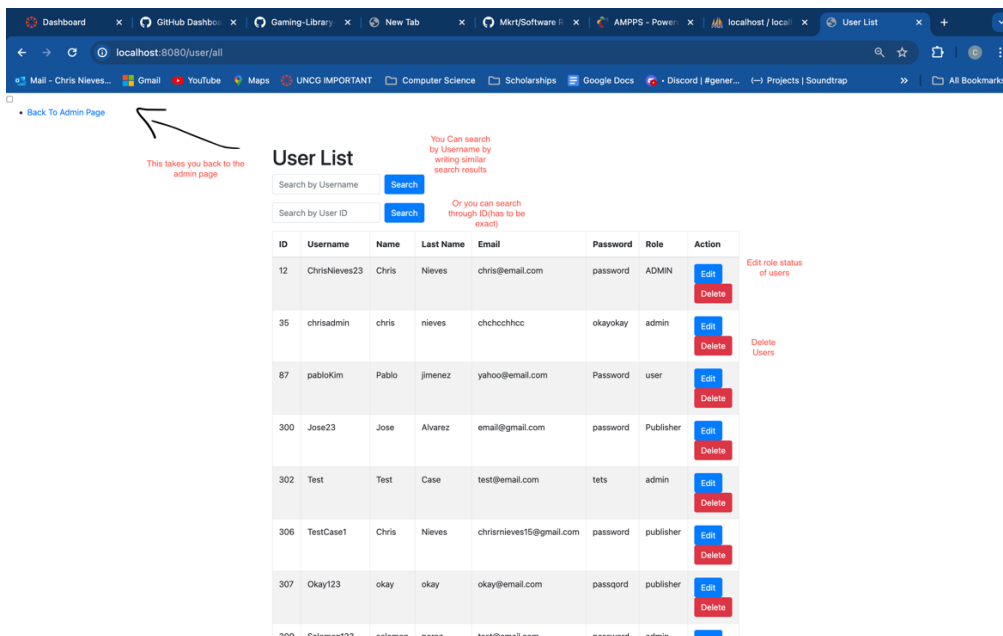
The screenshot shows a web browser window with the URL `localhost:8080/user/register`. The page has a title "Register New Account". Below the title, there are several input fields: "Username:", "Password:", "Email:", "Name:", "Lastname:", and "Role Code:". Each field has a corresponding text input box. At the bottom of the form, there is a "Register" button.

- On the admin page, the user has the ability to go through two different pages. (The account management and Manage Reviews. Which takes you to two different tables from a database



The screenshot shows a web browser window with the URL `localhost:8080/admin/index`. The page has a title "Admin Page". Below the title, there are two main sections: "Account Management" and "Game Reviews Management". Each section has a corresponding "Manage" button: "Manage Accounts" and "Manage Reviews".

-Once the Account Management button is clicked, the user will be redirected to the Account management pages where the user can edit or delete User



-If a user is searched, for example, the word “Chris” is inputted in the search bar. Then the corresponding users with Chris on it will pop up.

User List

ID	Username	Name	Last Name	Email	Password	Role	Action
12	ChrisNieves23	Chris	Nieves	chris@email.com	password	ADMIN	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
35	chrisadmin	chris	nieves	chchcchcc	okayokay	admin	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

-If the user searches by the number “12”. Then the corresponding user will pop up.

User List

ID	Username	Name	Last Name	Email	Password	Role	Action
12	ChrisNieves23	Chris	Nieves	chris@email.com	password	ADMIN	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

-If the edit button is touched then will be redirected to another page where the user can edit only the role field. Here we will change it to user.

Edit User Role

User ID	<input type="text" value="12"/>
Username	<input type="text" value="ChrisNieves23"/>
Email	<input type="text" value="chris@email.com"/>
Role	<input type="text" value="User"/>
<input type="button" value="Save"/>	

Then it will change once we search for it again... The role will be changed to “User”.

User List

Search by Username <input type="button" value="Search"/>							
Search by User ID <input type="button" value="Search"/>							
ID	Username	Name	Last Name	Email	Password	Role	Action
12	ChrisNieves23	Chris	Nieves	chris@email.com	password	USER	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

-If the delete button is pressed then the user will not exist any more.

User List

Search by Username <input type="button" value="Search"/>							
Search by User ID <input type="button" value="Search"/>							
ID	Username	Name	Last Name	Email	Password	Role	Action

-If we go back to the admin page and click Review Management, we will be redirected to another page with a Comments/Reviews table containing the user who wrote it and the content of the comments that were reported by users or “flagged”

• [Back To Admin Page](#)

Flagged Comments

ID	User ID	Content	Flagged	Action
2	0	This game sucks	true	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
99	0	This game is doodoo	true	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

-This one is similar to the User management section. If you think a comment does not need to be flagged. You can change the Flagged status from the edit button, where it will direct you to the edit status form. Where you are only allowed to edit the flagged status of the review.

Edit Flagged Status

Comment ID	<input type="text" value="2"/>
User ID	<input type="text" value="0"/>
Content	<input type="text" value="This game sucks"/>
Flagged	<input type="button" value="Flagged"/> <input checked="" type="button" value="Not Flagged"/>
<input type="button" value="Save"/>	

-Let’s say we edit the flagged status to not true. Once the save button is executed then It will direct me back to the table and the comment that was edited Will not be there.

Flagged Comments

ID	User ID	Content	Flagged	Action
99	0	This game is doodoo	true	Edit Delete

-Let's say the remaining comment is too offensive so we can delete it we press the delete button and the comment will be gone from the table and deleted from the database as well.

Flagged Comments

ID	User ID	Content	Flagged	Action
----	---------	---------	---------	--------

-Then the user can go back to the admin page by hitting the top left corner button

Written Scenario (Publish/Delete Games) Jacob Echevarria:

Written Scenario(Search, add Wishlist, leave a review) Elijah Carpenter: