Software Requirements Specification Document

MMAD Music

1.0
September 17th, 2024

MMAD Productions
Mickael Agustin, Dan Rascop, Mikey Lupa, Aayaan Shaikh

Submitted in partial fulfillment of the requirements of
IT 326 – Principles of Software Engineering

Table of Contents

1. Introduction

1.1 Purpose

The document's purpose is to provide a detailed description of the music app system “MMAD Music”. It will explain the scope of functionality that the system will have and what user interactions will be possible.

1.2 Scope

This software system will allow users to create accounts with usernames and passwords. It will store login information and allow users to alter their existing login by resetting their password. This system is intended to allow its users to search for, rate, review, and favorite songs, albums, and artists. Each user will be able to cultivate their own music libraries and display their favorite song, album, and artist on their profile. In addition, users can friend other users to view their profiles and create a shared playlist of mutually saved music. The software will not play the music locally, but rather just display the information regarding the record. The software will allow users to recommend and share music with their friends, expanding their musical horizons and fostering social connection around music.

1.3 Definitions

Home Page – Page that will display the user’s favorited music elements
Friends page - Page that will display the user’s friended users
Reviews page - Page that will display the user’s reviews

1.4 Overview

The next section of this document is primarily written with the developers in mind. It is intended to provide a technical description of the system's functionality and expectations.

2. General Description

2.1 Product Perspective

This system is unique to other systems on the market because it allows users to rate and combine music with other users unlike other systems which do similar things but restrict their functions to other media. Spotify, the closest popular system comparatively like ours, does not have a social network or the ability to filter music based on multiple user's common taste. Letterbox’d is a similar app with much of the same functionality except that is solely used for film.

2.2 User Characteristics

The system’s user demographics are those capable of using a terminal and want to rate and share their music with their friends. Anyone who wants to connect to others through the world of music.

2.3 System Environment

This system will be run in a terminal environment on a PC using Java Development Kit (JDK). It will have a MySQL database hosted on Amazon’s Relational Database Service (RDS).

2.4 Assumptions and Dependencies

- The system assumes the MySQL database hosted on Amazon's Relational Database Service (RDS) is online, accessible by the software application and capable of returning queries and storing data. It also assumes that the free service tier with Amazon will continue to meet our development needs and that the authentication credentials are accessible and valid.
- The system assumes that the Spotify API will be publicly accessible for the various search functions, that the rate limits are high enough for our software's functionality, and that the data is returned in an expected format.
- Lastly, the system assumes proper internet connectivity between the application and the online MySQL RDS server, that the server is going to be online when the application is run, and that the application will be able to both read and write to and from the database.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Search Song

a. Description – *The system will prompt user for the track title that the user would like to find and return a list of ten most relevant albums based on the entered string.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Search for song' command.*

d. Conditions

i. Pre – *Search bar must not be empty.*

ii. Post – *Returns a list of ten most relevant tracks.*

3.1.2 Search Album

a. Description – *The system will prompt the user for the album title that the user would like to find and return a list of ten most relevant albums based on the entered string.*

b. Actor(s) – *User*

e. Trigger – *User enters the 'Search for album' command*

c. Conditions

i. Pre – *Search bar must not be empty.*

ii. Post – *Returns a list of ten most relevant albums.*

3.1.3 Search Artist

a. Description – *The system will prompt for artist/band name that the user would like to find and return a list of ten most relevant artist/bands based on the entered string.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Search for artist' command.*

d. Conditions

i. Pre – *The Profile has no album displayed.*
ii. Post – *Returns a list of ten most relevant artists/bands.*

3.1.4 Add Favorite Album

a. Description – *The system will allow the user to add the selected album to their profile.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Add to favorite album' command while an album is selected from the search results.*

d. Conditions

i. Pre – *The Profile has no favorite album displayed.*

ii. Post – *The system will store the album information to be displayed on the user's profile.*

3.1.5 Remove Favorite Album

a. Description – *The system will allow the user to remove the displayed album from the user's profile.*

b. Actor(s) – *User*

b. Trigger – *User enters the 'Remove favorite album' command.*

d. Conditions

i. Pre – *The Profile has an album displayed.*

ii. Post – *The album information will be removed from the user's profile.*

3.1.6 Add Favorite Song

a. Description – *The system will allow the user to add the selected song to their profile.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Add to favorite song' command while a song is selected from the search results.*

d. Conditions

i. Pre – *The Profile has no song displayed.*

ii. Post – *The system will store the track information to be displayed on the user's profile.*

3.1.7 Remove Favorite Song

a. Description – *The system will remove the displayed track from the user's profile.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Remove favorite artist' command.*

d. Conditions

iii. Pre – *The Profile has a song displayed.*

iv. Post – *The song information will be removed from the user's profile.*

3.1.8 Add Favorite Artist

a. Description – *The system will allow the user to add the selected artist to their profile.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Add to favorite artist' command while a song is selected from the search results.*

d. Conditions

i. Pre – *The Profile has no artist displayed.*

ii. Post – *The system will store the artist/band information to be displayed on the user's profile.*

3.1.9 Remove Favorite Artist

a. Description – *The system will remove the displayed artist from the user's profile.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Remove favorite artist' command.*

d. Conditions

i. Pre – *The Profile has an artist displayed.*

ii. Post – *The artist's information will be removed from the user's profile*

3.1.10 Delete Review

a. Description – *The system will allow a review to be deleted from the user's reviews page*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Delete review' command on a selected review.*

d. Condition

i. Pre – *The Profile has at least one review.*

ii. Post – *The review is deleted from the user's reviews page.*

3.1.11 Add Friend

a. Description – *The system will allow the user to add another user as a friend, allowing the target user to view the user's profile.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Add friend' command on a selected user.*

d. Conditions

i. Pre – *The target user is not already on the friends list. The target user has an existing account.*

ii. Post – *The system will add the target user to the user's friend list.*

3.1.12 Remove Friend

a. Description – *The system will allow a user to remove another user as a friend.*

b. Actor(s) – *User*

c. Trigger – *User enters the Remove friend' command on a selected friend.*

d. Conditions

i. Pre – *The target user is a friend; the target user account exists.*

ii. Post – *The target user will be removed from the user's friend list.*

3.1.13 Navigate to Friends list

a. Description – *The system will allow the user to navigate to the friends page. The friends page will list all the profiles of other users that are friends with the user.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Navigate to the friends page' command.*

d. Conditions

i. Pre – *User must be on their own profile.*

ii. Post – *User is navigated to their own friend's page.*

3.1.14 Search User

a. Description – *The system will prompt the user for the username that the user would like to find and return a list of ten most relevant usernames based on the entered string.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Search user' command.*

d. Conditions

i. Post – *username is associated with a user.*

ii. Post – *Returns a list of ten most relevant usernames.*

3.1.15 Navigate to Friend's Profile

a. Description – *The system will display the user's friend's profile including their favorite song, favorite album, and favorite artist. The system will also allow the user to navigate the friend's Reviews page from this profile.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'View friend's profile' command on a selected friend.*

d. Conditions
    i. Pre – *The user has at least one friend.*
    ii. Post – *The system will display the selected friend's profile*.

3.1.16 Create Account

a. Description – *The system will allow the user to enter a username and password to create a new account. The user will also be asked to answer a security question for account recovery*.

b. Actor(s) – *User*

c. Trigger – *User selects "Create Account" option in the starting menu.*

d. Conditions
    i. Pre – *Username must not already be in use by another account.*
    ii. Post – *Stores new username, password, and security question in database.*

3.1.17 Login

a. Description – *The system will allow the user to log in using their username and password upon startup.*

b. Actor(s) – *User*

c. Trigger – *User enters "Login" command option in start menu.*

d. Conditions
    i. Pre – *User must have already created a valid account.*
    ii. Post – *Logs user into account if credentials are correct. If credentials are incorrect, the user will be notified and allowed another chance to enter the correct credentials.*

3.1.18 Change Password

a. Description – *The system will allow the user to change their password.*

b. Actor(s) – *User*

c. Trigger – *User enters the "Change password" command while signed in.*

d. Conditions
    i. Pre – *User must have already created a valid account.*
    ii. Post – *User's account password will be updated in the database.*

3.1.19 Write Review

a. Description – *The system will allow the user to write a new review of a song or album which includes a 10-point rating.*

b. Actor(s) - User

c. Trigger – *User enters the "Create new review" command on the selected song or album.*

d. Conditions

i. Pre – *Item to be reviewed exists*

ii. Post – *Adds the new review to the user's Reviews page.*

3.1.20 Navigate to Reviews Page

a. Description – *The system will allow the user to navigate to the reviews page where they can view their reviews.*

b. Actor(s) – *User*

c. Trigger – *User enters the 'Navigate to the reviews page' command.*

d. Conditions

i. Pre – *User must be on their own profile.*

ii. Post – *User is navigated to their own reviews page.*

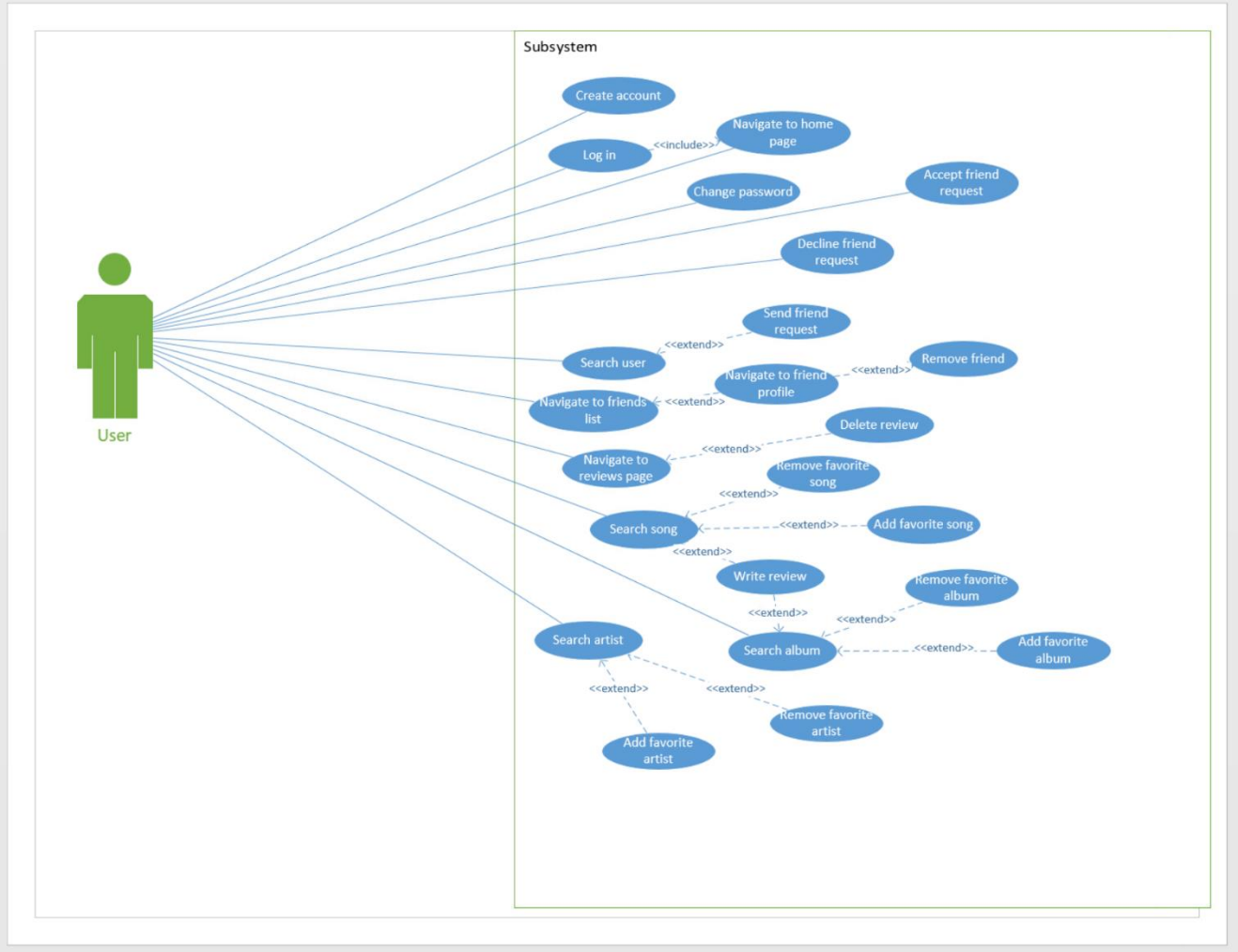


***Figure 1**. System Use Case Diagram*

3.2 Non-Functional Requirements

3.2.1 Performance

This system should respond to every user command in under 10 seconds.

3.2.2 Reliability

This system should perform predictably every time it's used.

3.2.3 Security

The team will follow best secure software development practices including securing account information and any data collected by the application.

3.2.4 Portability

This system should be functional on all devices capable of running JAVA code, with access to the database and a terminal.

4. Design & Development

The purpose of this section is to outline how the team plans to execute the design and development of the software product(s). It will cover the process model that the team chose to follow and justify it. It will also show a diagram of the process model that will provide details to our processes such as our sprint schedule and meeting frequency.

4.1 Software Development Process

The process model chosen for this project is that of the Kanban methodology while utilizing some aspects of Scrum model development. We picked the Scrum model as it fits easily into our weekly meeting schedule to conceptualize them as sprints. The Scrum model allows us to decompose large goals into manageable tasks to be completed each week while keeping an eye on group progress and getting updates each week. We use the Scrum model to plan and reach incremental development stages at the end of each sprint. At the start of each sprint, we create a plan to meet the due dates for deliverables and track our progress on our long-term project goals.

We meet every Monday from 5:30pm to 7:00pm to discuss updates, progress reports on key deliverables and any clarifications we need to ask the professor during Tuesday class. We communicate using the Slack application and maintain an open line of communication on urgent matters. We also do short informal check-ins during 355 class time just as an additional opportunity to discuss. Figure 2 visualizes the weekly workflow with the scrum process.

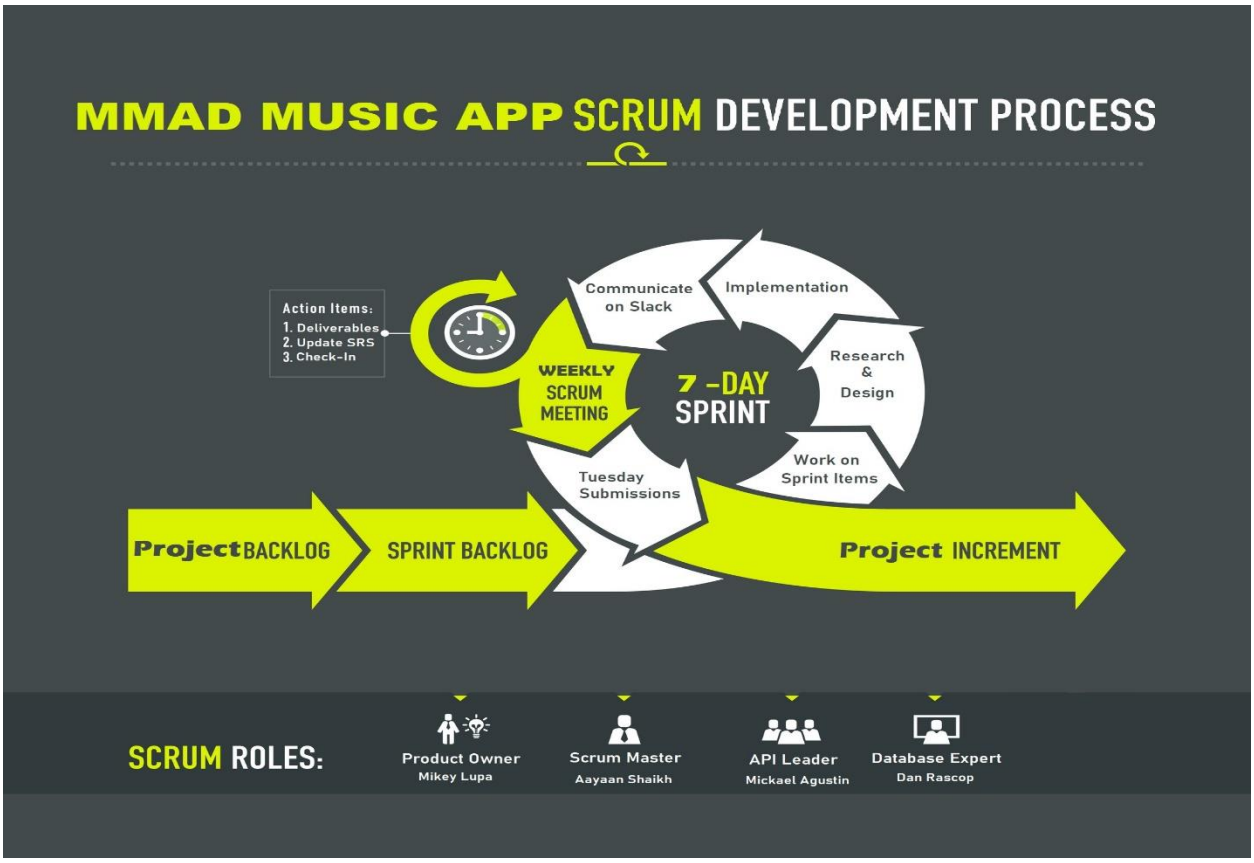


Figure 2. *Weekly Scrum Process Diagram*

We operate collaboratively, asking questions and getting clarifications from other team members as needed. Rather than the strict roles of typical Scrum development we opted to use a digital Kanban board maintained in our meeting notes document. The Kanban board helps us transparently visualize our near-term goals while maintaining a natural flow of task completion that is flexible enough to leave space for a teams' other obligations. The roles are flexible enough to allow each team member to have experience with each component of the project improving communication and inter-team collaboration. Figure 3 shows the digital Kanban board.

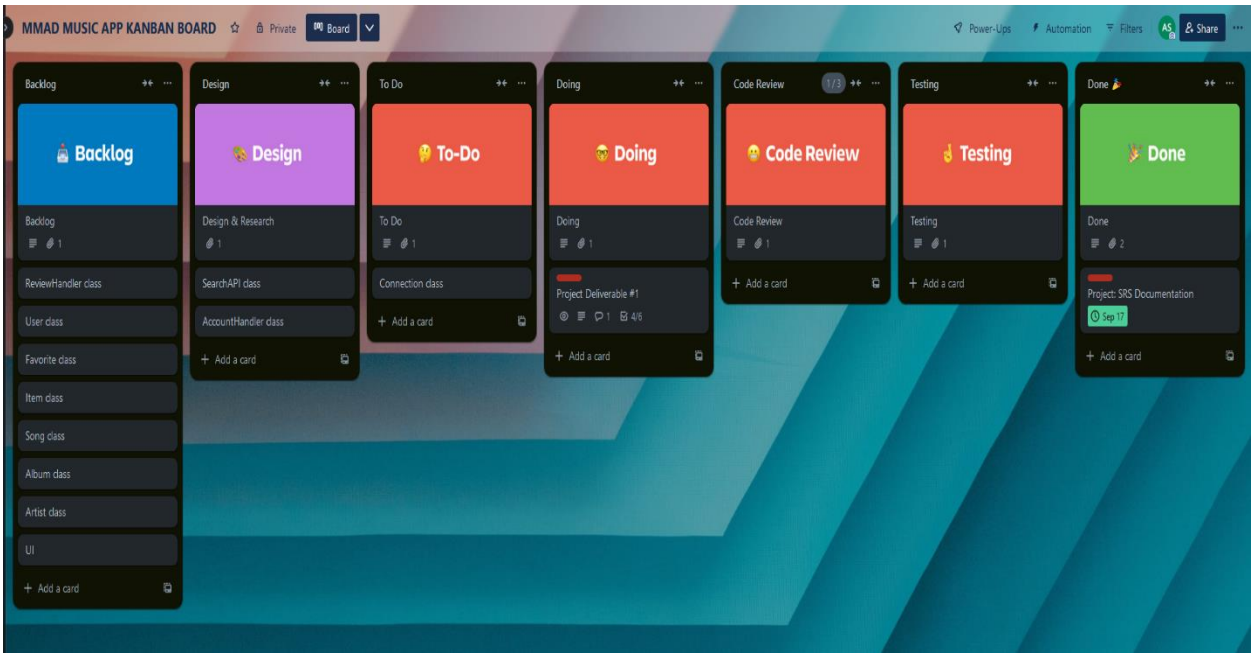


Figure 2. *Kanban Board*

4.2 Deliverable 1

4.2.1 Description

For this deliverable we have finished outlining our system design and have included and accounted for each system component we will use. We are ready to implement the design and develop the program. We have represented our previously established process design in a development process model and visualized our work items on a Kanban board. We have developed frameworks for our domain classes and have started to work on the database connection and AccountHandler.

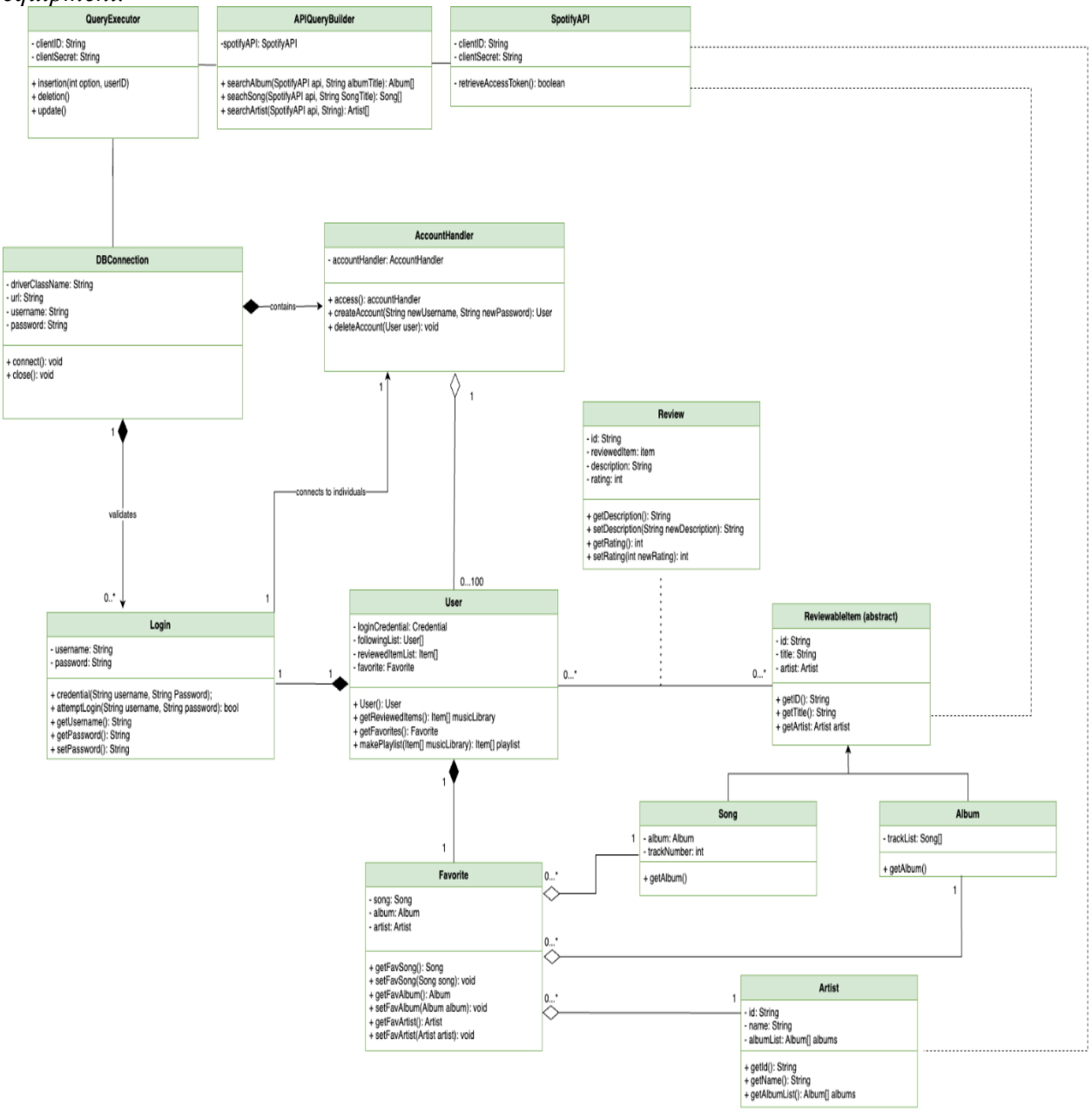
We discussed at length the overall structure of our program. As we were developing the class diagram, we were confronted with the ambiguities of our design. We were forced to re-design the domain classes multiple times to maintain good program design. Much time was spent discussing and arguing about how to design the interaction between the domain classes and the database. It was difficult to conceptualize the "in-between" classes we needed but ended up with the QueryExecutor and APIQueryBuilder classes to communicate with the database and the Spotify API.

NEEDS ONE MORE PARAGRAPH ON STUFF WE WORKED ON

4.2.2 Design

4.2.2.1 Class Diagram

This section explains the class diagram in detail. The explanation includes the responsibility of each class/abstraction and associations between them. For example,
The class diagram showed in Figure 1 captures the vocabulary of the system in terms of Equipment – captures all the equipment leased and stored in the warehouse, Transaction – captures the transaction information related to an equipment, Aisle – captures the location of an equipment.



4.2.2.2 Activity Diagram

This section explains each activity diagram in detail. The explanation should include the requirement it is modeling. For example,

The activity diagram shown in Figure 3 explains the activities involved and their flow for creating an account (Use case 17: Create Account).

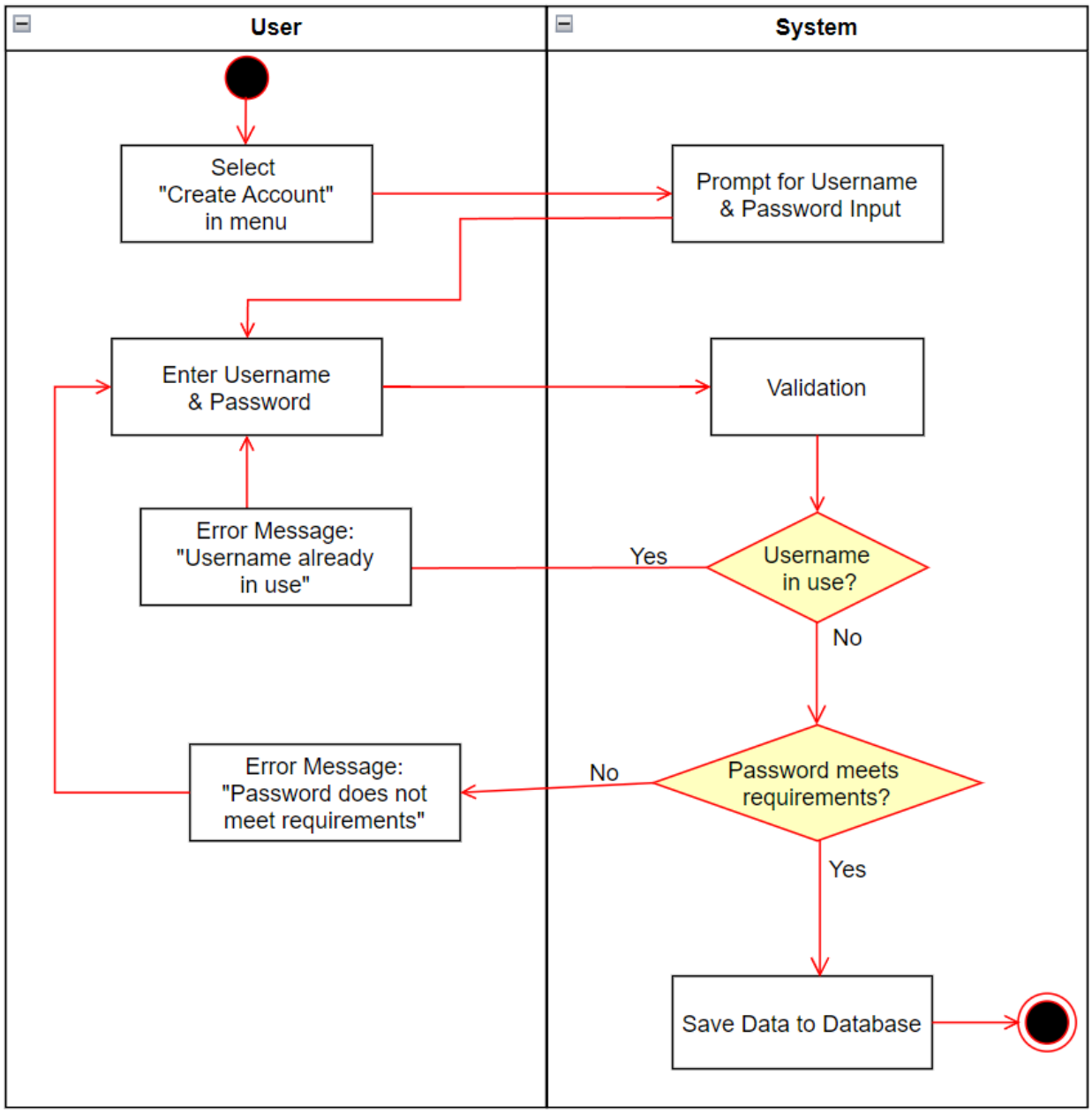


Figure 3. *Activity Diagram for "Create Account" Use Case*

The activity diagram shown in Figure 4 explains the activities involved and their flow for logging into an account (Use case 18: Login).

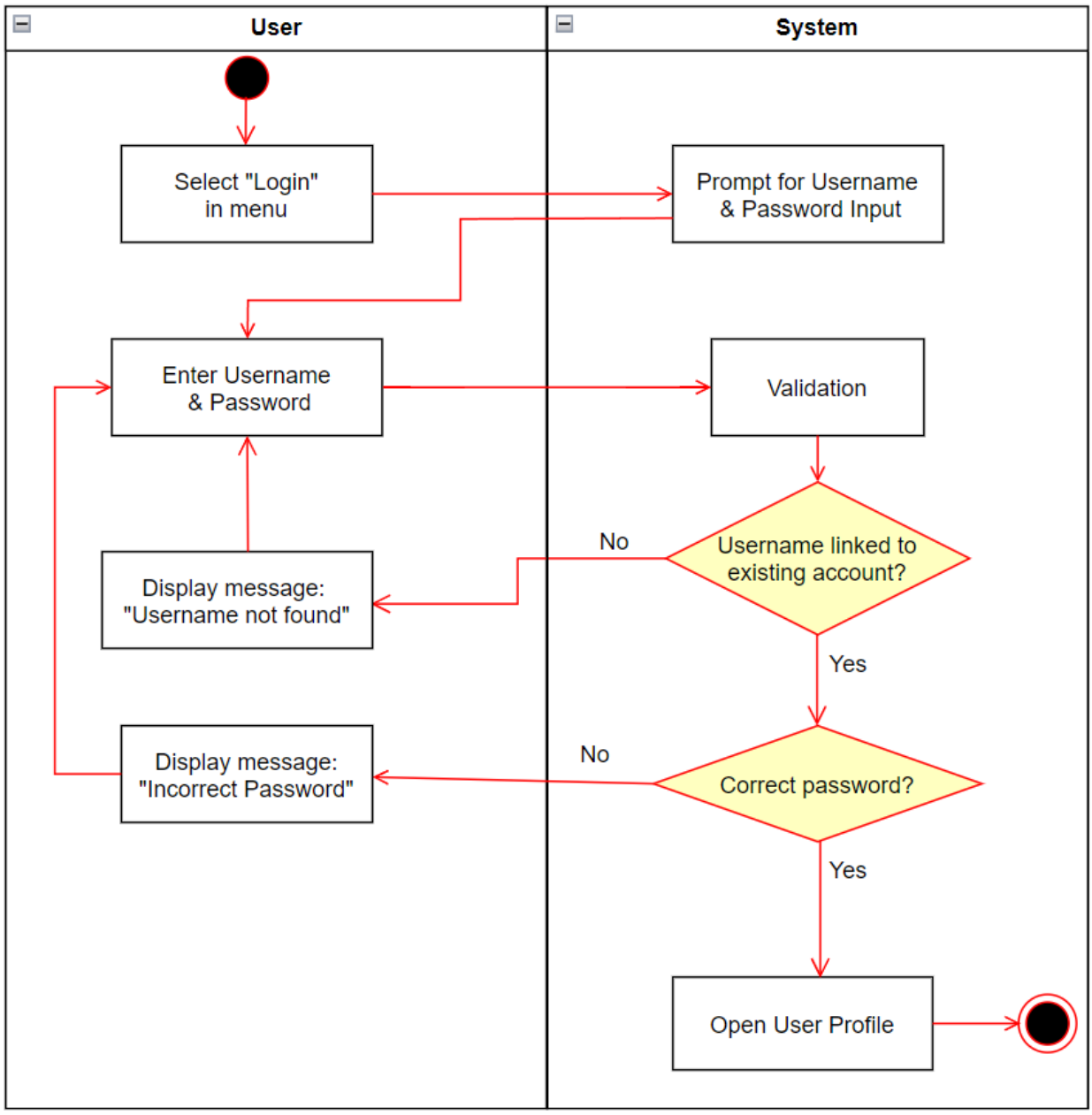


***Figure 4**. Activity Diagram for "Login" Use Case*

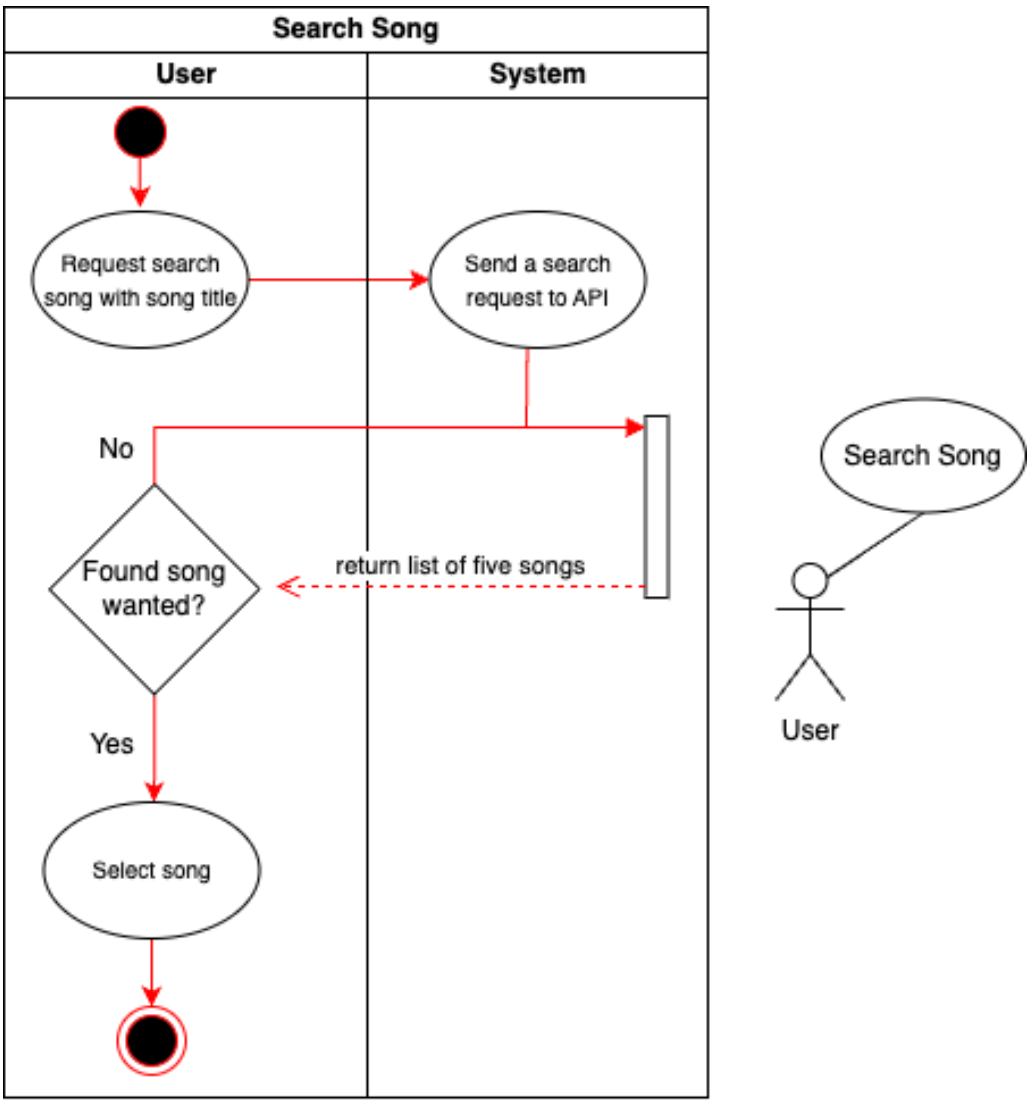


Figure 5. *Activity Diagram for "Search Song" Use Case*

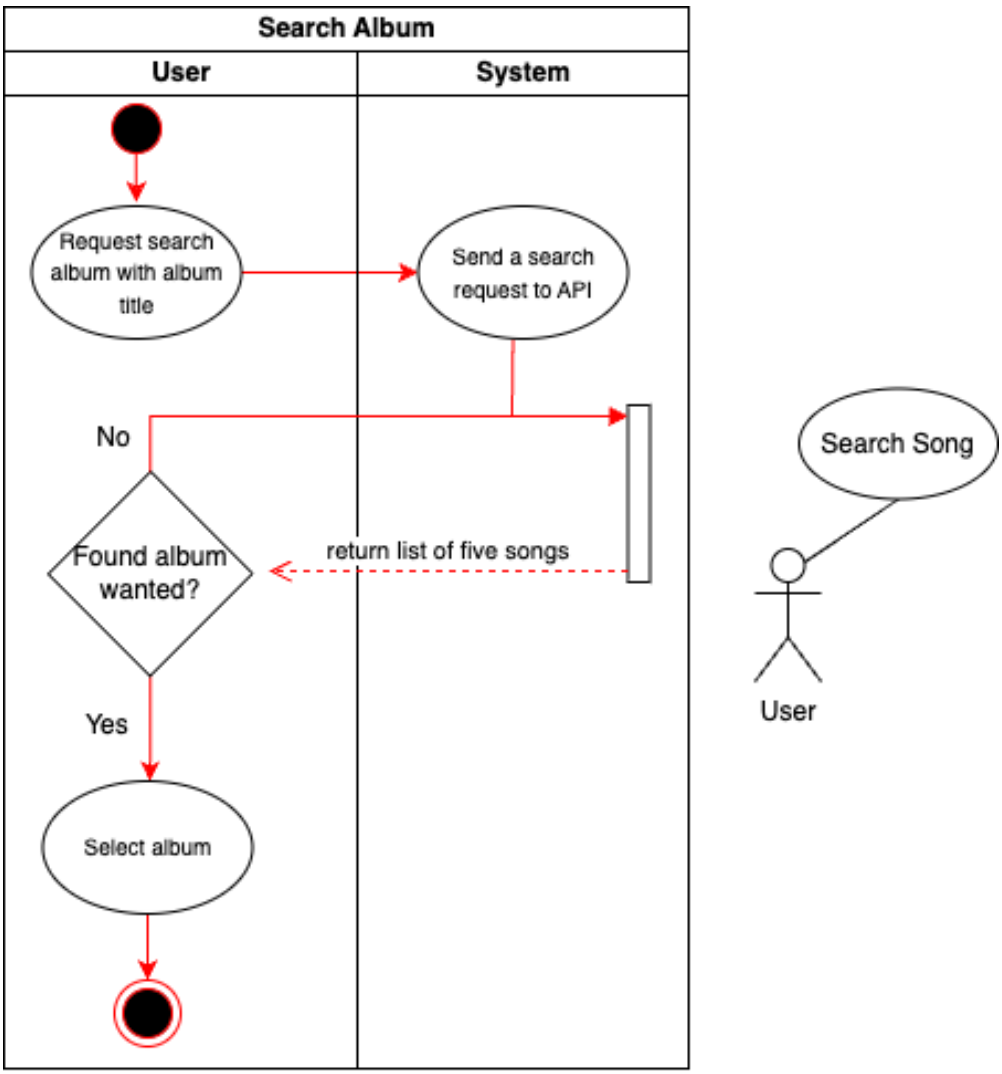


***Figure 6**. Activity Diagram for "Search Album" Use Case*

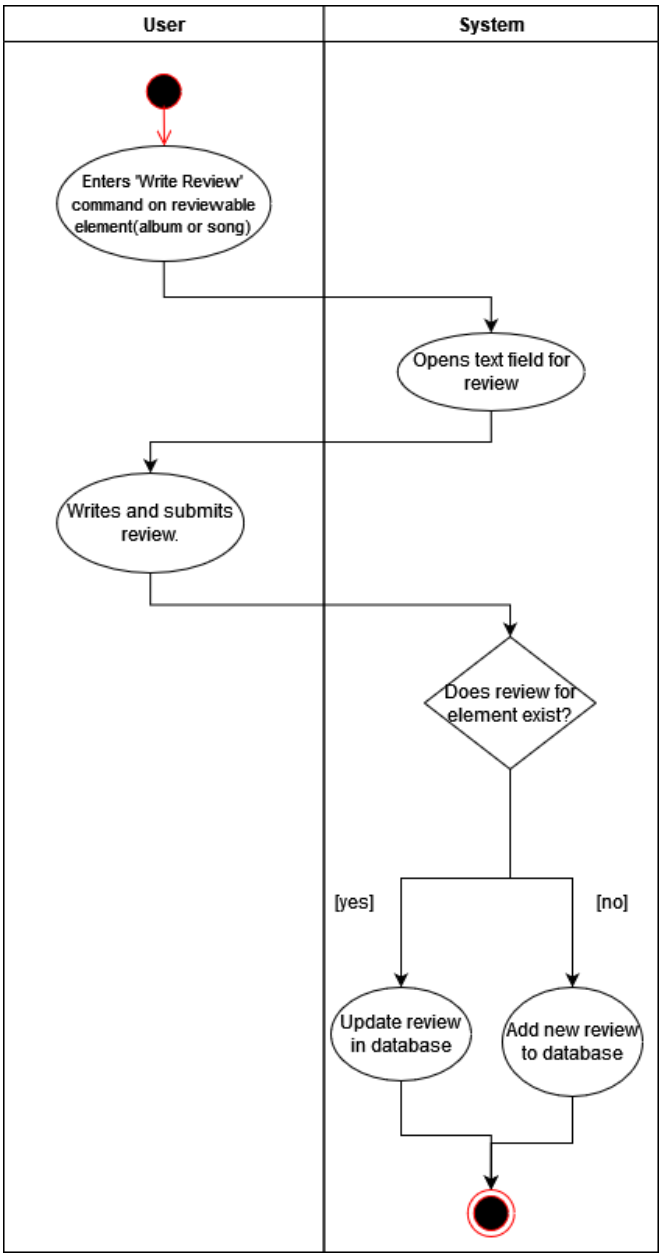


***Figure 7**. Activity Diagram for "Write Review" Use Case*

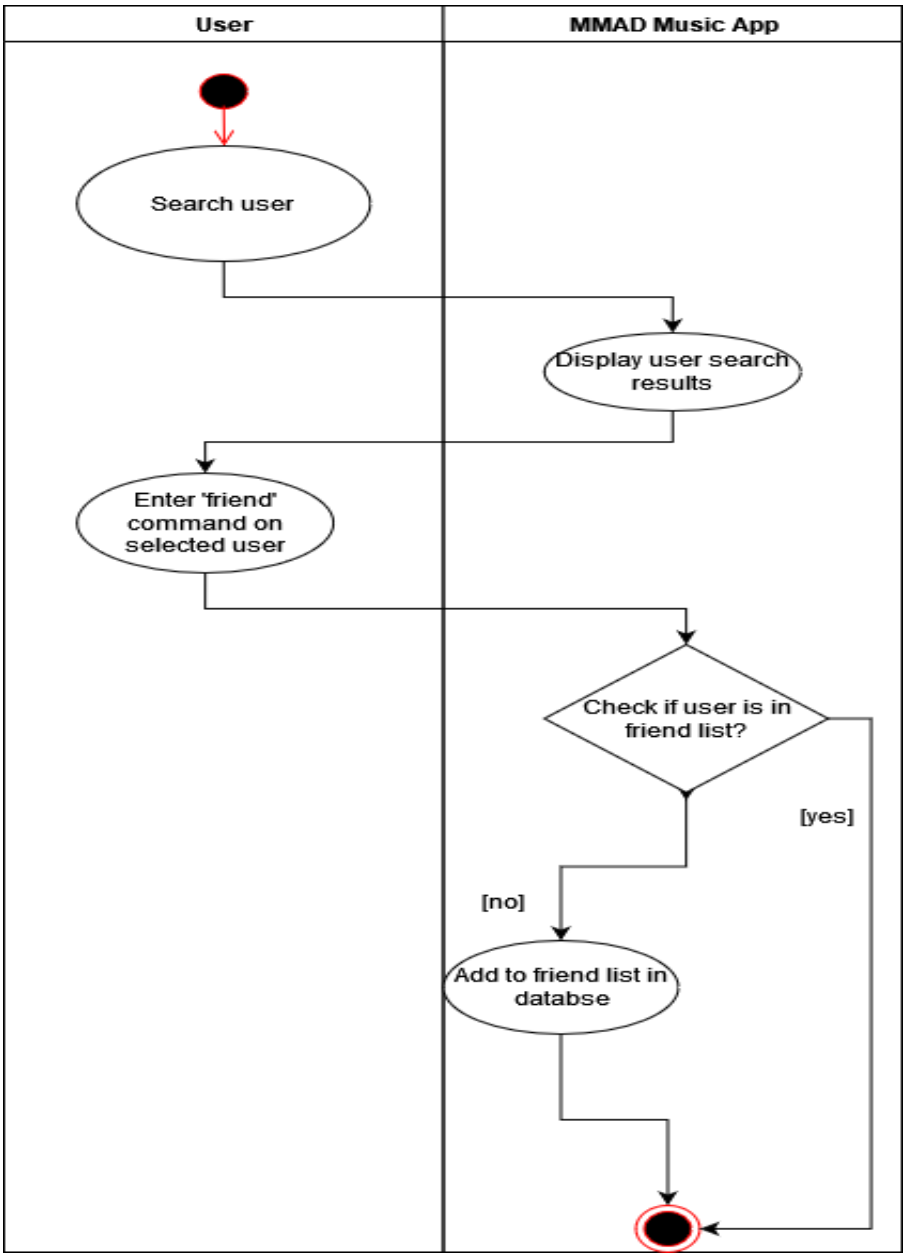


Figure 8. *Activity Diagram for "Add Friend" Use Case*

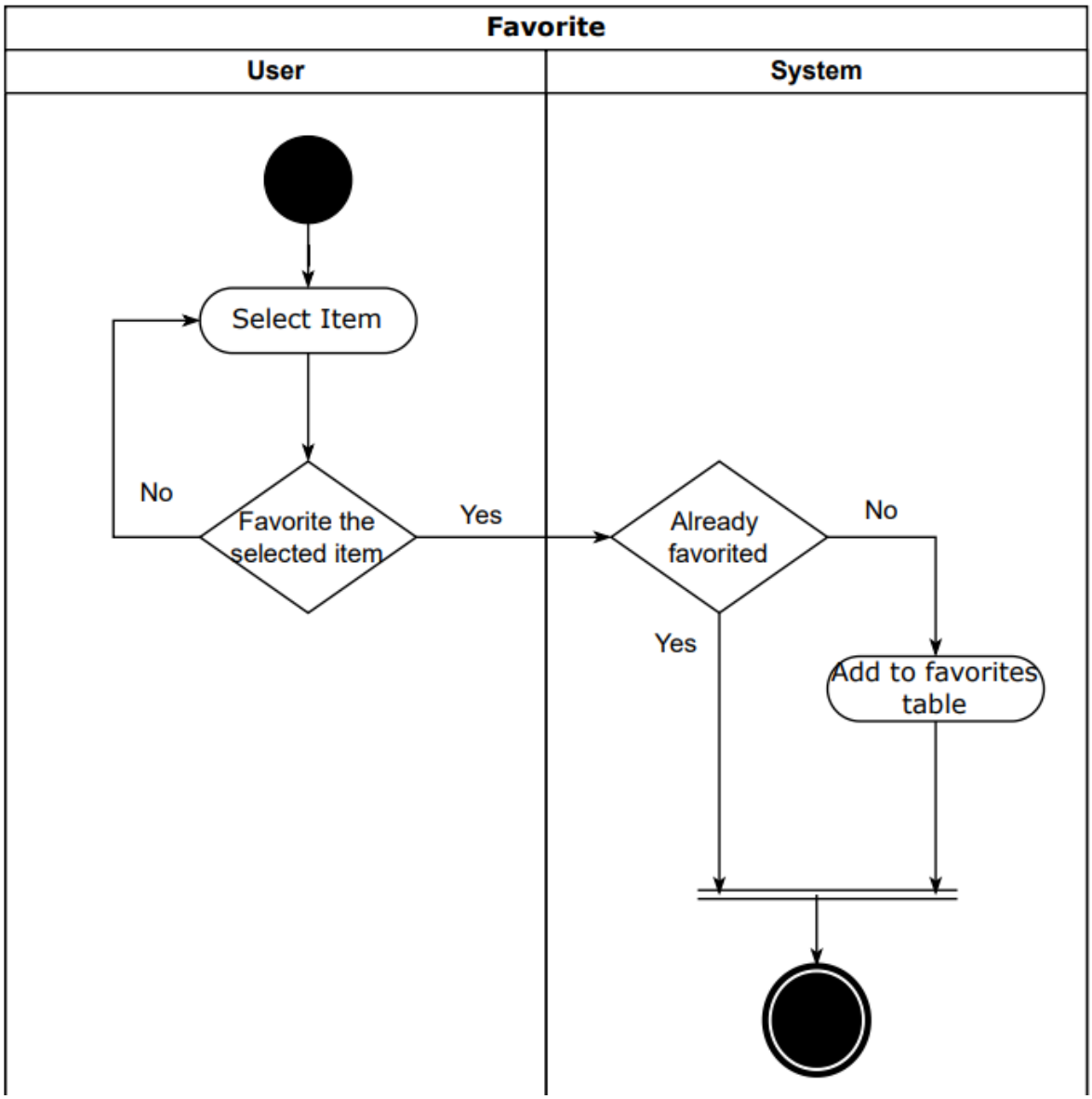


Figure 9. *Activity Diagram for "favorite" Use Case – Mikey Lupa*

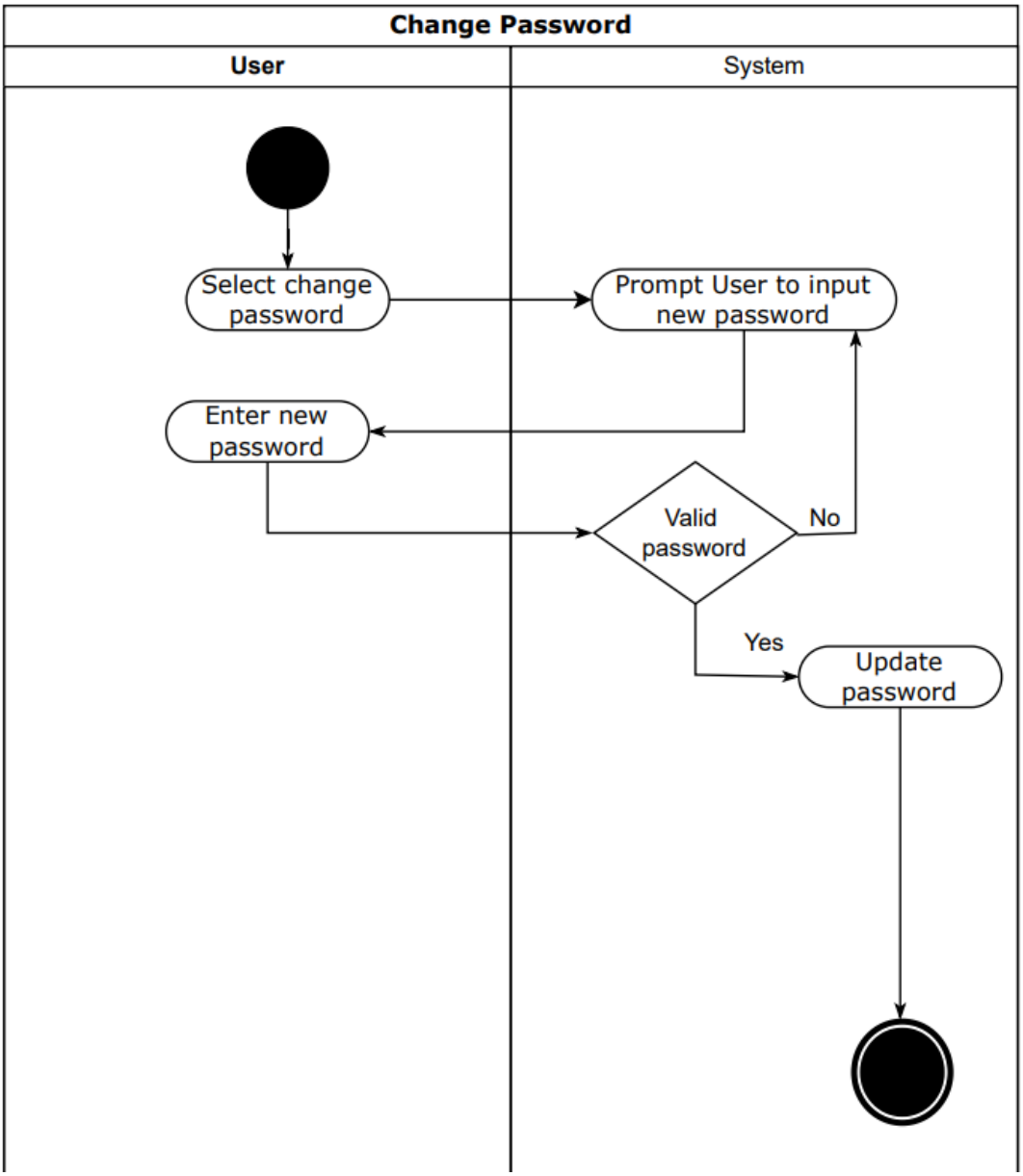


***Figure 10**. Activity Diagram for "Change password" Use Case – Mikey Lupa*

4.2.3 Development

4.2.3.1 Description

*The team MUST have started implementation of the class model or **at least finalized** the development environment (databases, Git, etc.). The team **MUST** provide me their git repository link with all team members on the repo.*
This section describes the development efforts for the deliverable, if any. The team can show their progress on their development by include screenshots of their application or code.

Git Repository: https://github.com/chewmonguy225/MMAD-Music