**Software Requirements Specification Template**

CrossFilmz

Software Requirements Specification

Version 1.0

6/9/2020

Anthony Gruber, Avery Peiffer, Samir Sherlaker, Carter Smith

Software Engineers

Prepared for

CS 1530 Music & Movie Group

# **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 6/9/2020 | Version 1 | CS Music & Movie Group | First Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Name** | **Title** | **Date** |
|  | Avery Peiffer | Software Eng. | 6/9/2020 |
|  | Carter Smith | Software Eng. | 6/9/2020 |
|  | Anthony Gruber | Software Eng. | 6/9/2020 |
|  | Samir Sherlaker | Software Eng. | 6/9/2020 |

**Table of Contents**

**REVISION HISTORY** [**I**](https://docs.google.com/document/d/1IJVZQUwgZweW8S4qjkK8oO_9YmnnsJZf/edit#heading=h.1hmsyys)

**DOCUMENT APPROVAL** [**I**](https://docs.google.com/document/d/1IJVZQUwgZweW8S4qjkK8oO_9YmnnsJZf/edit#heading=h.41mghml)

**TABLE OF CONTENTS I**[**I**](https://docs.google.com/document/d/1IJVZQUwgZweW8S4qjkK8oO_9YmnnsJZf/edit#heading=h.41mghml)

**1. INTRODUCTION** [**1**](https://docs.google.com/document/d/1IJVZQUwgZweW8S4qjkK8oO_9YmnnsJZf/edit#heading=h.2grqrue)

1.1 Purpose [1](https://docs.google.com/document/d/1IJVZQUwgZweW8S4qjkK8oO_9YmnnsJZf/edit#heading=h.vx1227)

1.2 Scope [1](https://docs.google.com/document/d/1IJVZQUwgZweW8S4qjkK8oO_9YmnnsJZf/edit#heading=h.3fwokq0)

1.3 Definitions, Acronyms, and Abbreviations [1](https://docs.google.com/document/d/1IJVZQUwgZweW8S4qjkK8oO_9YmnnsJZf/edit#heading=h.1v1yuxt)

**2. GENERAL DESCRIPTION 1**

2.1 Product Perspective 1

2.2 Product Functions 1

2.3 Constraints, Assumptions and Dependencies 2

**3. SPECIFIC REQUIREMENTS 2**

3.1 External Interface Requirements 2

*3.1.1 Software Interfaces 2*

3.2 Functional Requirements 2

*3.2.1 User login page 2*

*3.2.2 User account registration page 2*

*3.2.3 Search bar for finding movies 2*

*3.2.4 Ability to add a movie rating out of five stars 3*

*3.2.5 User profile containing previous ratings 3*

*3.2.6 Generate list of recommendations 3*

*3.2.7 Filter based on streaming service 3*

*3.2.8 Filter based on genre or other attribute 3*

*3.2.9 ‘More info’ link when clicking on each movie 3*

*3.2.10 Display Movies as a Responsive Grid 3*

3.3 Use Cases 3

*3.3.1 View previous ratings on user profile page 3*

*3.3.2 Click on a light bulb icon in the sidebar to generate a grid of recommended movies 4*

*3.3.3 Clicking on movie tile links to IMDb (or equivalent site) for more information 4*

*3.3.4 Clicking on logo of streaming platform filters results 5*

*3.3.5 Add items to a watch list 6*

*3.3.6 Login and register using Google credentials 6*

*3.3.7 User should search for movies with an autocomplete textbox 6*

*3.3.8 User can dynamically resize window with a responsive grid (makes it usable on mobile) 6*

*3.3.9 User can navigate the website using a sidebar 6*

*3.3.10 User should be able to rate movies from the recommendation results 6*

3.4 Classes / Objects 6

*3.4.1 Movie Class 6*

*3.4.1.A Attributes 6*

*3.4.1.B Methods 6*

*3.4.2 Shelf Class 7*

*3.4.2.A Attributes 7*

*3.4.2.B Methods 7*

*3.4.3 User Class 7*

*3.4.3.A Attributes 7*

*3.4.3.B Methods 7*

*3.4.4 Search Bar Class 7*

*3.4.4.A Attributes 7*

*3.4.4.B Methods 7*

*3.4.5 Grid Class 7*

*3.4.5.A Attributes 7*

*3.4.5.B Methods 7*

*3.4.6 App Class 7*

*3.4.6.A Attributes 7*

*3.4.6.B Methods 7*

3.5 Non-Functional Requirements 8

*3.5.1 Performance 8*

*3.5.2 Reliability 8*

*3.5.3 Availability 8*

*3.5.4 Security 8*

*3.5.5 Maintainability 8*

*3.5.6 Portability 8*

*3.5.7 Documentation 8*

3.6 Logical Database Requirements 8

3.7 Other Requirements 8

**4. ANALYSIS MODELS 8**

4.1 Use Case Diagrams 9

4.2 Class Diagrams 10

4.3 Sequence Diagrams 10

4.4 Activity Diagrams (DFD) 11

4.5 State-Transition Diagrams (STD) 13

**5. CHANGE MANAGEMENT PROCESS 13**

# **1. Introduction**

CrossFilmz is a movie rating and recommendation service that focuses on titles hosted on streaming platforms. It aims to streamline the process for finding out if and where a movie is available for streaming while providing useful, personalized recommendations to users.

## **1.1 Purpose**

This document will serve as a comprehensive overview of the CrossFilmz platform, including its purpose, requirements, and models. It is intended to aid Professor Sarwar, our peers, and ourselves in keeping a clear structure to the project as we begin the implementation phase beginning with Sprint 1.

## **1.2 Scope**

## The product will collect an individual user’s ratings for movies that they have watched, then use a content-based filtering system to recommend titles with similar attributes. CrossFilmz will serve as a convenient tool for those who subscribe to multiple streaming services, especially as titles change platforms with regularity. The system will maintain profiles for users so that their past ratings can be used to inform future recommendations.

## **1.3 Definitions, Acronyms, and Abbreviations**

Content-based recommender/recommendation system – A technique of making recommendations that is based on what an individual user likes or has purchased. Items are tagged with keywords that represent their attributes. The system will then recommend items with the same keywords as those that the user prefers.*.*

# **2. General Description**

This section outlines our high-level conception of the CrossFilmz platform, including perspective, overviews of the product’s functions, as well as constraints and assumptions that will need to be taken into account during development.

## **2.1 Product Perspective**

A significant inspiration for this project is the website *agoodmovietowatch*. It has a similar focus on movies available for streaming, though it lacks a true recommender system. Instead, movies are curated based on the reviews of humans.

## **2.2 Product Functions**

The software will have the following functions:

* User profiles containing past ratings
* Movie rating system
* Recommender system that generates movie recommendations based on a user’s ratings
* FIltering system based on streaming platform
* Links to IMDb (or equivalent) for further information about a recommended movie

## **2.3 Constraints, Assumptions and Dependencies**

Our filter will be limited by the number of streaming platforms for which there is readily available data regarding the titles they host, which will be found using the Utelly API. However, this should not be an issue for the platforms of considerable size, such as Netflix and Hulu. The Utelly API is also a significant dependency for this project.

Perhaps the most considerable constraint on the development of this project will be tagging the movies with attributes for the content-based recommendation system. The movies will need to have a detailed set of attributes for the system to provide useful recommendations. The OMDb API will be used to retrieve information about an individual movie; however, it could be prudent to start with a small, representative set of movies so as to not waste time on simple data entry.

The scope of the platform does make the assumption that a user subscribes to one or more streaming platforms; if a user does not use streaming services, there will be little to gain from using the platform.

# **3. Specific Requirements**

This section outlines the specific requirements necessary for the CrossFilmz platform, including functional/non-functional requirements, use cases, and classes.

## **3.1 External Interface Requirements**

### **3.1.1 Software Interfaces**

The Utelly API will be used to filter titles based on their availability across streaming services. Additionally, the OMDb API will be used to retrieve information about an individual title, both for filling out the movie’s attributes and providing useful facts to the user.

## **3.2 Functional Requirements**

## **3.2.1 User login page**

The login page should allow the user to log in to an existing profile and link to the page for registering a new profile. It should login and register using Google credentials.

**3.2.2 User account registration page**

The account registration page should have several fields for user information including, but not limited to: name, email, password. These fields should be used as the user’s credentials for the service.

### **3.2.3 Search bar for finding movies**

The search bar should search for movies when an entry is completed, as well as provide autocomplete suggestions while the user is typing.

### **3.2.4 Ability to add a movie rating out of five stars**

The user should be able to do this on the page for an individual movie. The user’s rating will then be stored in the database.

### **3.2.5 User profile containing previous ratings**

An icon that links to the user’s profile should be available on the main page. The profile should contain their past ratings from the database. These ratings should be synthesized into the types of movies that the user prefers. The profile will also contain the watch list that the user has cultivated from using the service.

### **3.2.6 Generate list of recommendations**

The recommendations should be generated when the user clicks a button, either in the main view or on the sidebar. These recommendations should then be filterable based on streaming platforms.

**3.2.7 Filter based on streaming service**

There should be logos for each of the supported streaming services on the side of the view. These logos should initially be faded so as to show that no filtering is being done. Clicking on one of the logos should illuminate it and cause the results to be filtered towards that streaming service.

### **3.2.8 Filter based on genre or other attribute**

There should be other options for filtering based on the attributes that are used to tag the movies. This will most likely be filtered based on genre, but other filters could be implemented depending on the various data points that are retrieved from the OMDb API call.

### **3.2.9 ‘More info’ link when clicking on each movie**

Each movie in the search results should have a link to its IMDb or Wikipedia page for more information. The user should be able to access this link by clicking on a movie’s tile in the search results.

### **3.2.10 Display Movies as a Responsive Grid**

The search results should be laid out in a grid view and displayed in an aesthetically pleasing manner. The ‘More Info’ link should appear when the user mouses over a title.

## **3.3 Use Cases**

Ten CrossFilmz use cases are outlined here.

### **3.3.1 View previous ratings on user profile page**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | 1 | | | |
| **Use Case Name:** | View previous ratings on user profile page | | | |
| **Created By:** | Avery Peiffer | | **Last Updated By:** | Avery Peiffer |
| **Date Created:** | 6/7/2020 | | **Last Revision Date:** | 6/9/2020 |
| **Actors:** | | Registered user | | |
| **Description:** | | The user’s profile page should contain their past ratings as well as a brief summary of the types of movies that they have rated highly. | | |
| **Trigger:** | | Clicking on the ‘User Profile’ icon in the main view | | |
| **Preconditions:** | | On main page | | |
| **Postconditions:** | | On user profile page | | |
| **Normal Flow:** | | Click user profile icon, navigate to user profile | | |
| **Alternative Flows:** | | Click ‘Rate a movie’, navigate to user profile | | |
| **Exceptions:** | | The user has not registered for a profile yet - a message could be displayed that links to the profile registration page  The user has not rated any movies yet - a message could be displayed suggesting that the user rates movies, with a link to the search functionality | | |
| **Includes:** | | None | | |
| **Frequency of Use:** | |  | | |
| **Special Requirements:** | | None | | |
| **Assumptions:** | | The user has rated movies and has a profile already registered on the service | | |
| **Notes and Issues:** | |  | | |

### 

### **3.3.2 Click on a light bulb icon in the sidebar to generate a grid of recommended movies**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | 2 | | | |
| **Use Case Name:** | Click on a light bulb icon in the sidebar to generate a grid of recommended movies | | | |
| **Created By:** | Avery Peiffer | | **Last Updated By:** | Avery Peiffer |
| **Date Created:** | 6/7/2020 | | **Last Revision Date:** | 6/9/2020 |
| **Actors:** | | Registered user | | |
| **Description:** | | The light bulb icon in the sidebar should trigger the user class to get recommendations. These should then appear in a grid format on the main view. | | |
| **Trigger:** | | Clicking the light bulb icon in the sidebar | | |
| **Preconditions:** | | On main page | | |
| **Postconditions:** | | On main page with grid of movies | | |
| **Normal Flow:** | | Click on light bulb icon and grid is generated dynamically | | |
| **Alternative Flows:** | | Click on ‘Get Recommendations’ and grid is generated dynamically | | |
| **Exceptions:** | | The user has not entered any recommendations yet - this could be fixed by preventing the user from clicking the light bulb icon in the first place | | |
| **Includes:** | | None | | |
| **Frequency of Use:** | |  | | |
| **Special Requirements:** | | None | | |
| **Assumptions:** | | The user has assigned ratings to movies previously | | |
| **Notes and Issues:** | |  | | |

### 

### **3.3.3 Clicking on movie tile links to IMDb (or equivalent site) for more information**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | 3 | | | |
| **Use Case Name:** | Clicking on movie tile links to IMDb (or equivalent site) for more information | | | |
| **Created By:** | Avery Peiffer | | **Last Updated By:** | Avery Peiffer |
| **Date Created:** | 6/7/2020 | | **Last Revision Date:** | 6/9/2020 |
| **Actors:** | | Unregistered user, registered user | | |
| **Description:** | | When the user mouses over a movie tile in the grid, the title should become blurry and a “More Info” link should become visible. Clicking on this should open a new tab on IMDb or a similar website that allows the user to get more information about the movie. | | |
| **Trigger:** | | Clicking on a movie tile in the grid view | | |
| **Preconditions:** | | On main page with grid of movies | | |
| **Postconditions:** | | On main page with grid of movies, additional tab open with more info | | |
| **Normal Flow:** | | Clicking movie tile on grid opens new tab with more info | | |
| **Alternative Flows:** | | None | | |
| **Exceptions:** | | A movie has no IMDb page - this could be worked around by not showing the ‘More Info’ option in the first place | | |
| **Includes:** | | None | | |
| **Frequency of Use:** | |  | | |
| **Special Requirements:** | | None | | |
| **Assumptions:** | | Movies have a similar format for their IMDb pages so the links can be easily generated | | |
| **Notes and Issues:** | |  | | |

### 

### **3.3.4 Clicking on logo of streaming platform filters results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | 4 | | | |
| **Use Case Name:** | Clicking on logo of streaming platform filters results | | | |
| **Created By:** | Avery Peiffer | | **Last Updated By:** | Avery Peiffer |
| **Date Created:** | 6/7/2020 | | **Last Revision Date:** | 6/9/2020 |
| **Actors:** | | Registered user | | |
| **Description:** | | The supported streaming platforms will be presented off to the side of the grid view, initially grayed out to show that no filtering has been done. When the user clicks on a logo for a streaming service, it will filter the recommendations in the grid view to only show those which are hosted on that streaming platform. | | |
| **Trigger:** | | Clicking on the logo of a streaming service | | |
| **Preconditions:** | | On main page with grid of recommended movies | | |
| **Postconditions:** | | On main page with grid of filtered, recommended movies | | |
| **Normal Flow:** | | Click on logo of one or more streaming services and update grid | | |
| **Alternative Flows:** | | Click on logo of streaming services before getting recommendations | | |
| **Exceptions:** | | There are no results. This can be worked around by not allowing the user to filter if there are no results. | | |
| **Includes:** | | None | | |
| **Frequency of Use:** | |  | | |
| **Special Requirements:** | | None | | |
| **Assumptions:** | | The user has a profile and is able to receive recommendations. | | |
| **Notes and Issues:** | |  | | |

### 

### 

### **3.3.5 Add items to a watch list**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | 5 | | | |
| **Use Case Name:** | Add items to a watch list | | | |
| **Created By:** | Avery Peiffer | | **Last Updated By:** | Avery Peiffer |
| **Date Created:** | 6/7/2020 | | **Last Revision Date:** | 6/9/2020 |
| **Actors:** | | Registered user | | |
| **Description:** | | When movie recommendations are laid out in a grid view, the user should have the option to add a title to their watch list. This watch list will be shown on their user profile. | | |
| **Trigger:** | | ‘Add to Watch List’ button in grid view | | |
| **Preconditions:** | | On main page with grid of recommended movies | | |
| **Postconditions:** | | On main page with grid of recommended movies, user profile has been updated | | |
| **Normal Flow:** | | Hover over movie tile and click to add to watch list | | |
| **Alternative Flows:** | | None | | |
| **Exceptions:** | | The user has not created a profile yet - could have the button redirect the user to the account creation page | | |
| **Includes:** | | None | | |
| **Frequency of Use:** | |  | | |
| **Special Requirements:** | | None | | |
| **Assumptions:** | | The user has made a profile | | |
| **Notes and Issues:** | |  | | |

### 

### **3.3.6 Login and register using Google credentials**

### **3.3.7 User should search for movies with an autocomplete textbox**

### **3.3.8 User can dynamically resize window with a responsive grid (makes it usable on mobile)**

### **3.3.9 User can navigate the website using a sidebar**

### **3.3.10 User should be able to rate movies from the recommendation results**

## **3.4 Classes / Objects**

### **3.4.1 Movie Class**

3.4.1.A Attributes

* String title
* String runtime
* String genre
* String language
* String country
* String posterLink
* float imdbRating

3.4.1.B Methods

* getTitle()
* getRuntime()
* getGenre()
* getLanguage()
* getCountry()
* getPosterLink()
* getImdbRating()

### **3.4.2 Shelf Class - aggregation with Movies**

3.4.2.A Attributes

* List Movies[]

3.4.2.B Methods

* getMovies()

### **3.4.3 User Class**

3.4.3.A Attributes

* String user
* Shelf ratedMovies
* Shelf recommendedMovies

3.4.3.B Methods

* getRecommendations()

**3.4.4 Search Bar Class**

3.4.4.A Attributes

* List validMovies[]

3.4.4.B Methods

* render()

**3.4.5 Grid Class**

3.4.5.A Attributes

* Boolean isLoading
* Shelf moviesToBeRendered

3.4.5.B Methods

* didComponentUpdate()
* render()

**3.4.6 App Class**

3.4.6.A Attributes

* Grid grid
* Search search

3.4.6.B Methods

* render()

## **3.5 Non-Functional Requirements**

### **3.5.1 Performance**

* The time from movie search to the time movies are rendered on the grid below should be under 1000ms

### **3.5.2 Reliability**

* After system failure, the system will retrieve last saved state for the user

### **3.5.3 Availability**

* System downtime may not exceed one minute per day

### **3.5.4 Security**

* System should use one OAuth endpoint for logins to minimize attack surface

### **3.5.5 Maintainability**

* Deploy changes in one commit accomplished via Github Actions
* Mean Time to Change (MTTC) for defects is less than 3 days

### **3.5.6 Portability**

* The website works on a variety of browsers, including Google Chrome and Firefox
  + The website also maintains layout on mobile and web browsers

**3.5.7 Documentation**

* The source code is documented by putting method comments in class docstring above each class

## **3.6 Logical Database Requirements**

CrossFilmz will utilize Datastore, a NoSQL database provided by the Google Cloud platform. Since the project’s Python backend will interact with the database, there will be no obstructive restrictions on data types. Instead, all data will simply be stored as a Python map. Data retention, integrity, consistency, and other such concerns are reliably handled by Google. As for the limitations on Datastore, CrossFilmz is highly unlikely to outstretch its boundaries. The maximum transaction size is 10 MiB, the maximum size of any given database entity is 4 bytes shy of 1 MiB, and the maximum write rate to a specific entity is 1 write per second. Moreover, because the magnitude of operations on the database will be small, Datastore will be free.

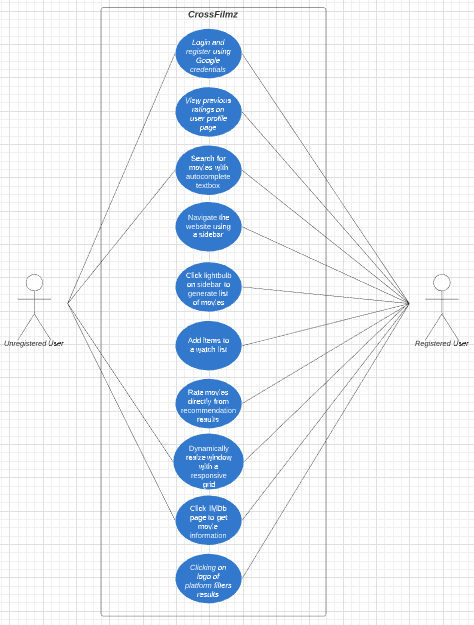
## **3.7 Other Requirements**

# **4. Analysis Models**

The UML analysis models are presented in this section.

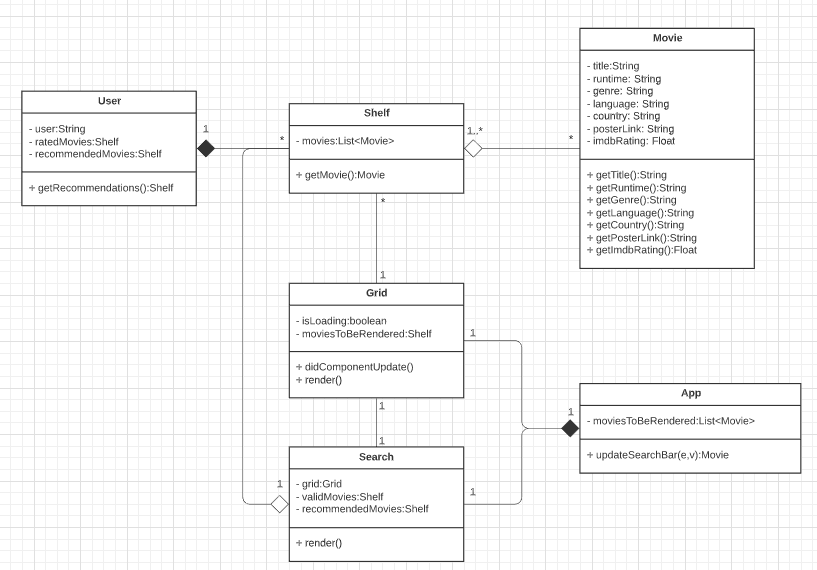
## **4.1 Use Case Diagrams**

The use case diagram is relatively simple, in that the only actors for the platform are registered and unregistered users. Registered users can carry out all actions on the platform, including: login and register using Google credentials, view previous ratings on user profile page, search for movies with an autocomplete textbox, navigate the website using a sidebar, click the lightbulb icon on the sidebar to generate a list of movies, add items to a watch list, rate movies from recommendation results, resize the window with a responsive grid, links to movies’ IMDb pages, and filtering the results by streaming platforms. Unregistered users can take fewer actions on the platform, but are still able to carry out the basic functions of registering, searching for movies, resizing the window, and navigating to IMDb page through the movie tiles.



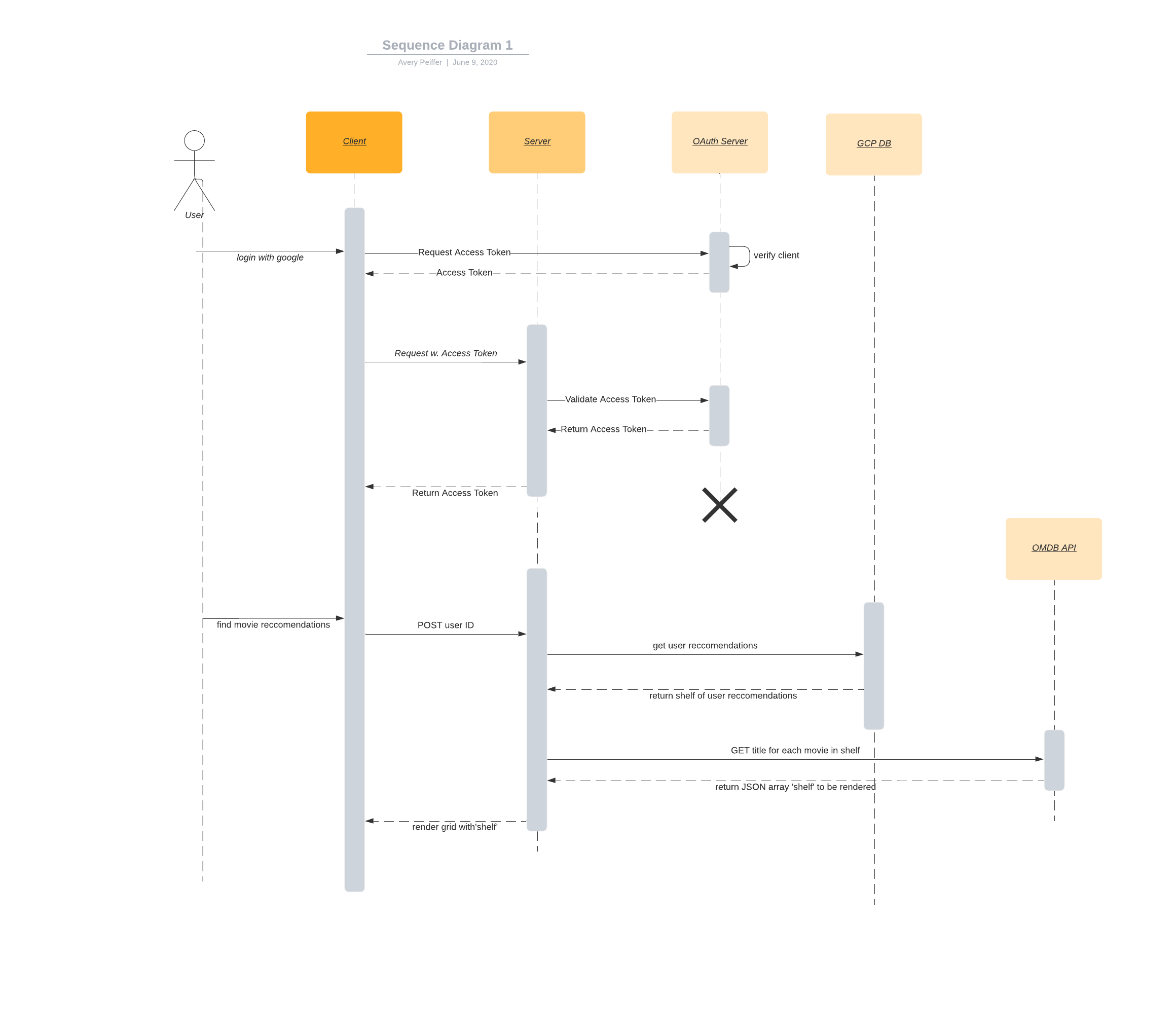
**4.2 Class Diagrams**

There are six main classes on the platform. The App class is the primary class that covers the entire platform. The Movie class contains general information about a title as well as getters for these attributes. An aggregation of movies makes up a Shelf, which is then used as part of the User class. The Shelves contain information about the movies a User has rated as well as the movies that User has been recommended. The Shelf is also used to make up the Search class by providing a list of valid Movies to return. The Grid class works with the Search class to display the Shelf of recommendations or rated movies.



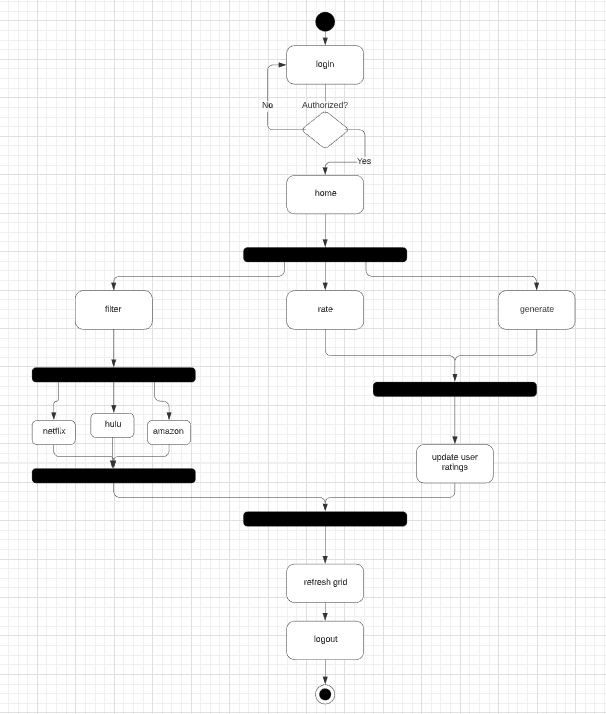
## **4.3 Sequence Diagrams**

The sequence diagram begins with a user trying to login. This will be accomplished using the OAuth Protocol, which is often seen as “continue with Google credentials” on many websites. First, a client requests an access token from the OAuth server, and if the Google username and password is verified by the OAuth server, the OAuth server then returns an access token. The user can then make requests with the granted access token to the server, and the server will validate that access token with the OAuth server and return the access token again. This is how authorization is performed. Further in time, once a user is verified, they can retrieve their recommendations. The client will post the user id to the server, where the server will then forward that to a database with a list of all the user’s preferences and movie recommendations. That list will then be returned to the server so that it can make a further GET request to the OMDb API in order to get the full details about a movie. The response from the OMDb API will then be returned as a JSON object to the server, and then ultimately back to the client so that it can render the grid and display user movie recommendations.



**4.4 Activity Diagrams (DFD)**

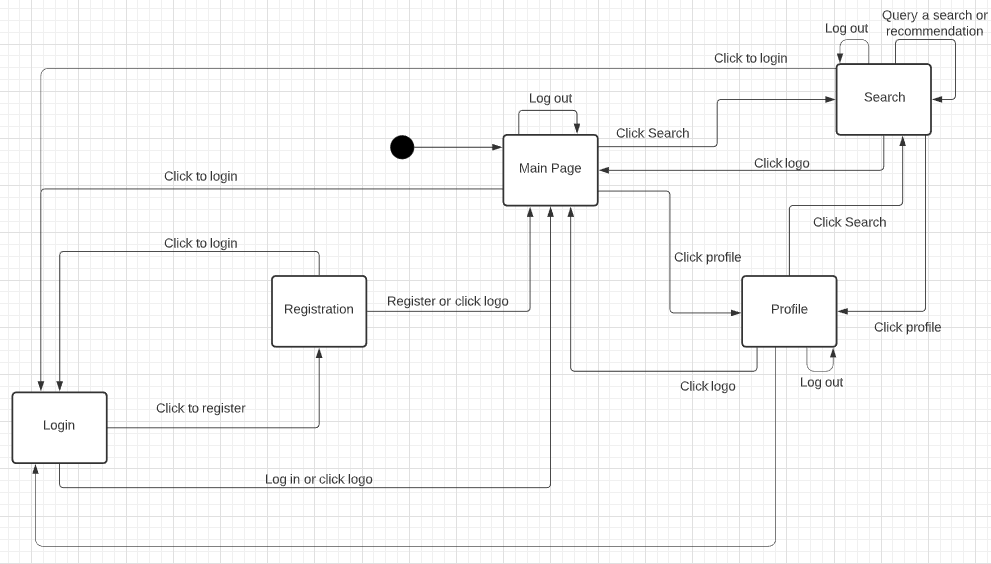
The activity diagram shows three main activities that a registered and authorized user can perform. They can either filter results, rate movies, or generate a list of movie recommendations based off of previous ratings. In the first case of filtering, they can have up to three options but may use more depending on how extensible the Utelly API is. To start out with they can choose Netflix, Amazon Prime, or Hulu. After they choose one option, it will refresh the grid of displayed movies. Similarly, after they rate a movie or generate new movie recommendations, it will update user preferences and their entries in the database, and then refresh the grid to reflect updated preferences.



## 

## **4.5 State-Transition Diagrams (STD)**

There are several states between which the software can transition. The starting state is the main page; from there, the user can click to navigate to their profile page, the login page, or the search bar. The main page can then be accessed from all other states by clicking the main logo for the platform. Notably, if the user tries to navigate to their profile or query a search or recommendation without being logged in, they will be prompted to do so before continuing.



# **5. Change Management Process**

Of course, everything in the project is subject to change. The SRS will be hosted and maintained on Google Docs for ease of collaboration and transparency. We will assume some division of labor within this project; for example, one person could work on the frontend, a second works on the backend, and the third and fourth work on the database. If the scope or requirements change that significantly affect a developer’s area of focus, then that developer will add a comment on the relevant portion of the SRS. The team lead will check in and be notified of any changes to the document, or it will be discussed in the daily stand up. The team lead will then discuss the comments further with the development team or approve the changes and allow the group to proceed. In this way, all members will be kept in the loop regarding any potential changes to the platform.