goodwatches

Final Project Documentation

Shawn Burnham, James Haywood, Justin Bushue

CS 321-02

Copyright 2022 Justin Bushue James Haywood Shawn Burnham
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3, (3 November 2008) or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license can be found at https://www.gnu.org/licenses/fdl.html.

Table of Contents

Table of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

1.0 Project Description

Finding a movie to watch can be a dilemma. People looking for entertainment know what interests them, but not necessarily which movies they may enjoy. The proposed system is a movie library tool, similar to book library tools such as Goodreads. The system will import a collection of available movies and provide the user the ability to search, sort, collect, review and save information to their personal collections.

2.0 Project Management

This is the description of the work that went into the project.

History

Over the course of the semester we have worked together to design and implement a system that allows the user to navigate and make use of a movie database. Up until about the middle of March we were only really doing the design portion of the project. There was some design implementation, but only things such as data classes like User or Movie and utility classes like AuthSystem. The back end portion of the implementation was done by the tail end of March and we really started implementing everything at the start of April. Implementation through April was fairly steady with progress being made in small increments. A lot of the features were completed around the same time (around our self imposed due date of 4/17).

Tentative timeline: Assigned Tasks ←(Our list of Action Items)

1. Back end and ability to login/create an account Complete (3/28)
2. Begin UI and functionality implementation (4/1)
3. Ability to create a custom MovieList (4/13)
4. Have AccountPage show users favorite movies & custom lists. (4/13)
5. Develop SearchPage to show a list of movies (around 4/15)
6. Developed MoviePage (4/16)
7. Ability to add movies to favorites or custom lists (4/17)
8. Implementation of the search bar and ability to search for a movie by different parameters (4/17).
9. Ability to write/read a review on the MoviePage (4/18)
10. Ability to view/edit user preferences (4/19)

Personnel

Justin Bushue: Senior CS major with extensive experience in programming via courses taken at multiple universities and through self-teaching. A student co-op at Siemens Digital Industry Software using Java, Python, Angular JS, Scala, and Perl in a professional environment. Mainly worked on the back end code and some UI.

James Haywood: Second year CS major who beyond the standard CS courses at UAH did some programming classes in high school. First time user of java and only experience with it is from this class. Mainly worked on UI front end implementation and functionality.

Shawn Burnham: Second year CS major at UAH. I took java in high school that I wasn't able to finish because of covid. No professional experience. Mainly worked on UI front end implementation and functionality.

Effort

Most of the time spent on the project was given towards the end during the implementation. The first three meetings (totaling about 5 hours) were all related to designing and brainstorming. After the third meeting Justin did a lot of work on the back end of the program. We then had two more meetings (2 hours 15 minutes) that were pretty spaced apart where we just discussed the current state of the program, specifics on how we want to implement certain things, and progress reports. The total amount of effort given to the project wasn't gigantically enormous and more effort definitely could have been given, but it also wasn't such low effort that we didn't try. Overall we put a healthy amount of effort into the project.

3.0 Use Cases

Actors - User using the program

UC1. User logs in or uses guest mode

Precondition - The program must be open and in focus.

a. If the user wants to log in they input their username and password.
    i. If the username exists and the password is correct, the user is logged in to their account.
    ii. If the username exists and the password is incorrect, the user is prompted to try again.
    iii. If the user wishes to log in, but lacks an account, the user can create an account using a unique username and a password.
b. If the user does not want to log in they may continue as a guest with certain features disabled (favorite/list creation).

UC2. User edits their movie preferences

Precondition - User must be logged in. The User cannot be in guest mode.

a. If the user wishes to set their preferences they can do so at account creation or by clicking the edit preferences button on AccountPage
    i. The user can select their favorite genres from a list of genres.
    ii. The user can select their favorite actors from a list of actors.
    iii. The user can select their favorite directors from a list of directors.
b. If the user wishes to skip the preference set up process they can press done until they reach AccountPage
    i. The user can edit their preferences later through the button on AccountPage

UC3. User can select a movie to favorite/unfavorite

Precondition - User must be logged in. The user cannot be in guest mode.

a. User opens/selects a movie page and selects the favorite button. The item is then added to the user's favorite list.
b. The user can unfavorite a movie through the same process as UC3.a except the favorite button will read unfavorite.

UC4. User can search for a movie

Precondition - User must be logged in or in guest mode.

a. The user can search for a movie by inputting their search query into the search by. The search can be done by multiple parameters which can be changed by toggling a button.
    i. By default the search will look for any movie that has the search query mentioned in any of its fields.
    ii. The user can search by title to search for a movie by its title.
    iii. The user can search by actor to search for a movie by its actor.
    iv. The user can search by director to search for a movie by its director.
b. If the search field is empty then the default list of movies is displayed.
c. If the search yields no results then "NO RESULTS" is displayed.

UC5. User can create/delete a custom list of movies

Precondition - User must be logged in. The user cannot be in guest mode.

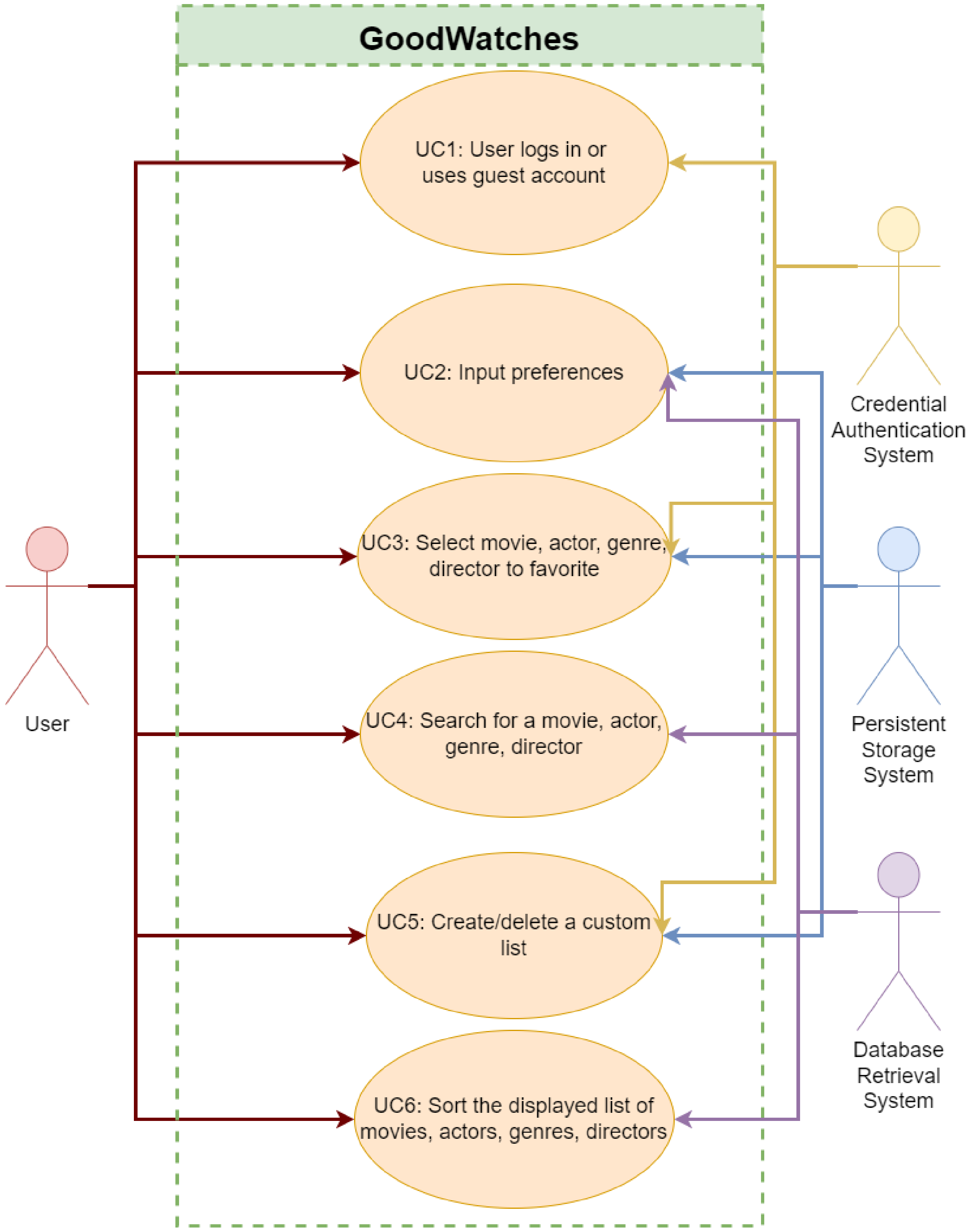
a. The user can create a list of movies similar to a song playlist where they can add any movie in the database.
b. The user can delete a custom list of movies by selecting a list of movies and selecting delete.
    i. PRECONDITION - User must have a list already created
c. The user can add/remove movies to a custom list after initial creation.
    i. This is done through opening a movies MoviePage.

UC6. User can get a movie recommendation based on their preferences

Precondition - User can be either logged in or a guest.

a. The user can click a button on the AccountPage that will select a random movie and open its MoviePage.
    i. This random movie will match the users preferences.
    ii. If the user has no preferences then the movie will be completely random.
    iii. If the user is a guest then the movie will be random because guests don't have preferences.

3.1 Use Case Diagram


GoodWatches
UC1: User logs in or uses guest account
UC2: Input preferences
UC3: Select movie, actor, genre, director to favorite
UC4: Search for a movie, actor, genre, director
UC5: Create/delete a custom list
UC6: Sort the displayed list of movies, actors, genres, directors
User
Credential Authentication System
Persistent Storage System
Database Retrieval System

4.0 Requirements

Functional Requirements

R1. Create an account

a. The user of the program can create a unique account with a password and username. The created account will generate a favorite movies, actor, and genre list that the user can fill with favorite movies, actors, and genres. This information will be saved so the user can later log into their account and continue where they left off.

R2. Log into an account

a. The user of the program will be able to input a unique username and password in order to log into their unique account. This account will hold all past information from prior program use such as favorite movies, actors, genres, and custom lists.

R3. Log out of account

a. The user of the program can logout of the account saving any changes made to their account and returning them to the MainPage.

R4. View information about a movie

a. The user will be able to select a movie and view information about the selected movie such as description, genre, director, actors, run-time, ratings. While viewing information about the movie the user can add the movie to their favorite/custom list.

R5. Search for a movie by a parameter

a. The user will be able to search for a movie using various parameters such as title, genre, directors, and actors. The results will be displayed in a list that the user can then select a movie from.

R6. Add a movie to user's favorite list

a. The user when viewing a movie can add the movie to their favorites list. User's favorites list is automatically created when creating an account.

R7. Delete a movie from user's favorite list

a. The user can select a movie in their favorites list and remove said movie from their favorites list.

R8. Create a custom movie list

a. The user may create a custom list that functions almost identically to the favorites list but is separate. A custom list will have a unique name.

R9. Delete a custom movie list

a. The user will be able to delete one of their unique custom made lists. Deleting a list will remove it from the user's account and completely delete it.

R10. Update a custom movie list

a. The user will be able to add/remove movies from the custom list as they please similarly to the favorites list.

R11. Add/remove actors to a user's favorite actors list

a. The user will be able to add/remove an actor from a user's unique list of favorite actors.

R12. Add/remove genre to a user's favorite genre list

a. The user will be able to add/remove a genre from a user's unique list of favorite genres.

R13. Add/remove director to a user's favorite directors list

a. The user will be able to add/remove a director from a user's unique list of favorite directors.

R14. Load/Save Data

a. On startup the program will find the file directory path to where user and movie cache data is stored. If the computer running the program is missing the necessary files it will create them. The program will save/update these files on user logout or program close.

User Interface Requirements

UI1. Account login/creation page

a. The login page will have two fields labeled username and password to allow the user to input their login information. Once the information is inputted they can select a button labeled login to complete the process. Additionally there will be another button labeled create an account which will bring the user to a similar page where it asks the user to input a username, password, and password confirmation. A button on this account creation page will confirm the account creation and once the account is successfully created take the user to the program home page. There will be another button labeled continue as a guest that will allow the user to use the program without logging in.

- If the username the user is attempting to use to create an account is already in use it will tell the user the username is taken.

UI2. LoginPage

a. The LoginPage will have two fields labeled username and password to allow the user to input their login information. Once the information is inputted they can select a button labeled login to complete the process.

- If the username and password are accepted by the AuthSystem then the user is logged into an account and taken to the AccountPage.

        - If the username/password the user inputted is incorrect then it will prompt the user to try again.

UI3. CreateAccountPage

a. The CreateAccountPage is where the user can create an account. It has multiple fields such as first name, last name, an about me text area, username, password, and confirm password. After correctly inputting necessary fields it will create the account.
    - The only fields necessary for creating an account are the unique username, password, and confirm password.
    - Passwords must be at least eight characters and contain only letters or numbers.
    - If passwords don't match or username is taken it will prompt the user to fix the issue.

UI4. PreferencePage

a. The PreferencePage is where the user can add/edit/remove their preferences. The user will be able to set favorite genres, actors, and directors in this page as well as remove them.

UI5. AccountPage

a. The AccountPage will contain information about the user such as their name, a logout button, their custom lists, and their favorite movies. User will be able to open their custom lists, navigate to the SearchPage, and navigate to the PreferencePage.

UI6. MoviePage

a. The movie info page will display information about the movie selected from the list. This information includes the movie's description, actors, directors, runtime, ratings, reviews, etc. There will be buttons on this page that allow the user to add/remove this movie from their favorite/custom list. Additionally this page will have a list of reviews that the user can view and create their own.

UI7. SearchPage

a. The search page will display a movie list and a search bar that the user can use to filter the movie list. The search bar will have a field to enter text and ways to set the search type (title, actor, director).
    - The displayed movie list should be able to be changed so when opening a viewing a custom movie list it can just display that custom movie list and not the movie database.
        - If viewing a custom list there should be a button to delete the list.

Future modification and extensions

Some future modifications could be made such as for the user to have the ability to periodically save as well as how the search works. Currently the program saves whenever the user logs out or closes the program, but we had wanted to include a save button that would let the user possibly discard changes they didn't mean to make. It currently does not work as intended as logging out wont discard unsaved changes. Furthermore, adding in the ability to have the MoviePage background color to be saved into the movie object so that the color is not being calculated every time each movie is being opened. Another modification would be to have a smoother/cleaner UI overall. As it stands the UI is rather utilitarian, although the program is in fact a utility, the overall design should cater to the users that would use such a tool. Having a more modern look would likely appeal greater to the masses.

Summary

1. The user should be able to use the program as a guest to quickly search for a movie or be able to login to an account to have access to more features as presented in Use Case UC1.
2. The user should be able to create an account that will let them access more features such as favorite, custom lists, etc than if they were a guest as presented in UC1.a.iii.
3. A user that is logged into an account should be able to add or remove movies to their favorites list as presented in UC3.
4. A user that is logged into an account should be able to create a custom movie list in which they can add/remove movies from it as presented in UC5. Furthermore they should be able to delete this custom list whenever they want.
5. A user that is logged into an account should be able to edit their preferences of favorite actors, directors, genres as seen in UC2.
6. The program shall allow the user, regardless of account status, the ability to search for a movie with a string input. Furthermore the user should be able to change the search condition to specific things such as title or genre. Presented in UC4.

Associated tests

A series of unit tests will be conducted on each function that adds, changes, or removes data from any structure. These tests will be done to ensure that the functions are behaving as expected. There will also be a large amount of manual testing done for the UI and its integration with the back end.

5.0 Design

Model design

What is in the model (the backend) and what are the ways in which it can change?

1. *Users*
    a. The user data is data that is specific to a unique user. For example the username and password, a user's list of favorite movies, a user's custom lists, user's settings, etc. This model can change whenever the user makes changes such as adding/removing a favorite movie, editing a custom list, changing profile settings, etc.
2. *Movies*
    a. The movie data is data that is specific to a unique movie. This data includes things such as title, description, rating, runtime, release date, actors, directors, genre, etc. The only piece of data in this model that changes is the review data where user can add a review to the movie. Otherwise everything else remains static.
3. *MovieLists*
    a. MovieList data is basically just a list of movies. The list can be something like the favorite list that each user has or a custom made list that has a unique name. Only significant way this model changes is the contents of the list (movies in the list).
4. *Movie Database*
    a. The Movie Database, otherwise known as the master movie list, is the movie list obtained from the .json file. The only way the Movie Database changes is if the file actual json file is swapped out.

View design

What are the views (the frontend) of the data and how are they generated?

1. *Displayed movie list*
    a. This view is just the MovieList displayed as a list to the user. The actual movies displayed can be changed by the search parameters.
2. *MoviePage*
    a. The movie page is displayed when the user selects a movie. It displays the data about the selected movie. (see model design 2.a)
3. SearchPage
    a. The SearchPage is where the user views a movie list. It's generated by taking a MovieList and displaying each movie in the MovieList.

4. AccountPage
    a. The AccountPage is where the user view can see their favorite movies and custom movie lists. It gets this data from the User class.

Control design

What are the elements by which the user can request a change to the model (the flow of the program)?

1. *Login page*
    a. The login page is where the user can enter a unique username and corresponding password to access their user data.
2. *AccountPage*
    a. The AccountPage is where the user can create a custom movie list.
3. *MoviePage*
    a. The MoviePage is where the user can add/remove a movie to their favorite movies. They can also add/remove a movie to a custom movie list from the MoviePage.
4. *Search parameters*
    a. Search parameters are a way to change the contents of the displayed movie list. For example, users can search by movie title and the displayed movie list would then only show movies of matching titles.
5. *PreferencePage*
    a. The preference page is where the user can edit their preferences.

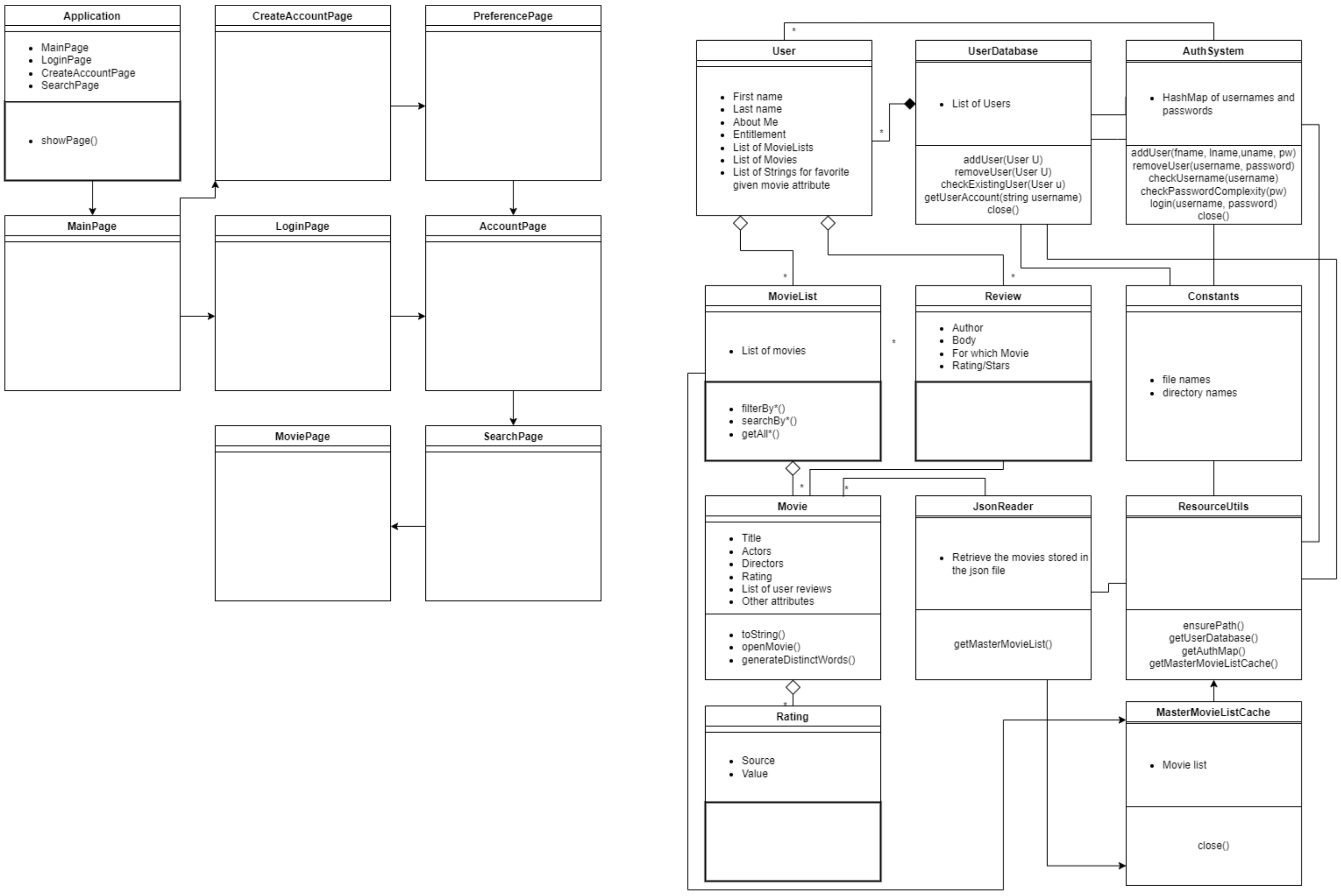


Figure 2 - Class Diagram

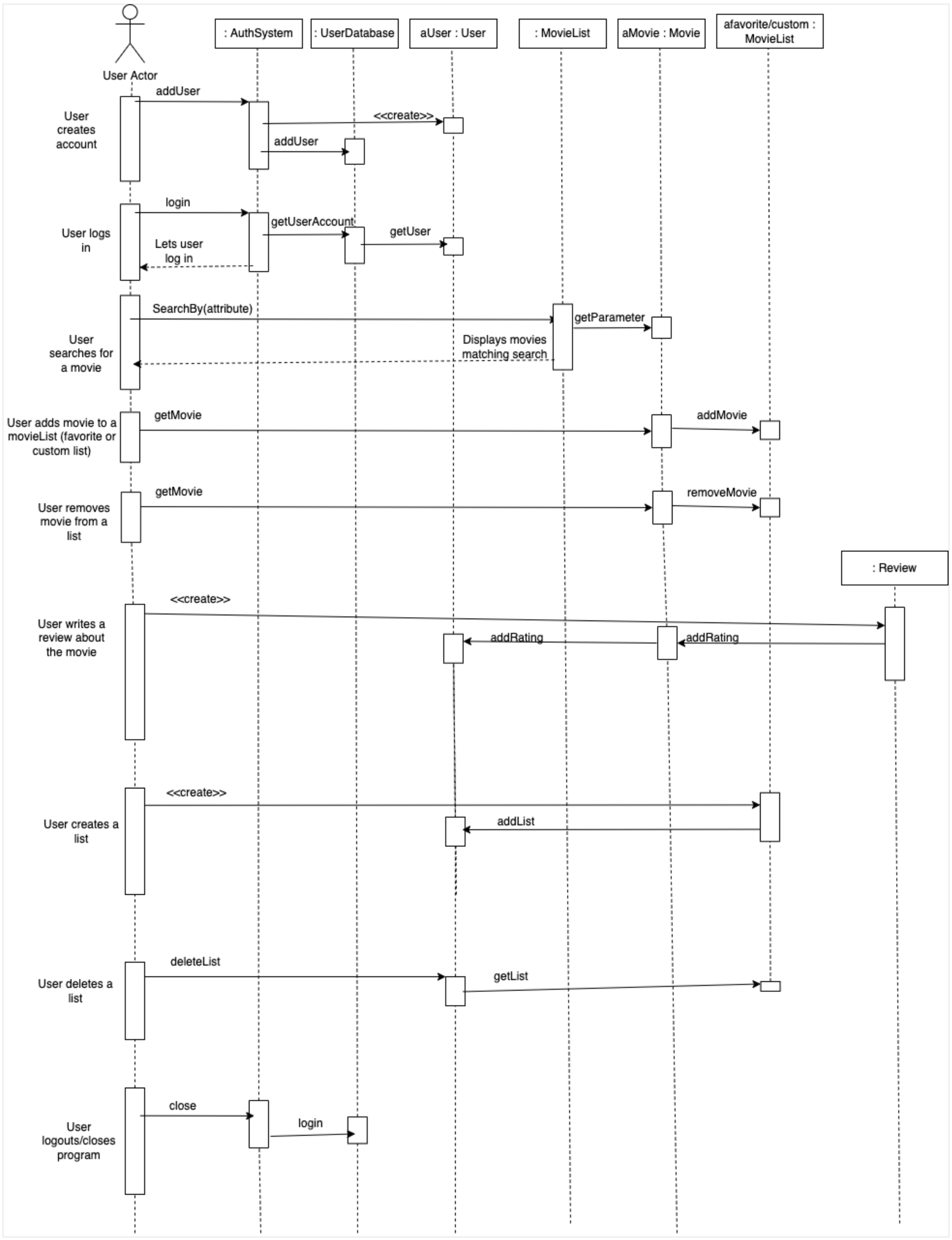


Figure 3 - Sequence Diagram

6.0 Implementation

Packages and classes

- P1. Backend
  - C1. Movie: The Movie class is a concrete object class that satisfies the model design 2 (Movies). It implements Serializable. It contains information about a movie such as title, plot, genre, etc.
  - C2. MovieList: The MovieList class is a concrete object class that satisfies the model design 3 (MovieLists). It implements Serializable. MovieList contains a list of Movie objects and is also used to satisfy the model design 4 (Movie Database).
  - C3. Review: The Review class is a concrete object class that implements serializable and is used to write reviews. It contains the author of the review, movie, rating, and text of the review.
  - C4. User: The User review class is a concrete object class that implements Serializable and is used to store user data. It contains user data such as username, reviews, custom lists, favorites, and preferences. It satisfies the model design 1 (Users).
  - C5. UserDatabase: The UserDatabase class is a concrete object class that contains the list of all users.
- P2. Frontend
  - C1. Application: Application is a concrete object class that extends JFrame. It is basically the main window that all other pages and functions are accessed through. It uses a composite design that combines multiple other pages to allow for quick access/swapping.
  - C2. MainPage: MainPage is a concrete object class that extends JPanel. It is the page that the user starts on after starting the program. It contains buttons that bring the user to different pages such as Loginpage, guest AccountPage, or CreateAccountPage. It uses a composite design where everything that is in the JPanel MainPage can be added to a window like Application.
  - C3. CreateAccountPage: Concrete object class that extends JPanel. It is the page where the user can create an account that is saved into the UserDatabase. It contains some text fields where the user can input their account details and a save button. It uses a composite design where everything that is in the JPanel CreateAccountPage can be added to a window like Application.
  - C4. LoginPage: Loginpage is a concrete object class that extends JPanel. It is the page where the user can enter their account details,

username and password, into some text fields and login. Satisfies control design 1. It uses a composite design where everything that is in the JPanel Login can be added to a window like Application.

C5. AccountPage: AccountPage is a concrete object class that extends JPanel that mostly displays information from the User class such as their favorite movies, custom lists, username, and about me. It satisfies view design 4 and control design 2. Additionally it contains buttons that help the user navigate the program such as bringing them to the SearchPage, PreferencePage, or logging them out. There's also a button that allows use of a recommended movie function. It uses a composite design where everything that is in the JPanel AccountPage can be added to a window like Application.

C6. PreferencePage: The PreferencePage is a concrete object class that extends JPanel. It satisfies control design 5 and is where the User can edit their preferences. It can be accessed either after account creation or a button on AccountPage. Furthermore there is a button in the PreferencePage that resets the users preferences. It uses a composite design where everything that is in the JPanel PreferencePage can be added to a window like Application.

C7. MoviePage: The MoviePage is a concrete object class that extends JPanel. It satisfies view design 2 and control design 3. The class is where the user can view the data from the Movie Class and is used whenever the User clicks on a movie to view it. It is also where the User can add the specific movie to their favorites or a custom list. Furthermore the User can write and read reviews for a specific movie. It uses a composite design where everything that is in the JPanel MoviePage is added to a pop-up JDialog.

C8. SearchPage: SearchPage is a concrete object class that extends JPanel. It satisfies view design 3 and control design 4. This page is where the user can view the movies in a MovieList and then filter the list with different search parameters. The main SearchPage that is navigated to from the “Search for a movie” button in AccountPage is added to the JFrame Application. The SearchPage is also used when the User is viewing a custom MovieList. When viewing a custom MovieList it will open a pop-up JDialog containing the SearchPage for the specified list. When viewing a custom MovieList there is a button on the page that allows the user to delete the custom list.

Utility classes and packages

PU1. Backend

CU1. AuthSystem: The AuthSystem class is a class that contains a HashMap of usernames and passwords. It also contains a method to add a user to the HashMap, a method to remove a user from the HashMap, a method to check if a username is in the HashMap, a method to check if a password is correct for a given username, and a method to log in a user.

CU2. Constants: The Constants class is an abstract class that just holds constants such as jsonFileName and windows/mac file directory path ways. It also holds the names of the files containing the master movie list cache, username/password list, and list of all users.

CU3. JsonReader: The JsonReader class is the class that is responsible for getting information out of the .json file containing the movie database.

CU4. MasterMovieListCache: The MasterMovieListCache class is an abstract class that reads and writes a MasterMovieListCache file.

CU5. ResourceUtils: The ResourceUtils class is an abstract class that handles the program resources. Primarily ensuring the file directory paths are correct, creating the pathways if they don't exist, getting the UserDatabase, etc.

CU6. Rating: The Rating class is a concrete object class that implements serializable.

PU2. Frontend

CU1. SimpleDialog: SimpleDialog is a concrete object class that extends JDialog. It is simply a pop-up dialog window that displays a message to the user. It's used for error messages.

Tested functionality

Most of the back end functions had been tested using unit testing. Since nearly all functionality barring the direct actions performed by UI elements is done in the back end, doing unit testing was the easiest approach to ensure the functionality expected was the functionality received. The specific tests ran are linked here. This pdf also includes console output from the tested functions, how long the test took to run, and the pass/fail status. Another useful tool we utilized was the code coverage report. This was able to show us what functionality was being executed during a specific execution order of the program. This interactive report can be viewed by checking out the github repository and opening "/Documentation/coverage report/index.html". For the UI, we did a substantial amount of manual testing by going through specific scenarios. These scenarios were not documented, but led us to find and fix several bugs that we otherwise would have not been privy to.

Untested functionality

All functionality was tested via unit testing or manual testing.

7.0 Discussion

James Haywood: One trade-off/issue that I have with the current implementation is on how the user views the MoviePage and SearchPage when viewing a custom list. I am the one that really implemented the way these pages open up and I kind of regret having them open in a JDialog instead of just adding them to the Application. It wouldn't be that hard to change because those classes are just JPanels, just time consuming and by the time I had the realization the project was nearly complete. If I restarted this project in regards to JUST the UI implementation, I feel like I could do it a lot more effectively and cleanly. This project definitely taught me a lot about java and especially GUIs.

Justin Bushue: The idea behind the way the persistent storage is implemented is to disallow users from making changes to the user accounts, or authentication system outside of the program. Although more measures could have been taken to ensure greater security, the serialization to binary seemed a good place to start. The inclusion of the MasterMovieListCache was due to the realization that reviews wouldn't get saved to a specific movie but rather only to the user that authored them. Since we were not planning to display the reviews in the account page we needed a way to easily add the reviews to the master movie list without having to manipulate the original JSON file. Having a configuration file to be able to specify the JSON movie list file would have been a better approach but the idea of doing this came too late in the implementation that would have been infeasible to get it implemented in the remaining amount of time. The implementation of the search by any, the hairy search if you will, was very entertaining to work on although the actual implementation was different than the original concept. The original idea was to convert all the distinct words in the movie's title, plot, actor names, director name, genre, etc to a numerical code and store the code in a binary tree. What ended up getting implemented was just storing the distinct words in an arraylist. The trade-off was using more memory and more execution time but I didn't think to use a TreeSet. The execution time forArrayList.contains() is O(n) whereas TreeSet.contains() is O(log n)[1]. The computation time for the numerical codes would negatively affect the overall execution time as well. A lesson learned, for likely all of us, is to use all of the time allotted to us for each phase of the project. There was a lot of procrastination involved in the making of this program but we were able to achieve a fairly well-rounded program with the time we allotted to ourselves.

Shawn Burnham: My favorite implementation feature of the project was the ability to see other people's reviews, which isn't that hard to implement, but makes it look more professional. A lesson I learned was how important it is to communicate with your teammates on what you're working on and assigning tasks so that your work doesn't overlap and break things.

[1] As per Java Generics and Collections by Maurice Naftalin, Philip Wadler.ISBN: 9780596527754

Appendix A

CRC Cards

Movie

Responsibilities

- Holds all information relating to a specific movie

Collaborators

- Rating

MovieList

Responsibilities

- Used to hold the Master movie list and can be used for the users custom movie lists
- Allows for filtering the master movie list and custom movie lists based on different movie attributes
- Allows for searching the master movie list and custom movie lists based on different movie attributes
- Allows for retrieving of all distinct values of a given movie attribute

Collaborators

- Movie

User

Responsibilities

- Manage about me
- Manage favorite movies
- Manage favorite actors
- Manage favorite genres
- Manga favorite directors

Collaborators

- None

Rating

Responsibilities

- Holds the Source and Value stored in the JSON array element Rating

Collaborators

- None

JsonReader

Responsibilities

- Parsing the json file stored in the resources folder of the project

Collaborators

- ResourceUtils
- Movie

UserDatabase

Responsibilities

- Holds a list of all registered users
- On open of the program, the list of registered users is read from a binary file
- On close of the program, the list of registered users is written to a binary file
- Adding of Users
- Removal of Users
- Retrieval of a specific user account

Collaborators

- User
- AuthSystem

AuthSystem

Responsibilities

- Allows the user to create a user
- Allows the user to remove a user
- Verifies the user exists
- Checks password complexity
- On open of the program, the list of registered usernames and passwords is read from a binary file
- On close of the program, the list of registered usernames and password is written to a binary file

Collaborators

- User
- UserDatabase

ResourceUtils

Responsibilities

- Collection of methods that handle retrieving the file(s) stored in the resource folder i.e. json file
- On initialization, handles the determination of if a user is using Windows or macOS for selecting the proper location to store the AuthSystem and UserDatabase outputs.

Collaborators

- Constants

Constants

Responsibilities

- Holds the various string and other constant values for use in other places of the program

Collaborators

- None

LoginPage

Responsibilities

- Getting the username and password from the user
- Check if the username and password combination is valid
- Allows for creating a new user
- Allows for continuing as a guest account
- Retrieves the user account and passes it onto the next

Collaborators

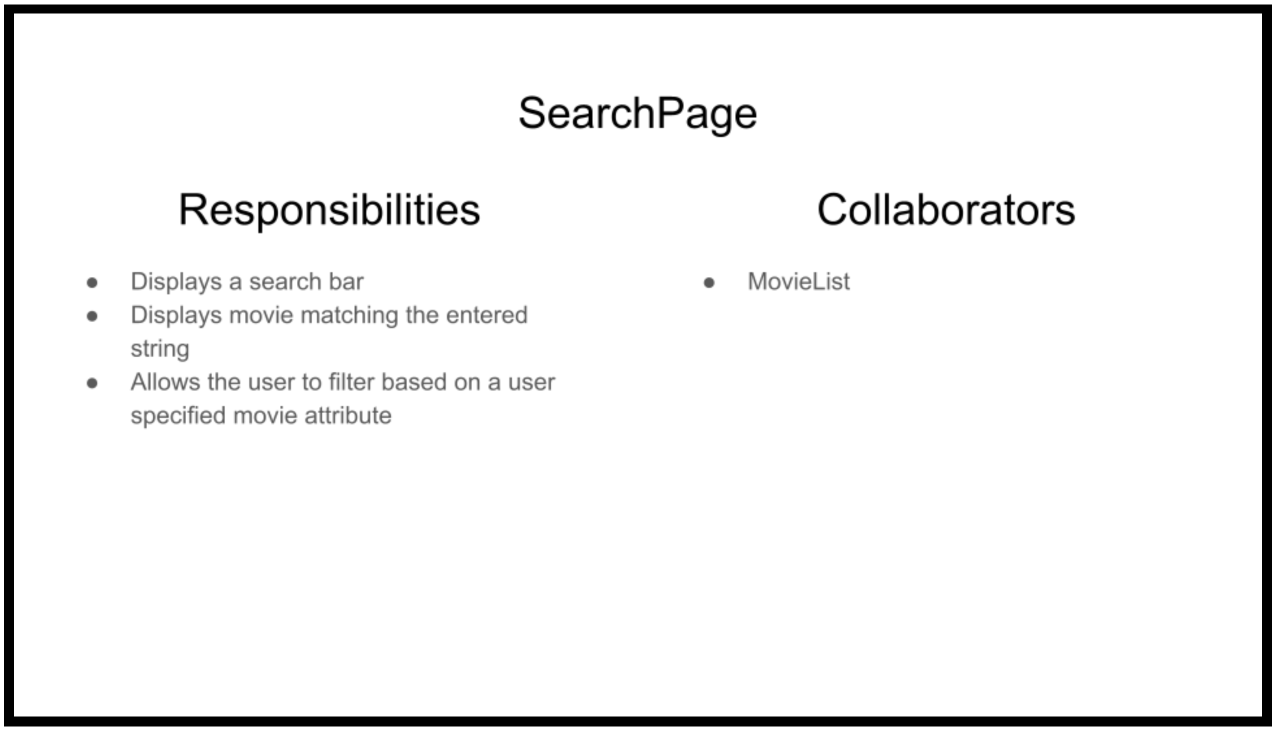
- AuthSystem
- UserDatabase

PreferenceSelectionPage

Responsibilities	Collaborators
• Displays all of a given movie attribute in the master movie list as a selectable list of buttons • Saves preferences into corresponding user favorites lists	• User • MovieList

MoviePage

Responsibilities	Collaborators
• Displays all pertinent information about a movie	• Movie



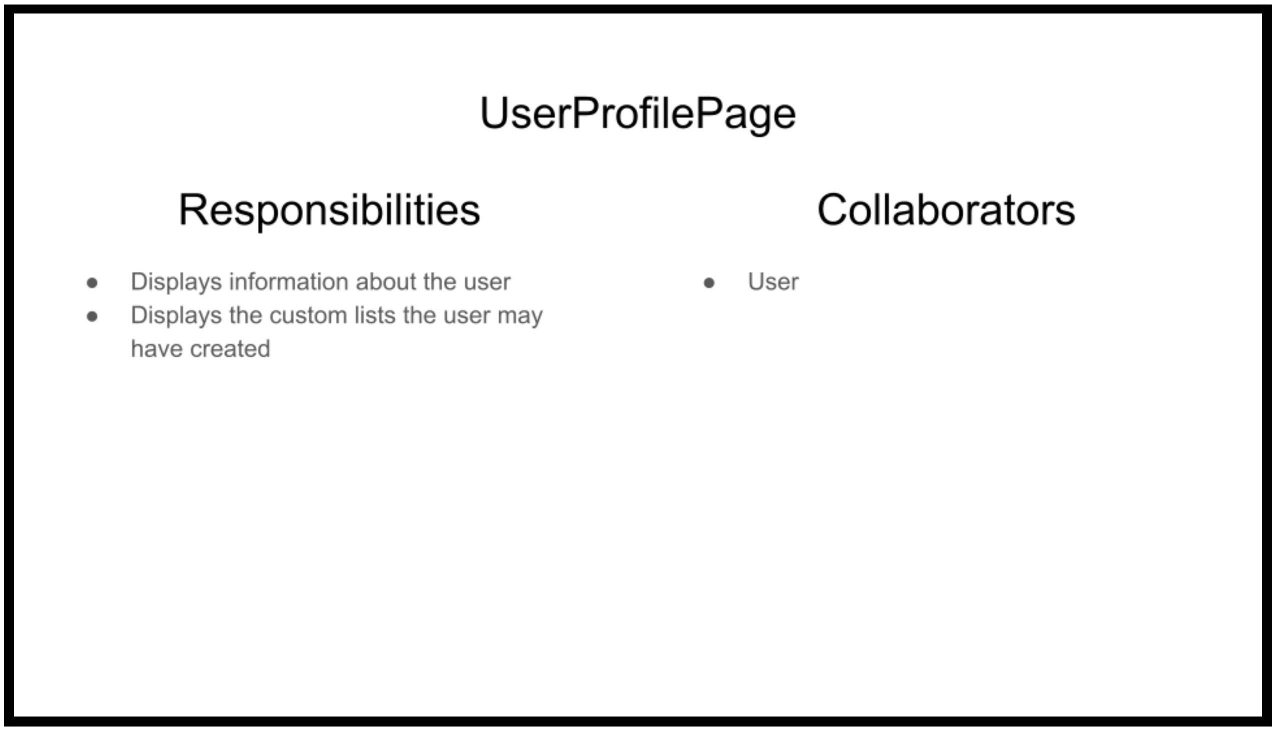
SearchPage

Responsibilities

- Displays a search bar
- Displays movie matching the entered string
- Allows the user to filter based on a user specified movie attribute

Collaborators

- MovieList



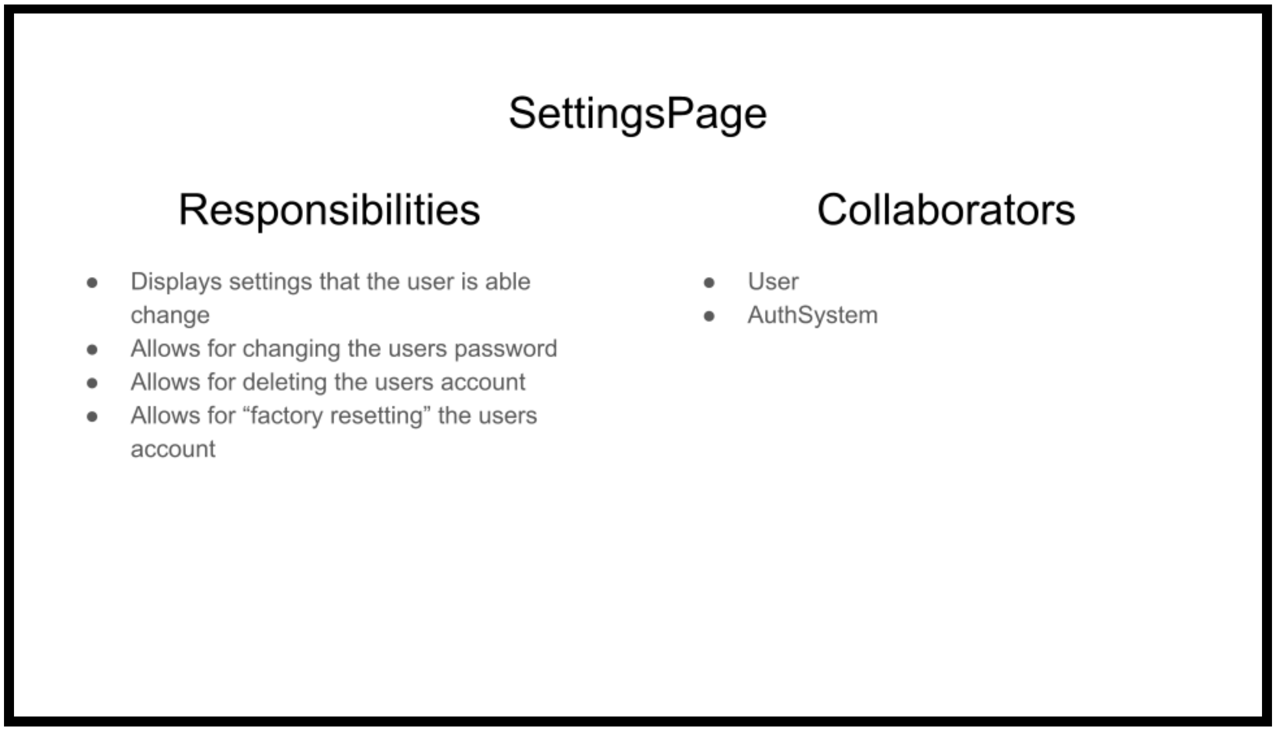
UserProfilePage

Responsibilities

- Displays information about the user
- Displays the custom lists the user may have created

Collaborators

- User



SettingsPage

Responsibilities

- Displays settings that the user is able change
- Allows for changing the users password
- Allows for deleting the users account
- Allows for “factory resetting” the users account

Collaborators

- User
- AuthSystem