Flick Finder Final Report

https://github.com/trallen22/flick-finder

Group 1: Tristan Allen, Daniel Carter, Will Cox, Josiah Jackson

April 30th, 2024

1. Introduction

Our project is titled FlickFinder. The idea for FlickFinder stemmed from the issue of browsing time before watching a movie. Many people spend countless minutes debating which movie to pick, because they are unsure which movie they might enjoy the most. The issue of finding a movie that everyone might enjoy is only compounded with larger groups of friends. FlickFinder aims to fix this issue by using a content based machine learning recommendation algorithm to personalize recommendations for each user and provide a concise list of movies tailored to their likes. With our system we hope to streamline movie selection times to save our customers time that they can then spent actually watching a movie. Our customers for this project are our friends and roommates at Armfield Courtyard, as well as any movie enthusiast. For the development of our system, we chose to follow the Agile-Scrum framework, due to our past experiences with this method.

2. Novelty

The novelty of our system is based on its use of the content-based machine learning algorithm to succinctly offer a variety of movies to a user based on their prior preferences. We designed an application that would help a user quickly find a movie that they would enjoy based on their interactions with other movies. We differ from popular streaming services in that our main focus is not to give users a platform to watch movies,

rather we aimed to design a system that could help users find an enjoyable movie without wasting time. Similarly, we tried to go beyond the capabilities of websites such as IMDb and Letterbox, where they're main focus is to allow users to curate a list of movies that they may like, and actually provide movie recommendations to assist in choosing an enjoyable movie.

3. **Customer**

Primary, Secondary, Other Stakeholders

The primary stakeholders outside of our team were Tristan and Will's roommates, as well as friends that live in Armfield Courtyard. Secondary stakeholders would be movie enthusiasts. Our stakeholders want a system that will give personalized recommendations in hopes that it will decrease the time that it takes to choose a movie that the user would enjoy..

Problems and Requirements

Meeting Log:

Date	Description of Customer	Description
2/5	Roommate A	Want to be able to have a concise list of movies. They're overwhelmed by the amount of options displayed by common streaming services. Wants to be able to have a list of movies that they have previously seen that they can look at when they are recommending movies to others

2/6	Roommate A	Wanted to get movie recommendations for a specific genre. Need to be able to filter movies by genre.
2/13	Roommate B	Customers want to set their favorite movies. Need to have a favorite button for movies.
3/4	Friend A	Wanted to rate a previously seen movie. Need to add a search function to be able to like previous movies.
3/15	Friend A	When shown the current implementation of the system they felt that the profile needed to include movies that were interacted with.

Product Backlog:

- Be able to filter movies by genre
- Want to be able to rate, like, and favorite movies that users have previously watched
- Have a profile page where they can see all of the movies they have previously seen and interacted with
- Have a friendly home screen where they can browse movies
- Want to be able to “archive” movies. Make it possible to be able to see their past enjoyment of movies if they want to.
- Want to be able reset a forgotten password to avoid having to create a new account

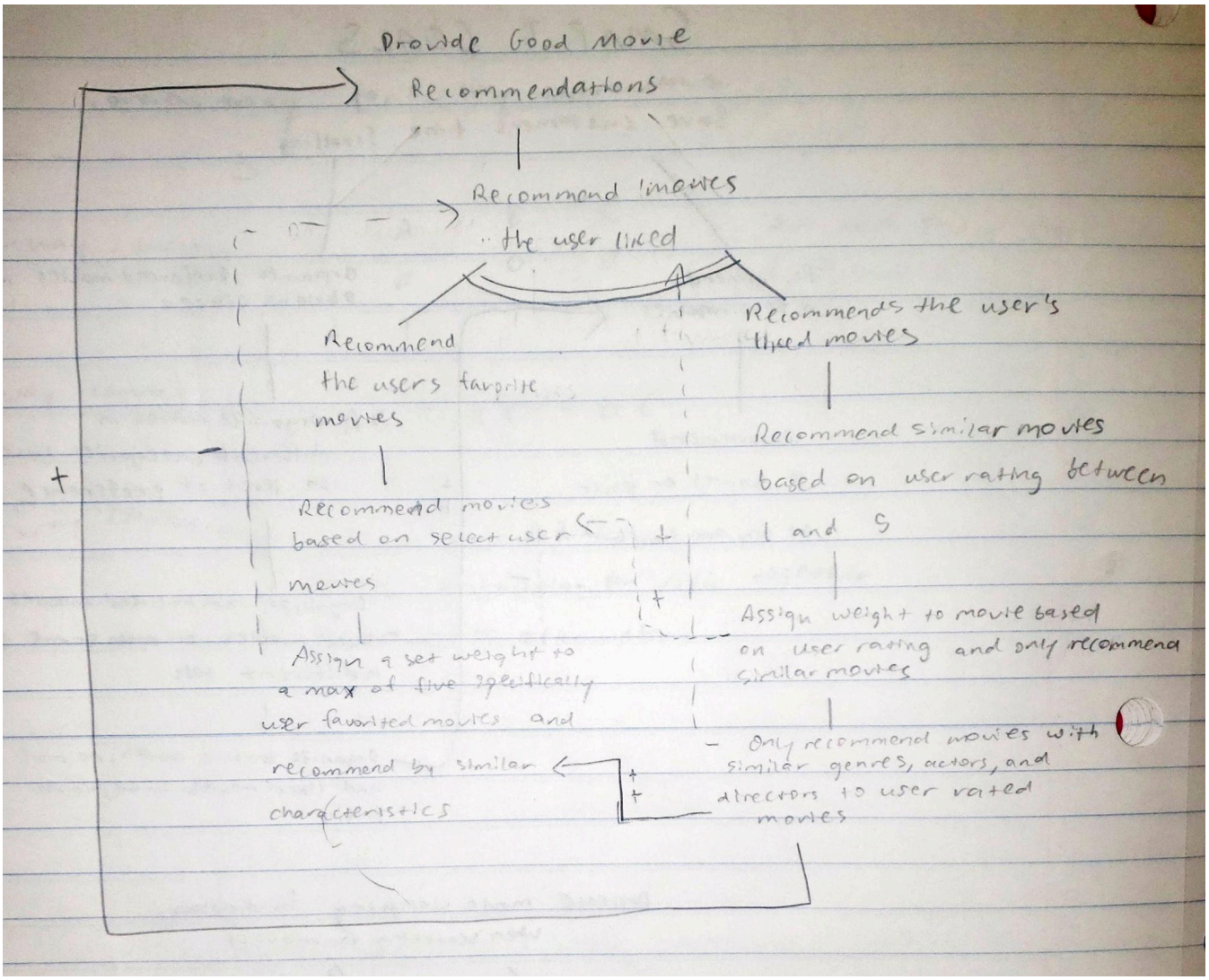
- Want to have age restrictions to prevent inappropriate movies from being recommended to younger users
- Would like a search feature so that a user can find specific movies

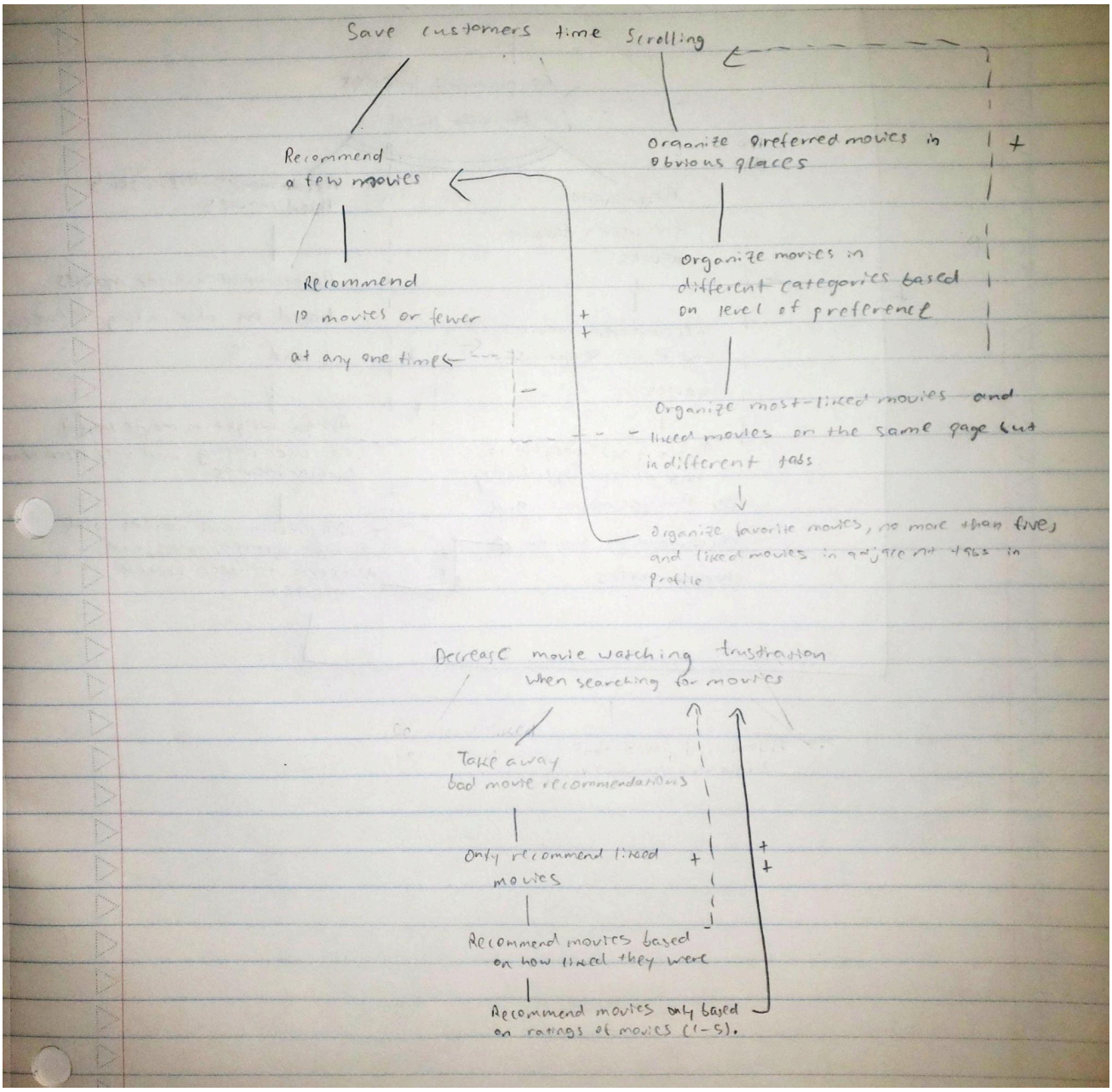
Overall Customer Experience:

Our desired customer experience is dependent on three things: movie recommendations, movie archiving, and movie interactions. For recommending movies, we want users to be able to find personalized recommendations that are easy to digest. We don't want the customer to feel overwhelmed by the amount of options that are supposed to be "for them". By giving shorter lists of recommendations, we feel that this experience is satisfying. Something our customer also emphasized was being able to interact with movies (rate, like, favorite). This combined with the ability to see past movies that have been interacted with on a user's profile page gives our user easy access to the movies they have watched and the ability to see their level of enjoyment at a glance.

Judging the current implementation of our system, we feel that we have successfully delivered the desired experience to our customers. Of course there are things that can continue to be developed, such as refining the user interface, but in the context of our ten week deadline we have delivered a functional system that meets the desired experience of our customer. We have implemented a page to display a short list of recommended movies, along with a user profile page that shows movies that have been interacted with by user.

4. SMART Goals





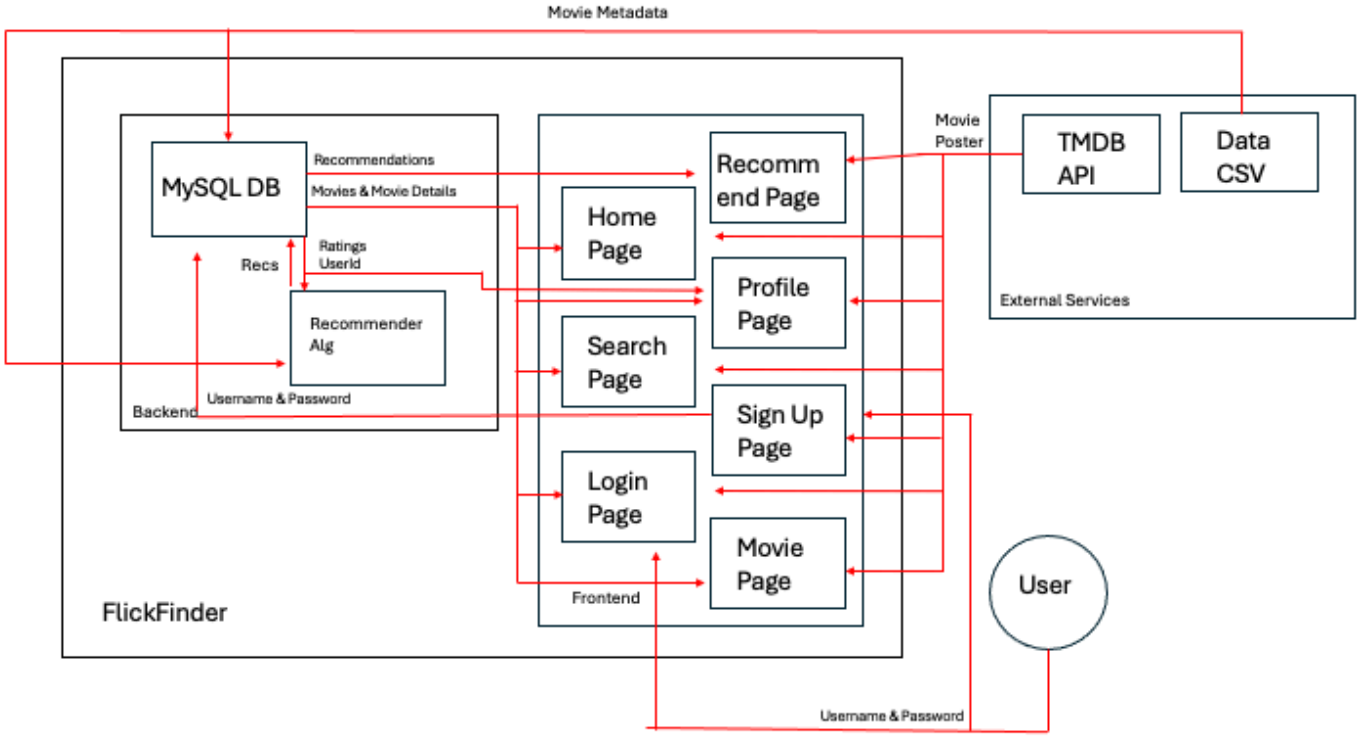
5. Sprint Backlog

- Implementing database
    a. Data importer
    b. Database schema

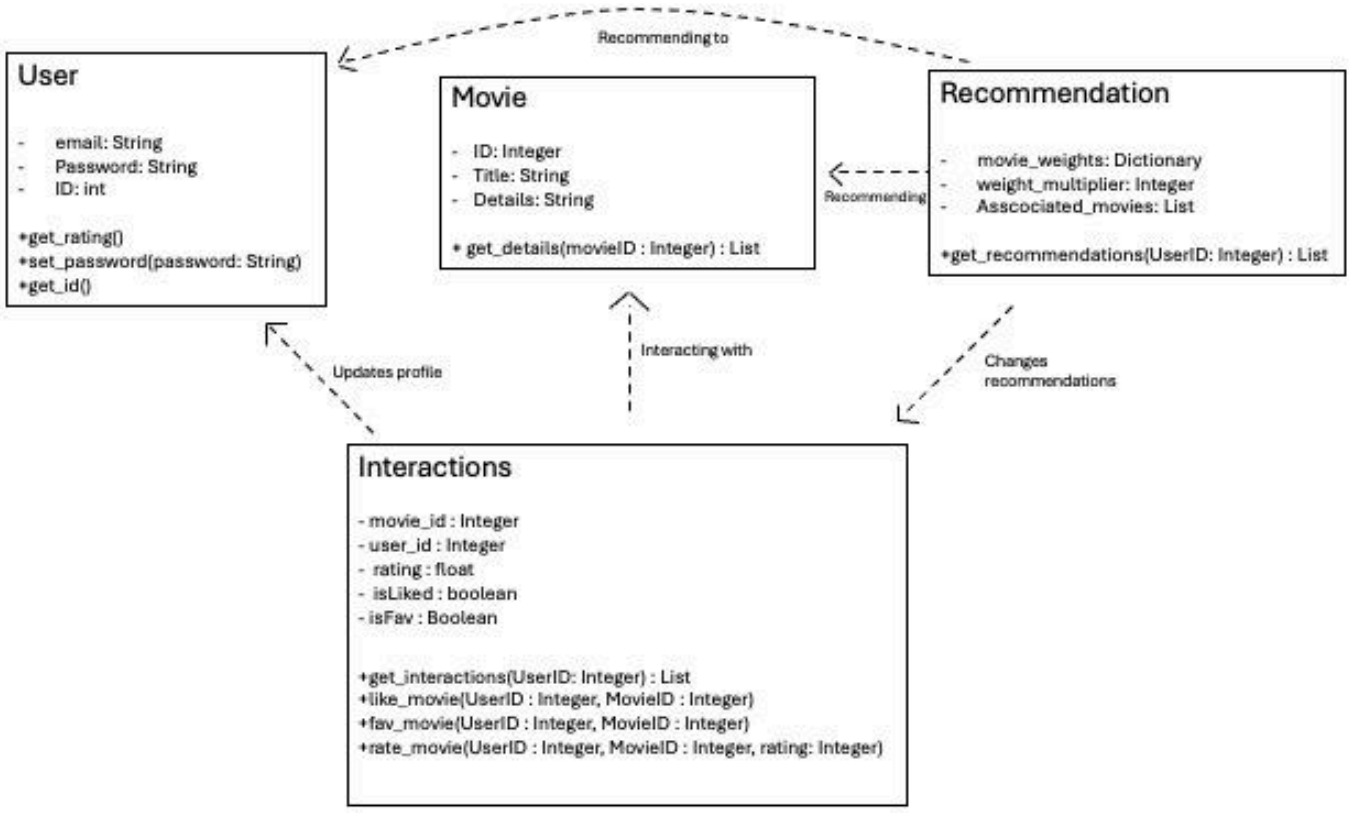
- Login functionality/User Authentication
- User Interface
    a. Wireframes for each page
    b. Prototype frontend
- Machine Learning Algorithm
- Create a sign up page for users to make accounts
- Implement forgot password/password process
- Create user accounts settings page
- Email verification
- Implement length and character requirements for password security
- Update frontend to log status responses for POST requests
- Add home page
- Add backend function to get all previous movie interactions (like, rate, favorite
- Implement contains functionality in search bar (searching “cars” would return cars 1, 2, and 3.)
- Style all pages
- Add sort by genre to home page
- Implement navigation bar to transition pages

6. System Description

Block Diagram:

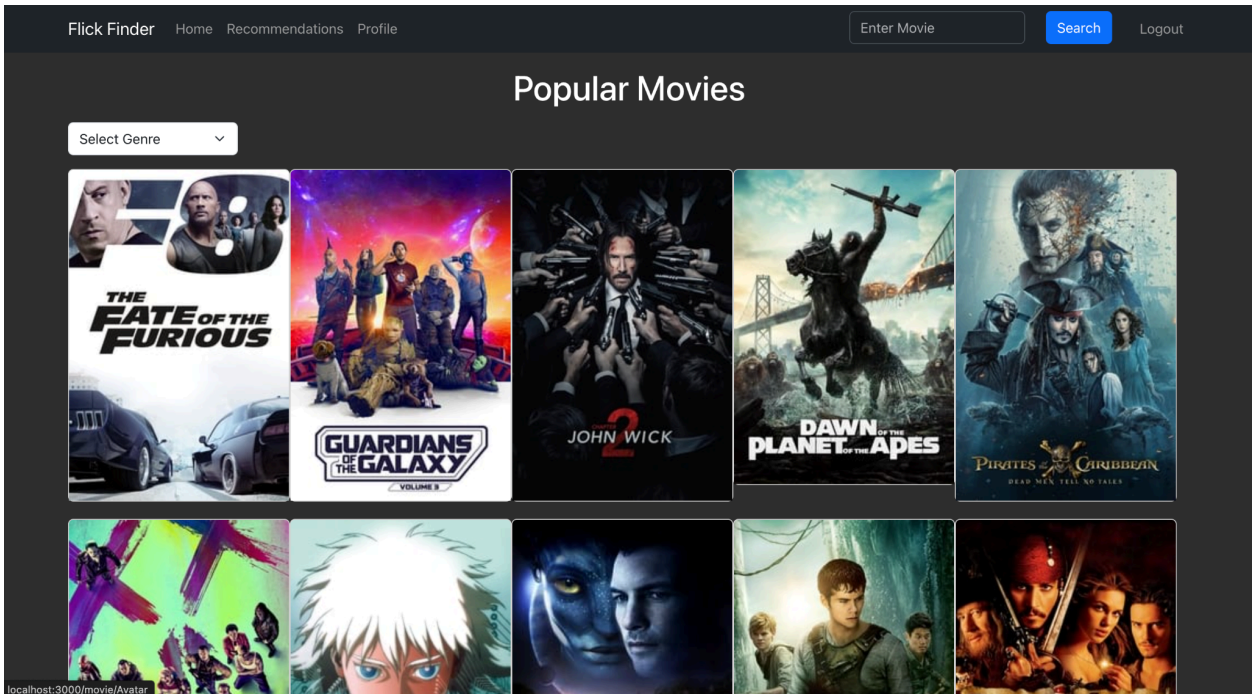


Class Diagram:

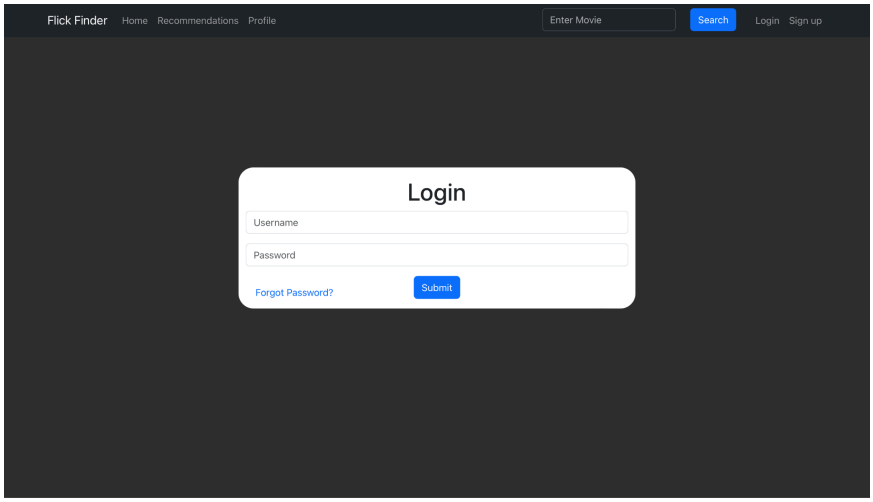


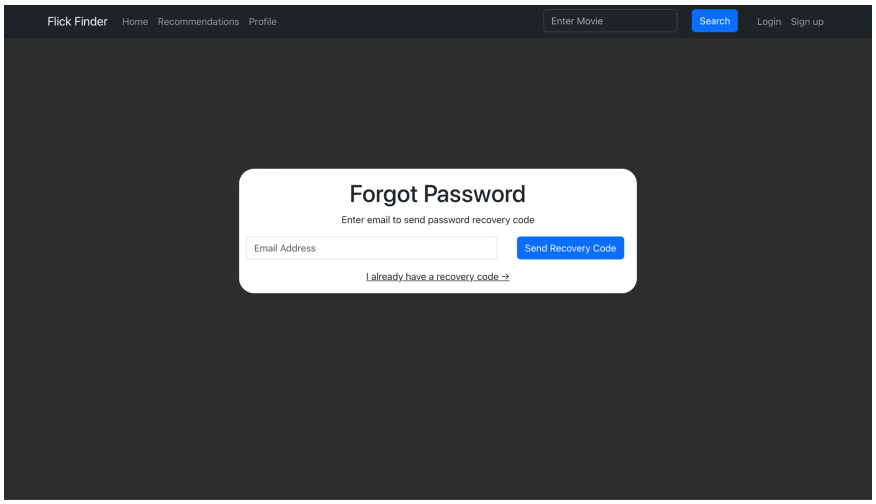
7. Current Status

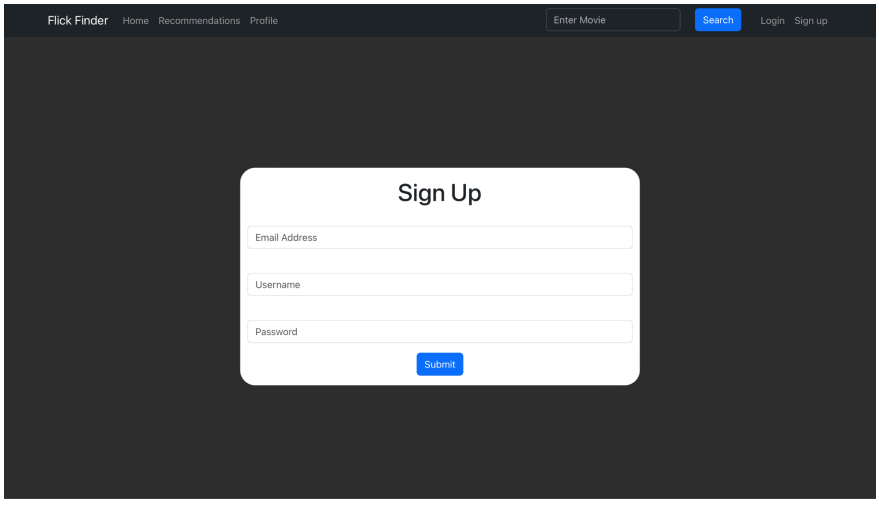
Screenshots:



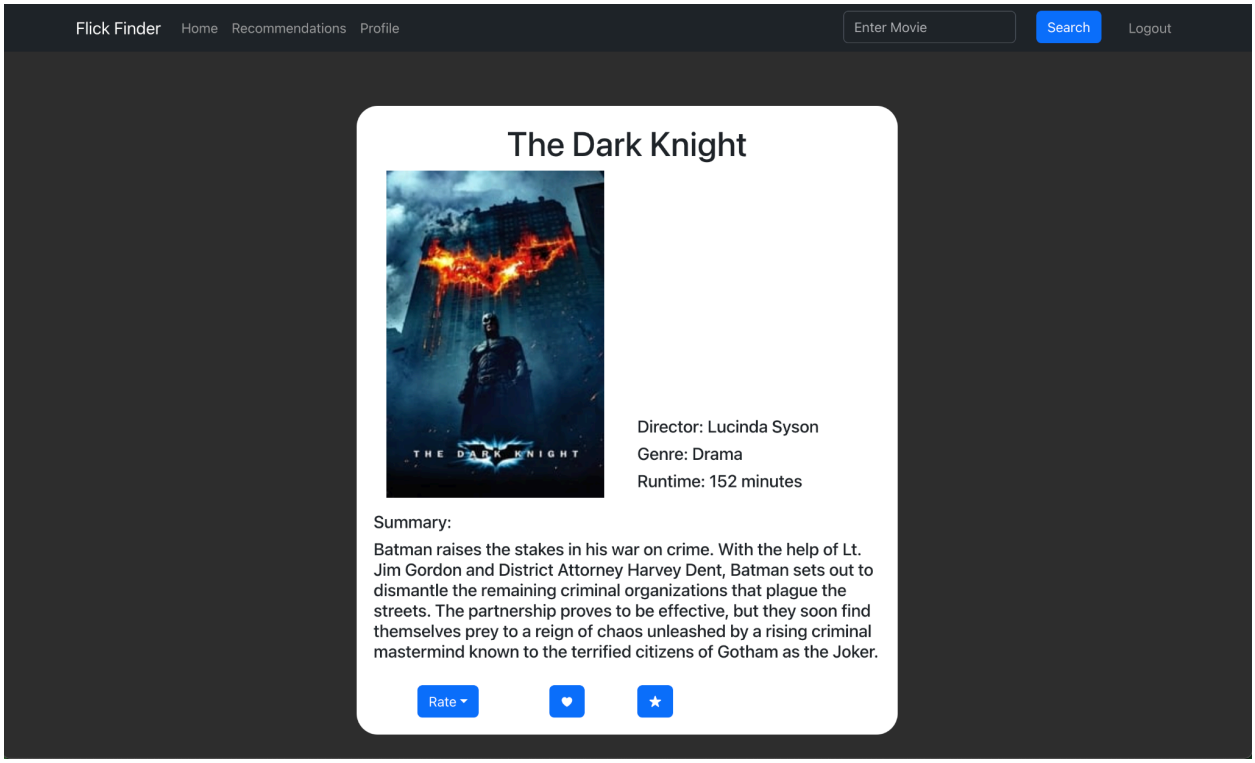
The above screenshot shows the home page that a user would land on when they initially visit our webpage. This is part of the frontend home page component. The possible inputs for this part of the system would be the ability to filter movies by genre and the movies being displayed would be the output. This part of the system was important so that the user would have a friendly landing page whenever they visited our website.


Flick Finder
Home
Recommendations
Profile
Enter Movie
Search
Login
Sign up
Login
Username
Password
Forgot Password?
Submit

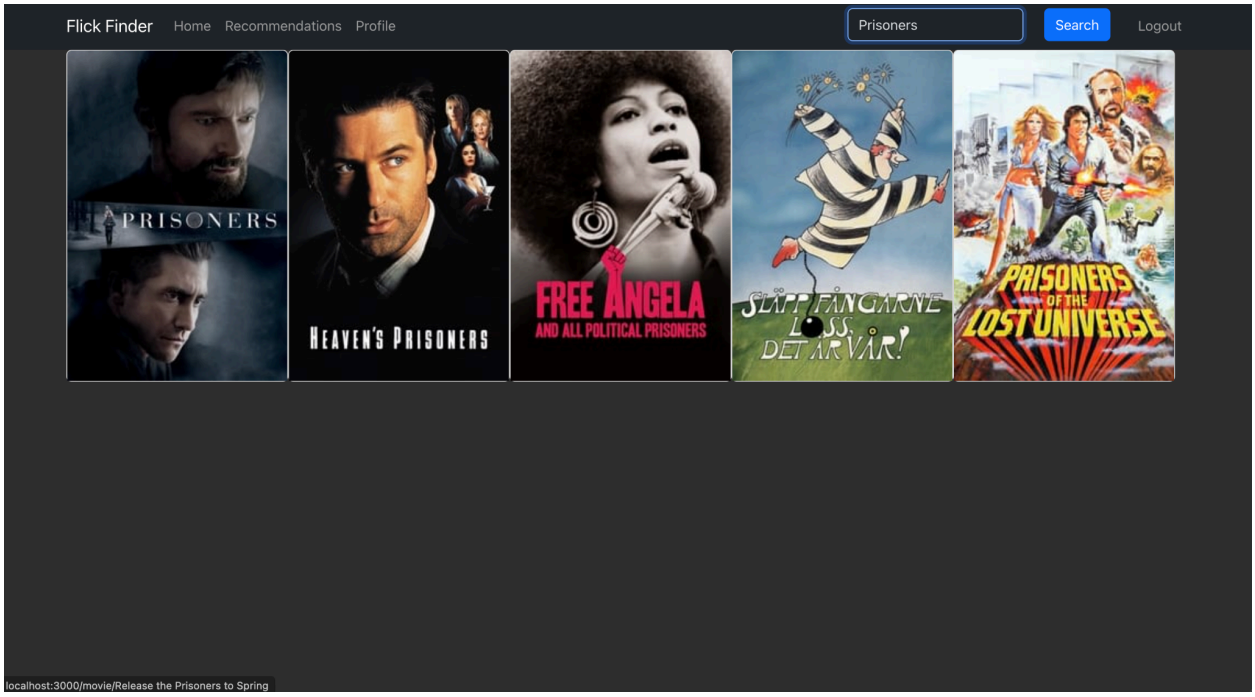

Flick Finder
Home
Recommendations
Profile
Enter Movie
Search
Login
Sign up
Forgot Password
Enter email to send password recovery code
Email Address
Send Recovery Code
I already have a recovery code →


Flick Finder
Home
Recommendations
Profile
Enter Movie
Search
Login
Sign up
Sign Up
Email Address
Username
Password
Submit

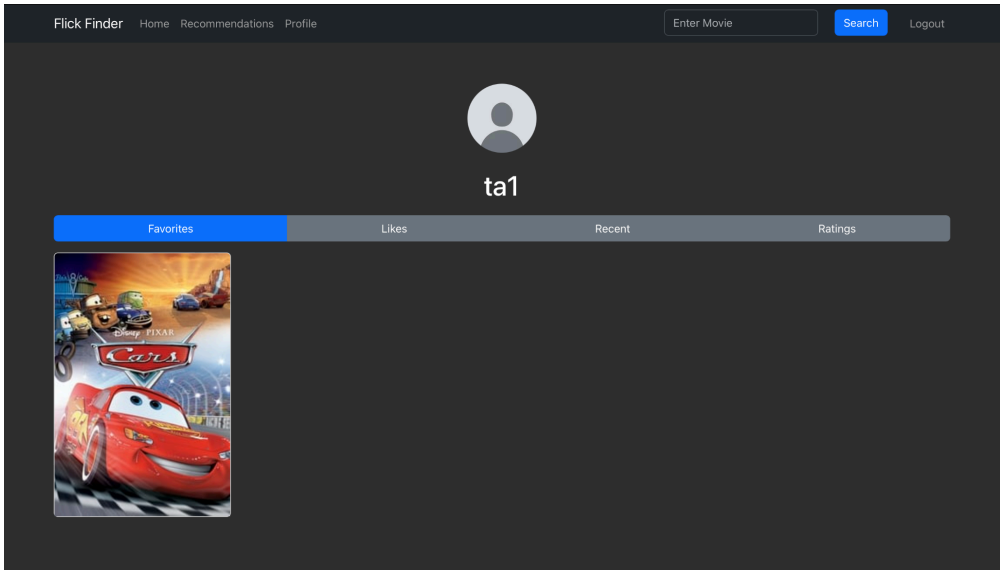
The above three screenshots show the sign up and login pages along with a forgot password page in the event a user needs to reset their password. This is part of the sign up and login frontend components. The inputs would be a username and password, along with an email while signing up. The above pages are important to allow users to create accounts that would be unique to them.

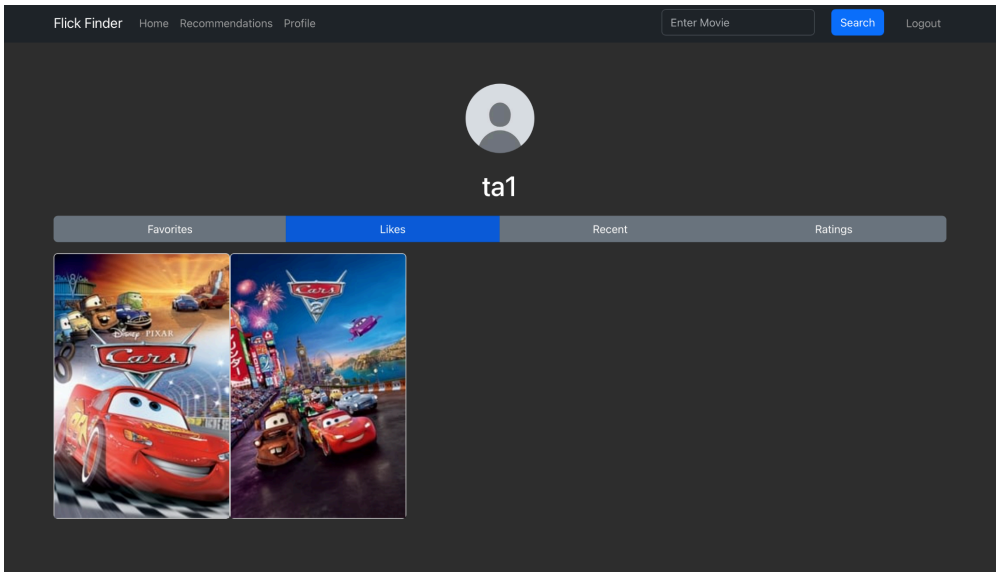


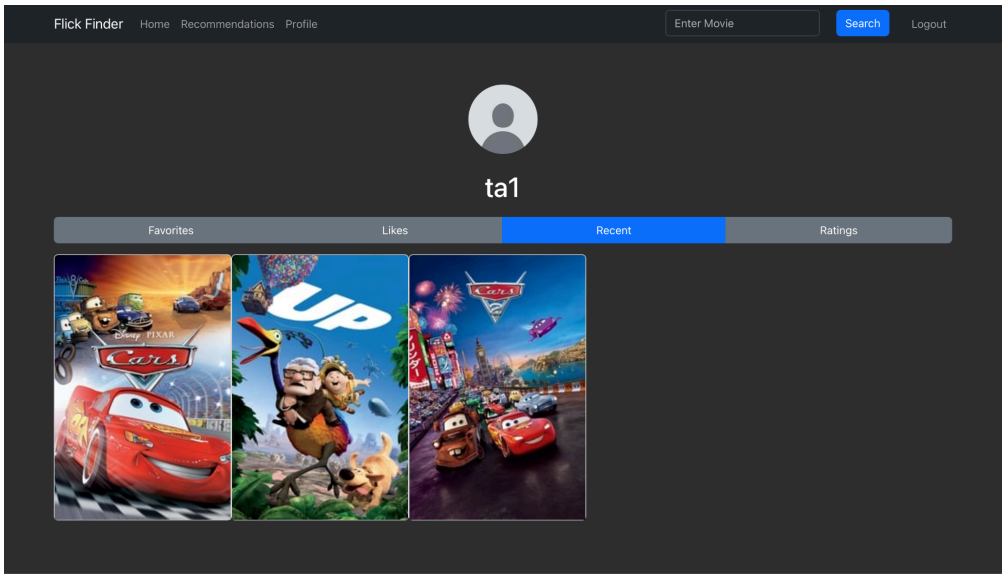
The above screenshot shows the movie page and the addition of components to rate, like, and favorite a movie. This page belongs to the movie, users and interactions classes. Each movie's details are found for the movie using the get_details() operation in the movie class and then displayed in the frontend. For each component the interactions class uses rate_movie() and like_movie() to add the user's input to the database. The user_id from the user class is used in each interaction call to ensure that the information is saved for the correct user.



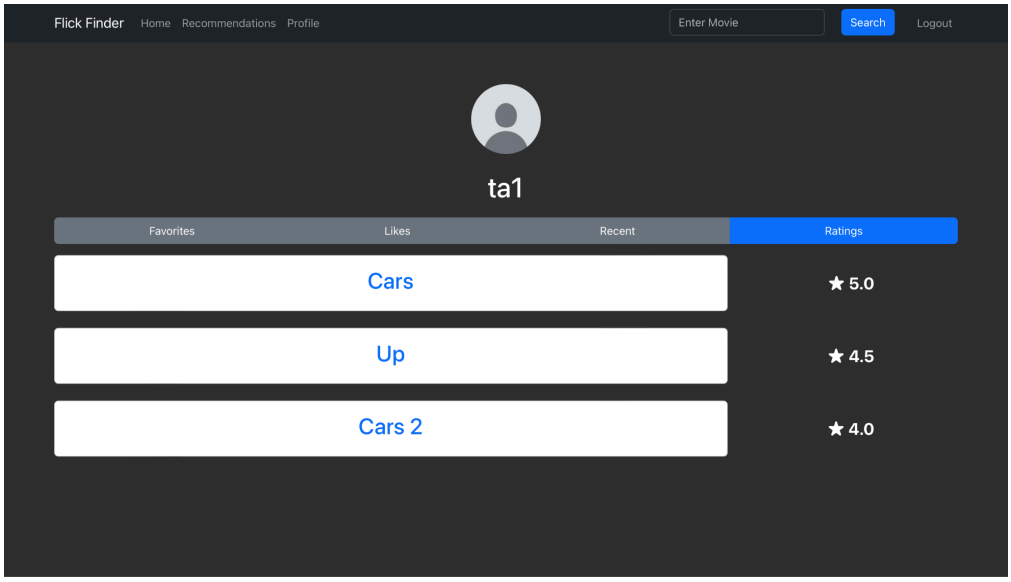
The above screenshot shows the search bar functionality. This shows the frontend search page component. The input is a movie or phrase that the user would like to search and the output is all the movies that contain the search keyword/s. This was important for the user to be able to search for movies they might have already seen so that they could interact with them and improve their future recommendations.


Flick Finder
Home
Recommendations
Profile
Enter Movie
Search
Logout
ta1
Favorites
Likes
Recent
Ratings


Flick Finder
Home
Recommendations
Profile
Enter Movie
Search
Logout
ta1
Favorites
Likes
Recent
Ratings



The above three screenshots show the user profile and the favorites, likes and recents pages. All three are part of the frontend user profile components. The inputs are the “favorites”, “likes”, and “recent” buttons and the output is the movies displayed associated with the given input for each user. This is important to the system because it allows the user to see the movies they interacted with and how they might have interacted with it (whether it be a movie they favorited, liked, or just rated).



The above screenshot shows the ratings section of the user profile page. This page is part of the frontend user profile component. The input is clicking the “ratings” button and the output is the above display of all movies rated by the user in descending order along with the rating given by the user. This is important because it allows the user to see all the movies that they have rated in one easy to follow location.

Test Cases:

- **Test Case 1**: Recommending Movies On the recommendations page, when I click the recommendation button, I am recommended five different movies based on movies that I have previously rated. Justification: This is important for our FlickFinder project because it proves that our system can successfully (1) receive my command (i.e., recommend), (2) send a request to execute the machine learning algorithm, (3) receive the data from the machine learning algorithm and, (4) display the data correctly on the front-end. Additionally, this feature is key to our project because we are building a movie recommender website and our project would just be a website where you could search for movies without this feature.
- **Test Case 2:** Liking Movies After I have watched a movie, when I click on the movie I am able to like the movie and it will show up on my profile in the “liked” movies section. Justification: This feature is important to our project because it demonstrates that our system can successfully (1) receive my command (i.e., like), (2) insert the user and movie data into our database and, (3) display the movie under the liked section of our website. Additionally, we believe that this feature is key to our project because our main goal is to recommend the user's

movies based on their thoughts about previous movies that they have watched. With having a like feature on our website we believe that it will enable us to provide better recommendations as we will have data on which movies the user likes.

- **Test Case 3:** Favoriting Movies After I have watched a movie, when I click on the movie I am able to favorite the movie and it will show up on my profile in the "favorited" movies section. Justification: This is important because it provides a more succinct selection of "favorited" movies. This differs from the like feature as the likes are more for user cataloging a large and generally liked selection of movies. The "favorited" section will only have a capacity of five movies to ensure that the movies they contain are only those which the user is most likely to enjoy and, therefore, most likely to rewatch. By putting a five movie limit on the "favorited" section, it will ensure that the users most enjoyed movies are easily accessible to the user.
- **Test Case 4**: Rating Movies When clicking the rate option for a particular movie, the user will be prompted to enter a value of 0-5 stars which will be saved in the database. Justification: This feature is important because it demonstrates that our system can successfully (1) receive my command (i.e., rate), (2) send a request to the front-end to prompt the user to input a rating, (3) receive the user-inputted data from the front-end and, (4) insert the user rating into our database. Another thing to mention is that the saved user data on ratings and likes will be used to recommend a unique and more fitting combination of movies to the user.

- **Test Case 5:** Searching Movies When clicking on the search bar, the user will be able to input the name of a movie and get a result back with a poster and details concerning the movie. Justification: This feature is important because it demonstrates that our system can (1) receive a user input, (2) properly query the database to receive the right movie, and (3) return details to the user in a comprehensive and efficient manner. This function will also serve as a base to all other functions allowing users to find, rate, and like previously seen movies to better enhance the quality of their movie finding experience

Combinatorial Testing:

Factors	**Values**
User Logged In	(0, 1)
User Rating	0-5
User Likes	Liked/Not-liked (0, 1)
User Favorites	Favorite/Non-favorite (0, 1)

8. Project Management

Date	Description
1. 3/10 2. 3/26 3. 3/26	1. Decided to implement user authentication in backend instead of frontend

	2. Added a likes table to the SQL database to store user information on liked movies 3. Added a recommendations table to the SQL database to store similar movies so that the recommendation process is more efficient.

9. Review and Retrospective

9.1 Sprint Review

- Our customers were interested in being able to follow/view other accounts likes/favorites/dislikes. Along with this, we had thought about altering our recommender algorithm to account for movies that other users with similar interests might have interacted with. Our idea was to move from a strictly content based recommender to a hybrid recommender using content and collaborative algorithms. We did not believe this was in scope for our project due to the extensive amount of overhead that this would require. In a 10 week timeframe, we decided to keep our focus on the system and its necessary functions (recommending, liking, favoriting, etc). In the future, if we were to keep maintaining and building upon this system, we would like to evolve our system with things like user to user interactions, but this is outside the current scope of

our project. Our customer enjoyed the product that was delivered, and would like to see the system continue to be developed.

9.2 Sprint Retrospective

- Our sprint was very productive leading into our final presentation. We completed all of our tasks that we intended on completing for the sprint, and ultimately delivered working software that we could demo for our presentation. We were able to meet all of the goals for this current iteration and for the project overall. Some of the limitations of our dataset were noticed during this iteration, but there was nothing that hindered our progress. Looking back, we might have been able to distribute a wider range of work to each group member. However, due to differing experience and confidence levels, we were unable to branch out as far as might have been helpful and had more mentoring type dynamics than pure developer relationships.

10. Team Management

Tristan Allen - Scrum Master/Developer

- Contributions: Primarily backend development. Implemented initial PoC to make sure that our plan for the software stack was feasible. Led development of the backend and API. Assisted with frontend development of user profile and login/logout pages.

Daniel Carter - Developer

- Contributions: Primarily focused on the database, by importing and organizing the data in our MySQL database, helped out a little with the profile and account settings page on the front-end

Will Cox - Product Manager/Developer

- Contributions: Primarily frontend development. Worked in the backend for multiple features (search, recommend). Styled the frontend.

Josiah Jackson - Developer

- Contributions: Primarily backend error checking. Added error handling for inputs of usernames and passwords. Contributed to frontend design aspects.

We faced multiple problems throughout the development of our system. The most prevalent problems were: Challenges scheduling meetings around busy schedules/athletics and different levels of experience. Here is what the problem was and how we overcame it.

Difficulty scheduling: Like many of the other groups, it was hard for us to align our schedules to have consistent meetings every week. While there are gaps in our class schedule, there were typically other scheduling conflicts that would arise due to classwork in other classes. We overcame this by having regular meetings on Sunday nights to discuss progress for the sprint and possible tasks moving forward. Additionally we would briefly talk before/after each class session on Tuesdays and Thursdays to ask any clarifying questions or resolve any major roadblocks that could've come up. If anyone ever had anything small that was impeding their progress on work items we always had an open line of communication through text.

Different levels of experience: Tristan and Will had more experience with system interactions and team development through previous internships in which they worked on teams using the agile-scrum framework. Since they were the most experienced, they took on most of the early

development tasks so that Daniel and Josiah could get more familiar with the technologies that were being used in our system. Daniel and Josiah also worked in their own branches so that they could experiment with our system.

If we were a third party giving advice to our team, we would recommend being less dependent on outdated data CSV for movie data. While many people's favorite movies are still contained in this data, it would be better for the user experience to have up-to-date movies. This could be done in different ways, one way that we thought of was through the use of an external API., such as TMDB.