Software Engineering CSC 648/848

The MovAI Project - Milestone 2

Section 4, Team 2

Team

Team Lead	Ashmitha Pais
Scrum Master	Steve Betts
Github Lead / Back-End	Preet Dhaliwal
Back-End Lead	Chris Farnsworth
Front-End Lead	Abdul Barrie
Product Owner / Front-End	Nathan Loo

History

Initial Draft	28/03/2023

1. Data Definitions V2

A. Movie

a. Movie ID
   i. Meaning: Unique ID referring to a movie
   ii. Usage: Used to identify a specific movie, allows for differentiation between movies that share the same title

b. Title
   i. Meaning: Title of movie
   ii. Usage: Used in searching and displaying movies to a user

c. Alternate Title
   i. Meaning: An alternative title for a movie
   ii. Usage: Used for searching and displaying movies that may have multiple releases with different titles, such as non-english language films that have an english title for their US release.

d. Year
   i. Meaning: The year in which a movie was released
   ii. Usage: Used to display the movie release year, and for filtering recommendation results

e. Runtime
   i. Meaning: The length of a movie in terms of minutes
   ii. Usage: Used to display the length of a movie, and for filtering recommendation results

f. Genres
   i. Meaning: The genres to which a movie belongs (could be more than one)
   ii. Usage: Used to display the genre(s) of a movie, and for filtering recommendation results

B. User

a. User ID
   i. Meaning: Unique ID referring to a user
   ii. Usage: Used internally to identify users and map users to movies

b. Username
    i. Meaning: Unique handle that refers to a user
    ii. Usage: Used for login and allowing a user to uniquely identify themselves.
c. Email
    i. Meaning: Email address used by a user for account creation
    ii. Usage: Used for account creation and for things such as account recovery.
d. Password
    i. Meaning: Passcode for securing entry to a user's account
    ii. Usage: Used to keep users' account information private
e. User Recommendations
    i. Meaning: A list of past recommendations for the user
    ii. Usage: Used to track past recommendations and to prevent duplicate suggestions

C. Genre

a. Genre ID
    i. Meaning: Unique ID referring to a genre category
    ii. Usage: Used in mapping genres to movies
b. Genre Title
    i. Meaning: Name of a movie genre
    ii. Usage: Used for putting movies into traditional categories familiar to users for filtering recommendations

D. User Movie

a. User ID
    i. Meaning: ID of a User entity
    ii. Usage: Used to map a user to a movie
b. Movie ID
    i. Meaning: ID of a Movie entity
    ii. Usage: Used to map a movie to a user
c. Rating
    i. Meaning: A users rating of their enjoyment of a movie
    ii. Usage: Used as a parameter in API calls for movie recommendations

E. OpenAI API Call

a. Required Parameters

i. Meaning: Parameters passing the data used to generate recommendations

ii. Usage: Used to pass the list of movies a user has watched and their rating of those movies to OpenAI

b. Optional parameters

i. Meaning: additional parameters to narrow recommendations

ii. Usage: Used to allow users to narrow results by restricting recommendations to certain time periods, genres, etc.

F. OpenAI API Response

a. Recommended Movie

i. Meaning: A movie that is recommended to a user based on ratings of previously viewed movies

ii. Usage: Used to return recommendations, and to track movies recommended to users in past requests

b. Response Generation Timestamp

i. Meaning: A timestamp of when a movie recommendation was created

ii. Usage: Used in showing users previous recommendations

2. Functional Requirements V2

REQ 1: [Priority 1] Users can add movies from a database to their account.

- They type in a movie title from a search bar and press ‘add’ to add that to their list of movies.

REQ 2: [Priority 2] Users can view the movies they've been recommended to or added to their account.

- The list of movies they’ve added have 5 attributes: Title, Runtime, Year of Release, Genre(s), and Rating.

REQ 3: [Priority 1] Users can ‘Like’ or ‘Dislike’ movies listed on their account.

- This rating will help the AI generate recommendations.
- Ratings start at a default of null or nothing, to indicate they haven’t watched it.

REQ 4: [Priority 1] Users can generate new movie recommendations based on the following attributes:

1. Liked & Disliked Movies (Unwatched movies are not part of generation)
2. Genre (1 to 3 - intersectionality)
3. Age Rating (G, PG, PG13, R)
4. Runtime (Under X number of minutes)
5. Year (“I want a movie made between 1940 and 1960”)

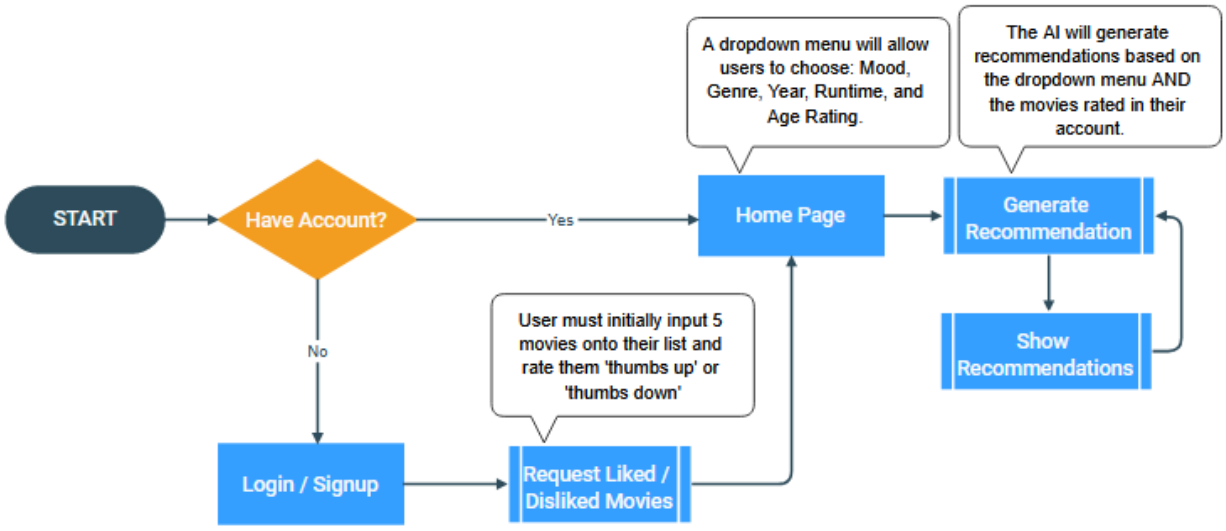
REQ 5: [Priority 3] Users can modify their existing recommendations to generate new recommendations.

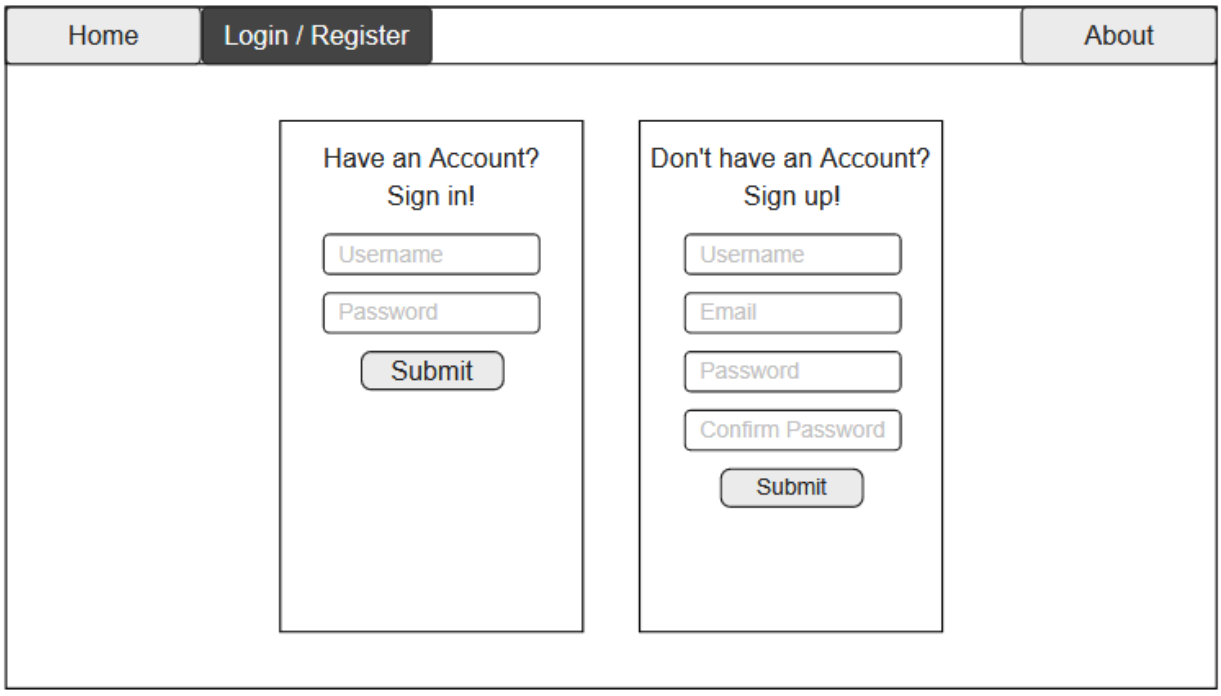
- They can change a single one of the attributes above to generate the new attributes.

REQ 6: [Priority 3] User recommendations are added to their account, which can also be ‘Liked’ or ‘Disliked’ to further fine tune their new recommendations.

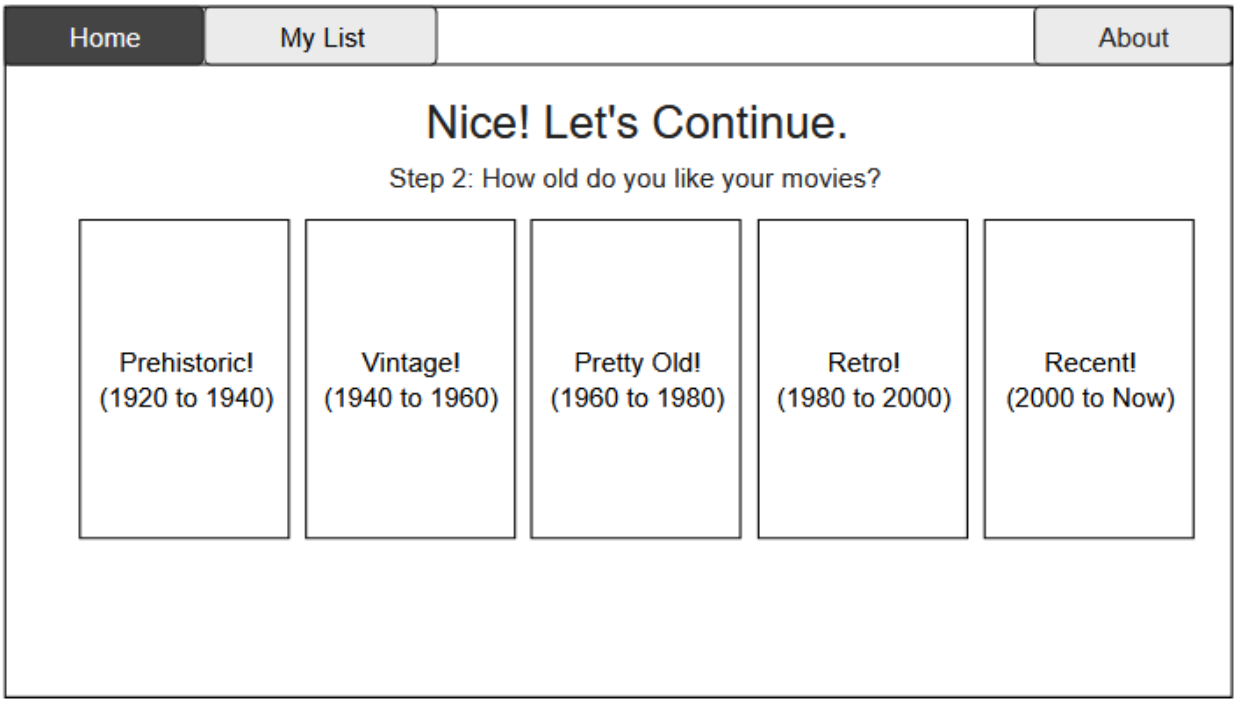
- There will be an ‘add’ button next to the recommendations that hover over, which will add it to their list.

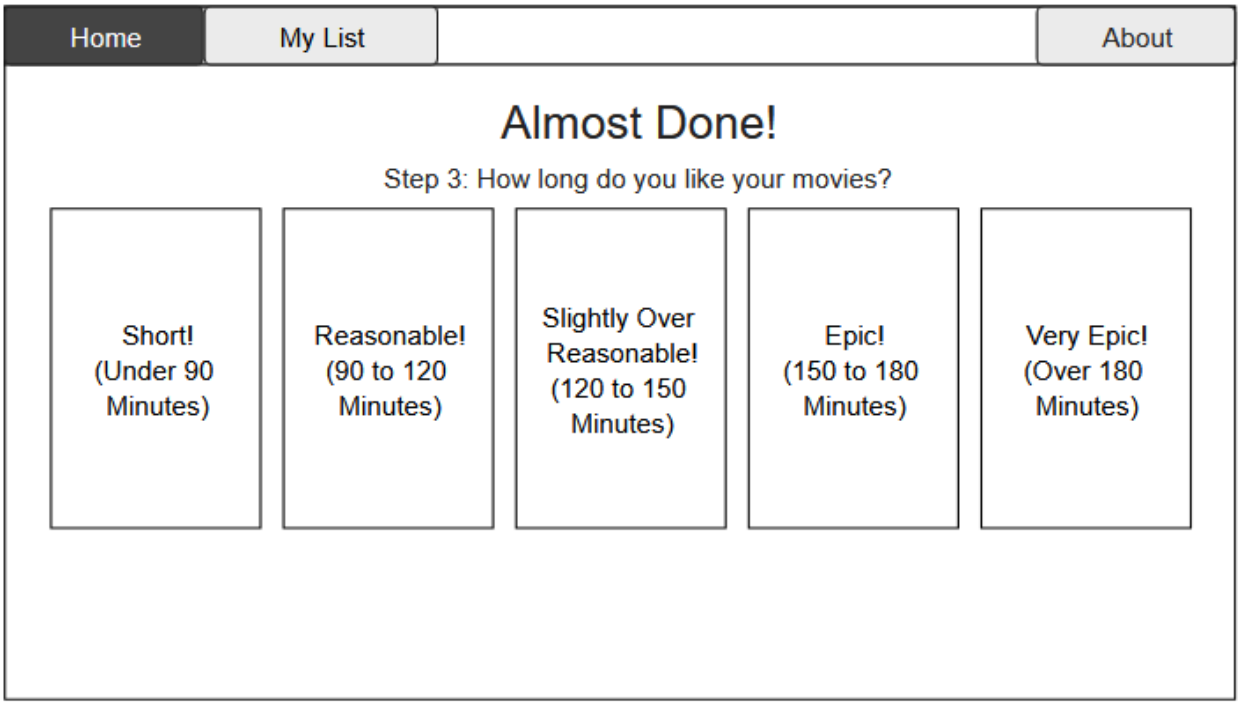
3. UI Mockups and Storyboards

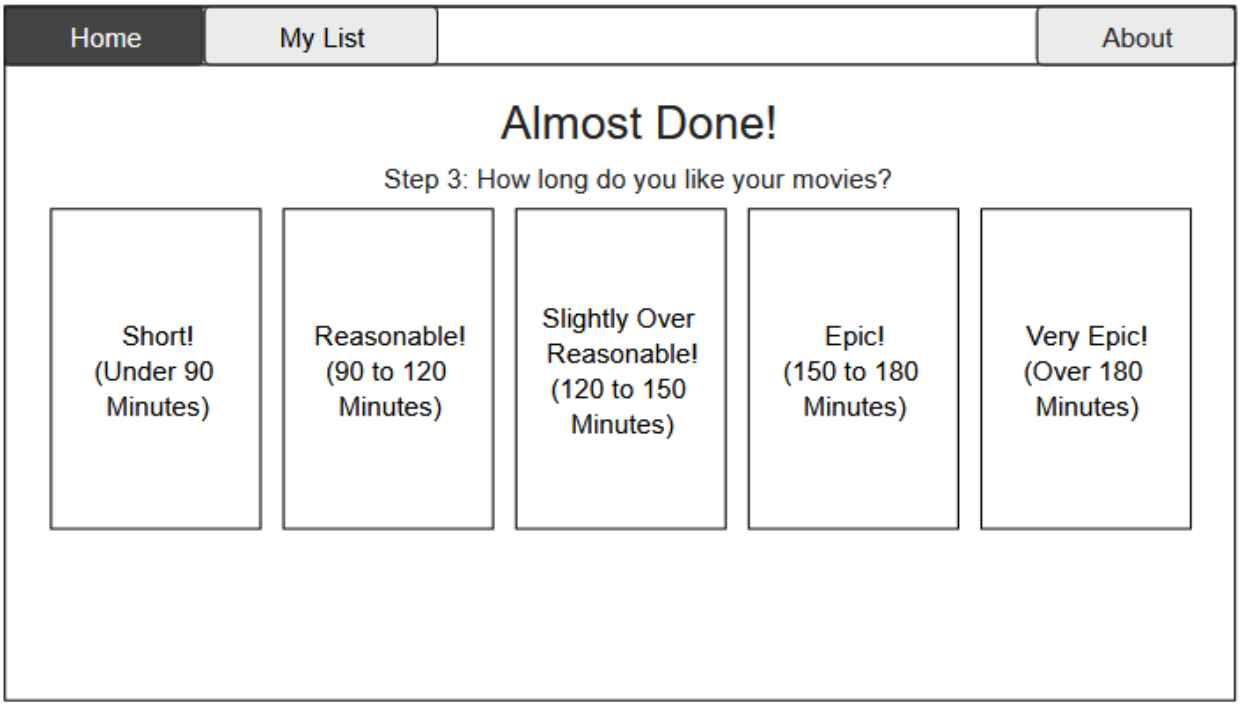


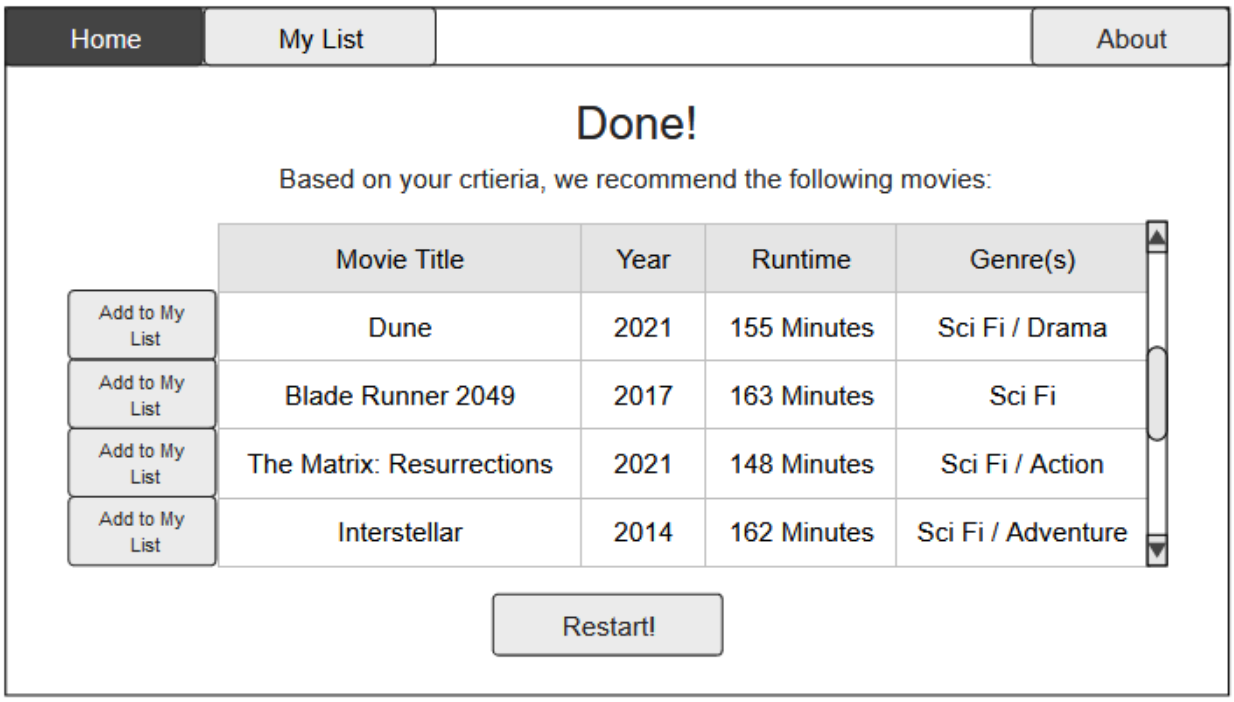


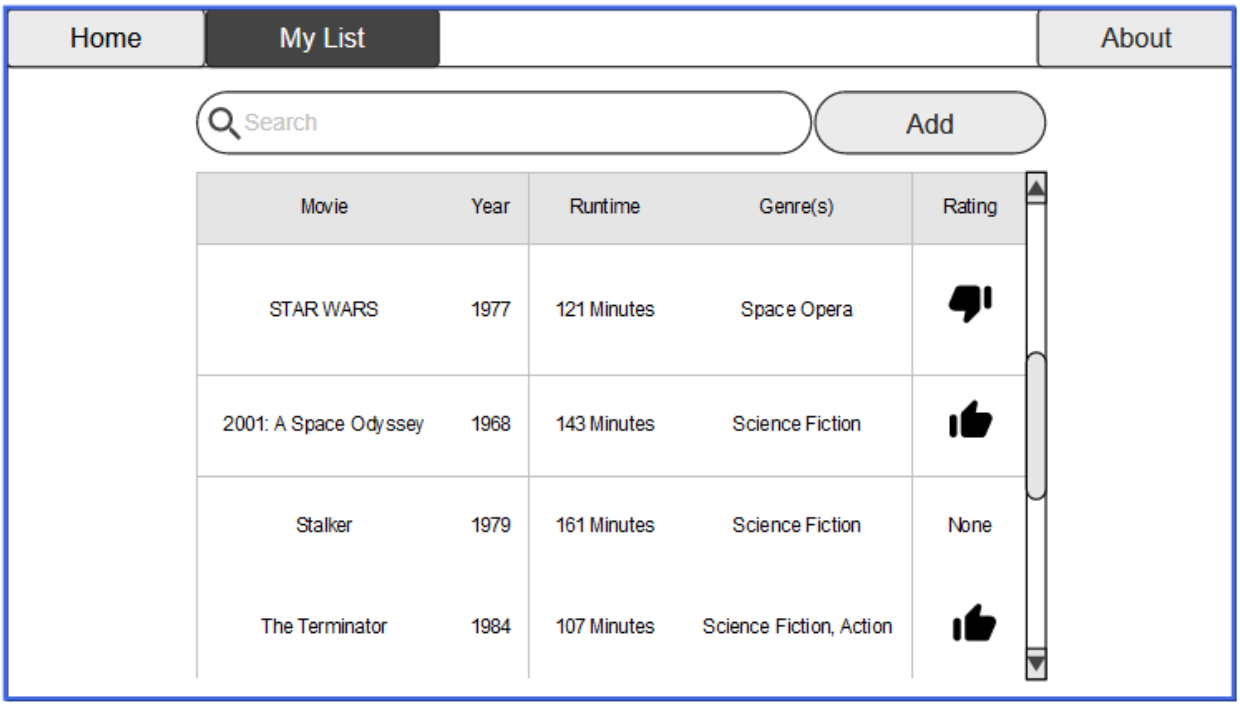

Home
My List
About
Lets get started!
Step 1: What genres do you want?
Action
Comedy
Drama
Horror
Romance
Thriller
Mystery
Crime
Animation
Sci Fi
Fantasy
Adventure


Home
My List
About
Nice! Let's Continue.
Step 2: How old do you like your movies?
Prehistoric!
(1920 to 1940)
Vintage!
(1940 to 1960)
Pretty Old!
(1960 to 1980)
Retro!
(1980 to 2000)
Recent!
(2000 to Now)

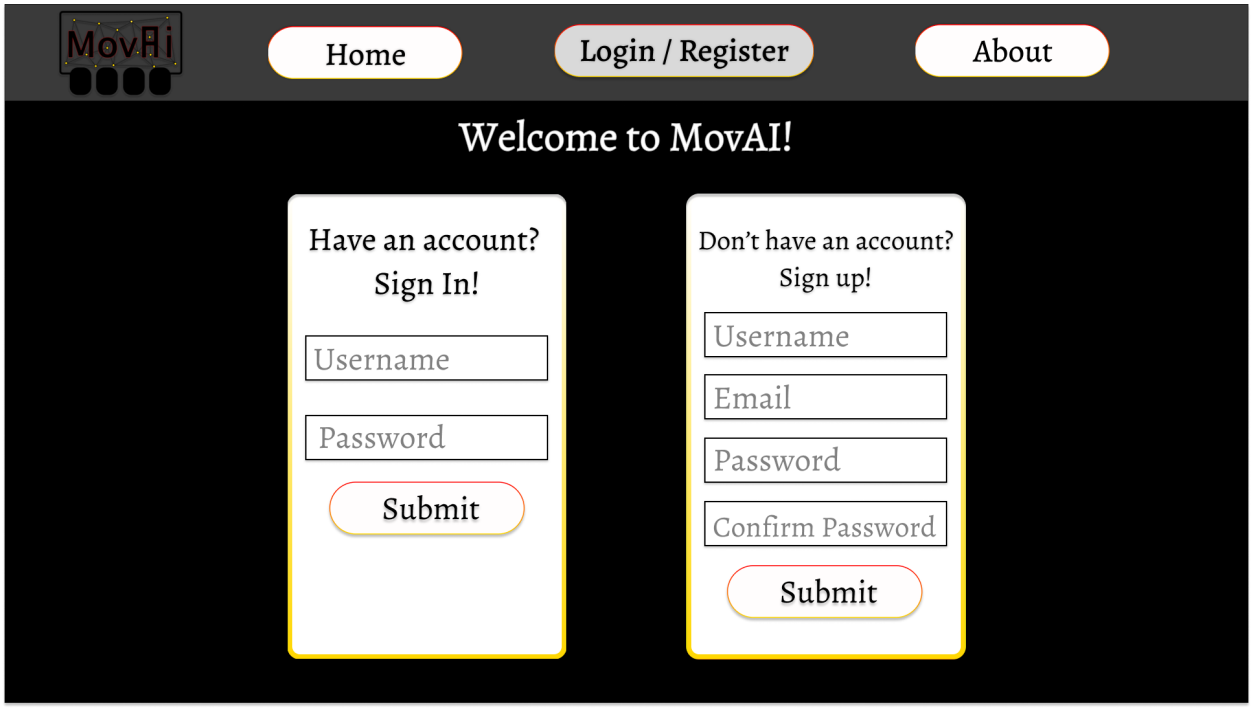

Home
My List
About
Almost Done!
Step 3: How long do you like your movies?
Short! (Under 90 Minutes)
Reasonable! (90 to 120 Minutes)
Slightly Over Reasonable! (120 to 150 Minutes)
Epic! (150 to 180 Minutes)
Very Epic! (Over 180 Minutes)


Home
My List
About
Almost Done!
Step 3: How long do you like your movies?
Short! (Under 90 Minutes)
Reasonable! (90 to 120 Minutes)
Slightly Over Reasonable! (120 to 150 Minutes)
Epic! (150 to 180 Minutes)
Very Epic! (Over 180 Minutes)

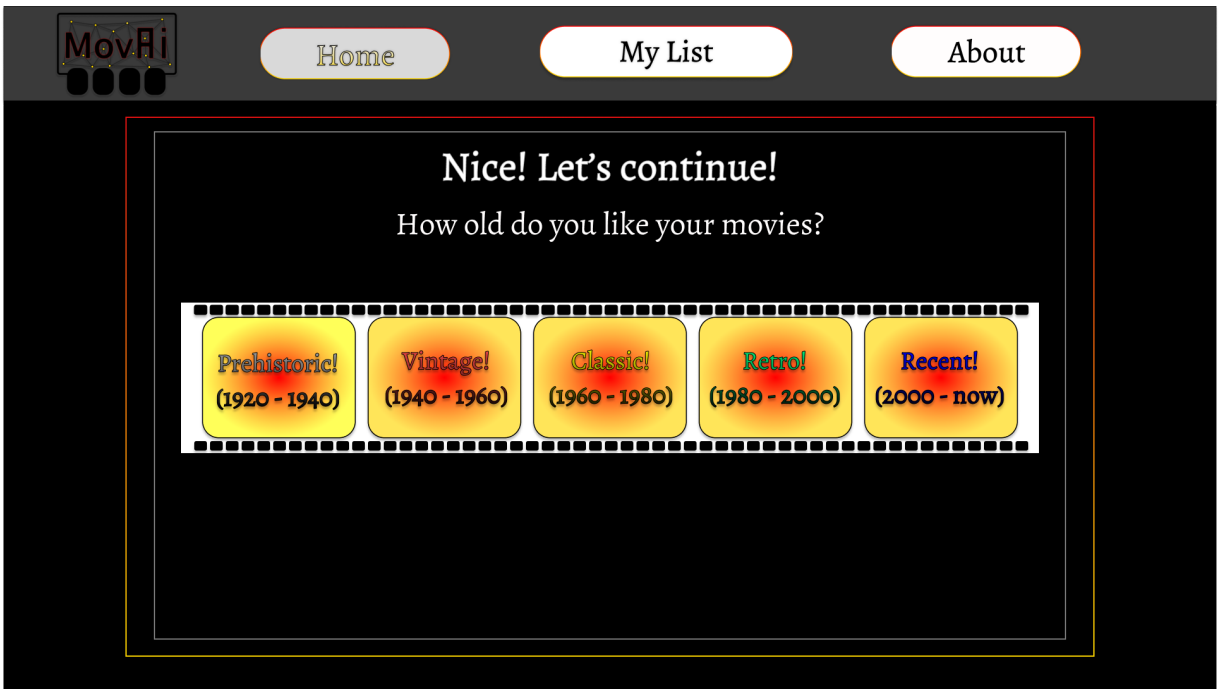


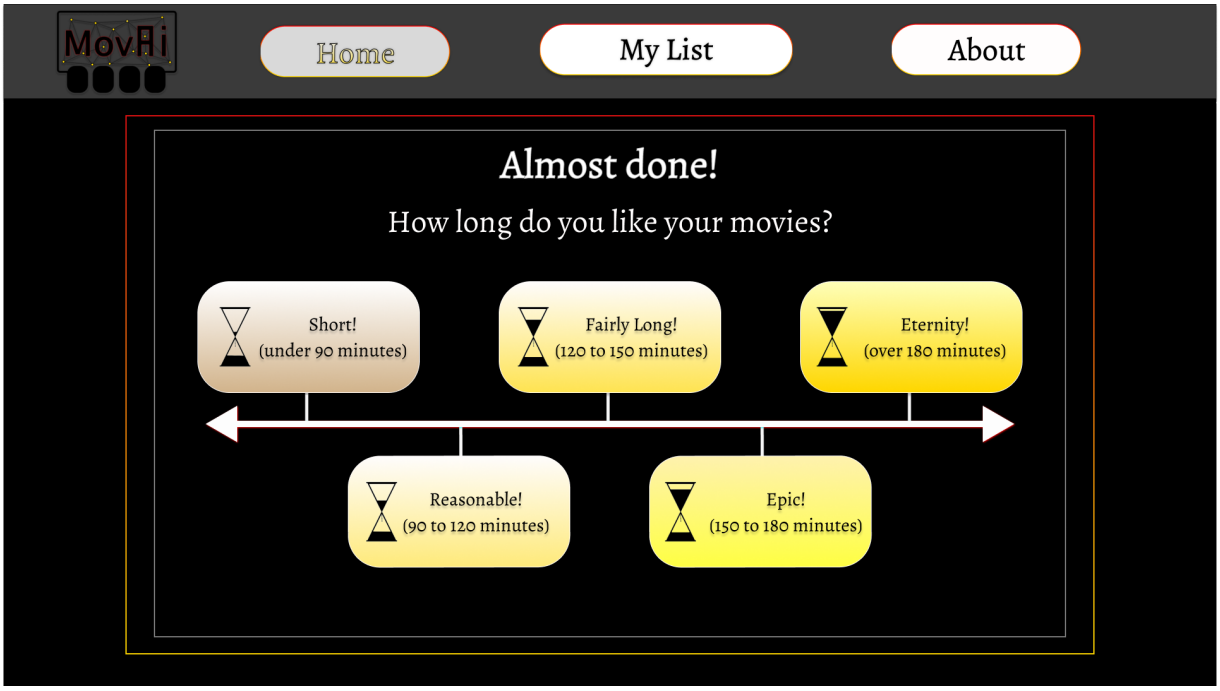


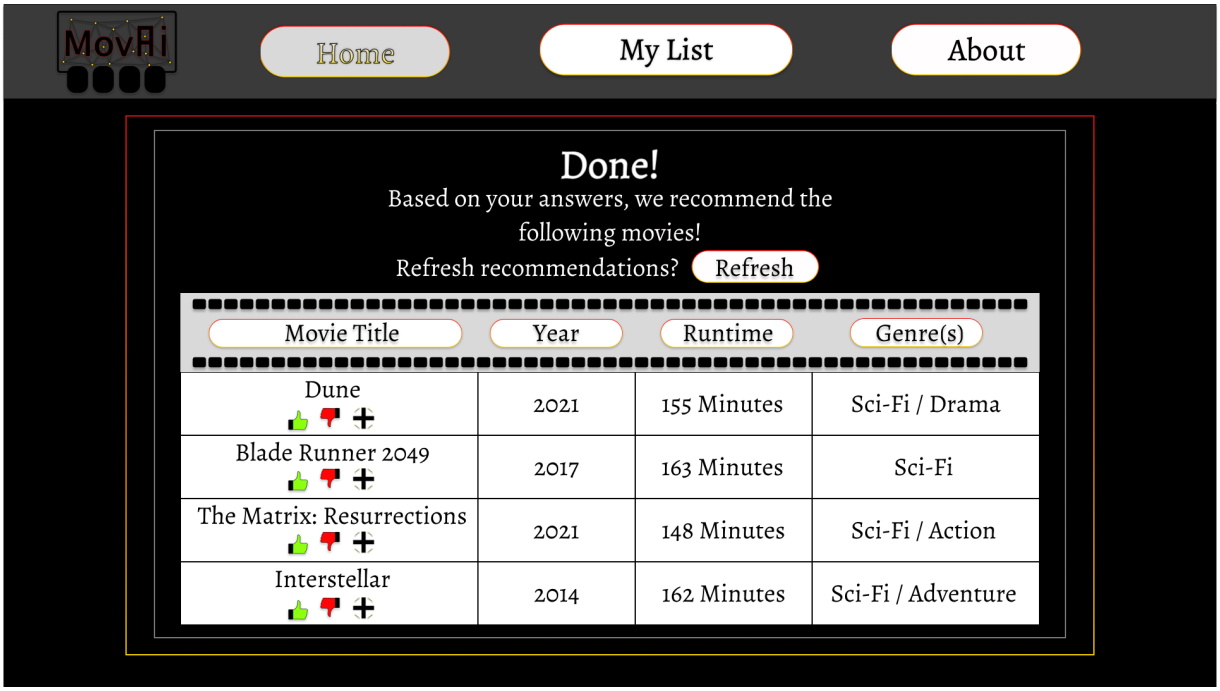
UI Mockups - Figma Designs

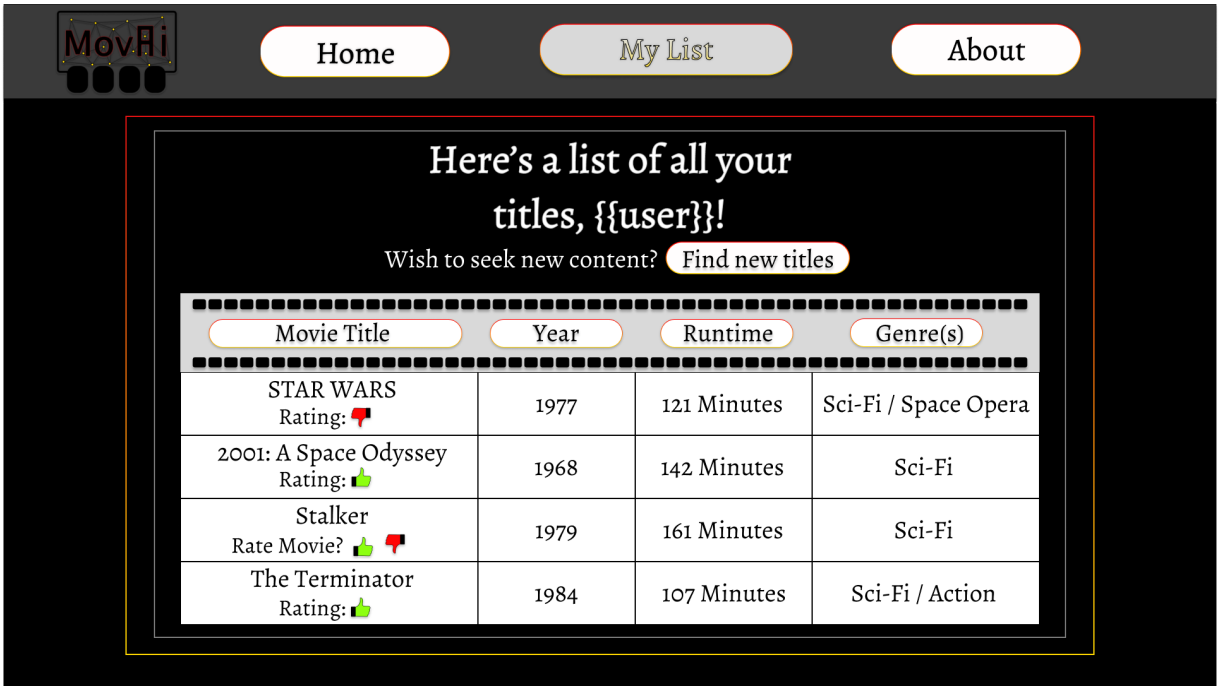

MovAi
Home
Login / Register
About
Welcome to MovAI!
Have an account?
Sign In!
Username
Password
Submit
Don't have an account?
Sign up!
Username
Email
Password
Confirm Password
Submit


MovFi
Home
My List
About
Let's get started!
First off, what are your favorite genres?
Action
Adventure
Comedy
Drama
Horror
Thriller
Mystery
Crime
Animation
Sci-Fi
Fantasy
Romance


MovFi
Home
My List
About
Nice! Let's continue!
How old do you like your movies?
Prehistoric!
(1920 - 1940)
Vintage!
(1940 - 1960)
Classic!
(1960 - 1980)
Retro!
(1980 - 2000)
Recent!
(2000 - now)


MovFi
Home
My List
About
Almost done!
How long do you like your movies?
Short!
(under 90 minutes)
Fairly Long!
(120 to 150 minutes)
Eternity!
(over 180 minutes)
Reasonable!
(90 to 120 minutes)
Epic!
(150 to 180 minutes)


MovFi
Home
My List
About
Done!
Based on your answers, we recommend the
following movies!
Refresh recommendations?
Refresh
Movie Title
Year
Runtime
Genre(s)
Dune
2021
155 Minutes
Sci-Fi / Drama
Blade Runner 2049
2017
163 Minutes
Sci-Fi
The Matrix: Resurrections
2021
148 Minutes
Sci-Fi / Action
Interstellar
2014
162 Minutes
Sci-Fi / Adventure



Link to Figma Designs:

https://www.figma.com/file/DiIu8eSnUKclwibu3W0Wyh/MovAI-Design-%231---Login%2FSignup?node-id=31%3A1833&t=rZIsoqpzTkoKqhrH-1

4. High-Level Architecture & Database Organization

Database Organization:

Tables & columns:

---

user — Allowed operations: search, add, delete, display

- id
- username
- email
- password

---

movie — Allowed operations: search, display

- id
- title
- alt_title
- year
- runtime

---

user_movie — Allowed operations: search, add, delete

- id
- movie_id
- user_id
- rating

---

user_rec — Allowed operations: search, add, delete

- id
- movie_id
- user_id

---

genre — Allowed operations: search, display

- id
- genre

---

movie_genre Allowed operations: search

- id
- movie_id
- genre_id

APIs

MovAI API:

- POST: signup
  - Creates and authenticates new user
  - If successful, responds with redirect to main app
- POST: login
  - Authenticates a user
  - If successful, responds with redirect to main app
- POST: get_movie_recommendations
  - Receives genres, years, runtime as POST data
  - Constructs GPT query based on POST data and user movie ratings
  - Responds with parsed GPT results
- POST: rate_movie
  - Updates a user_movie table entry based on POST data
- GET: view_movies
  - Responds with all movie entries associated with logged-in user via the user_movie table

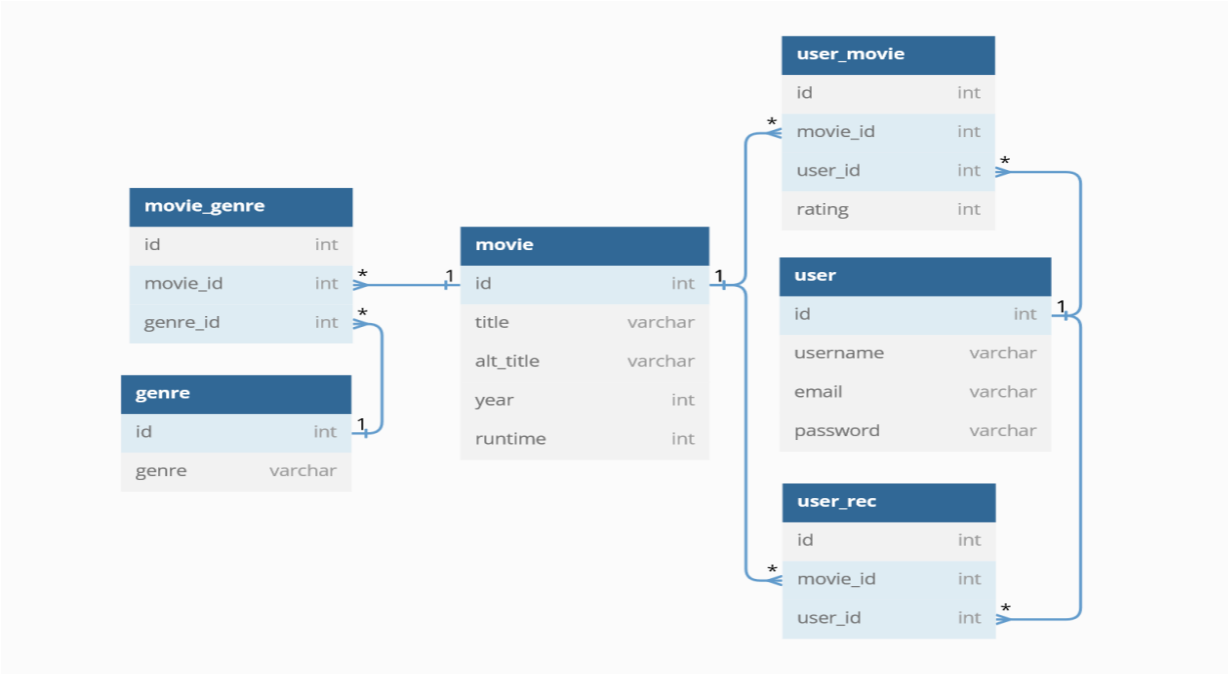
3rd Party APIs:

- OpenAI API
  - OpenAI.Completion.create()

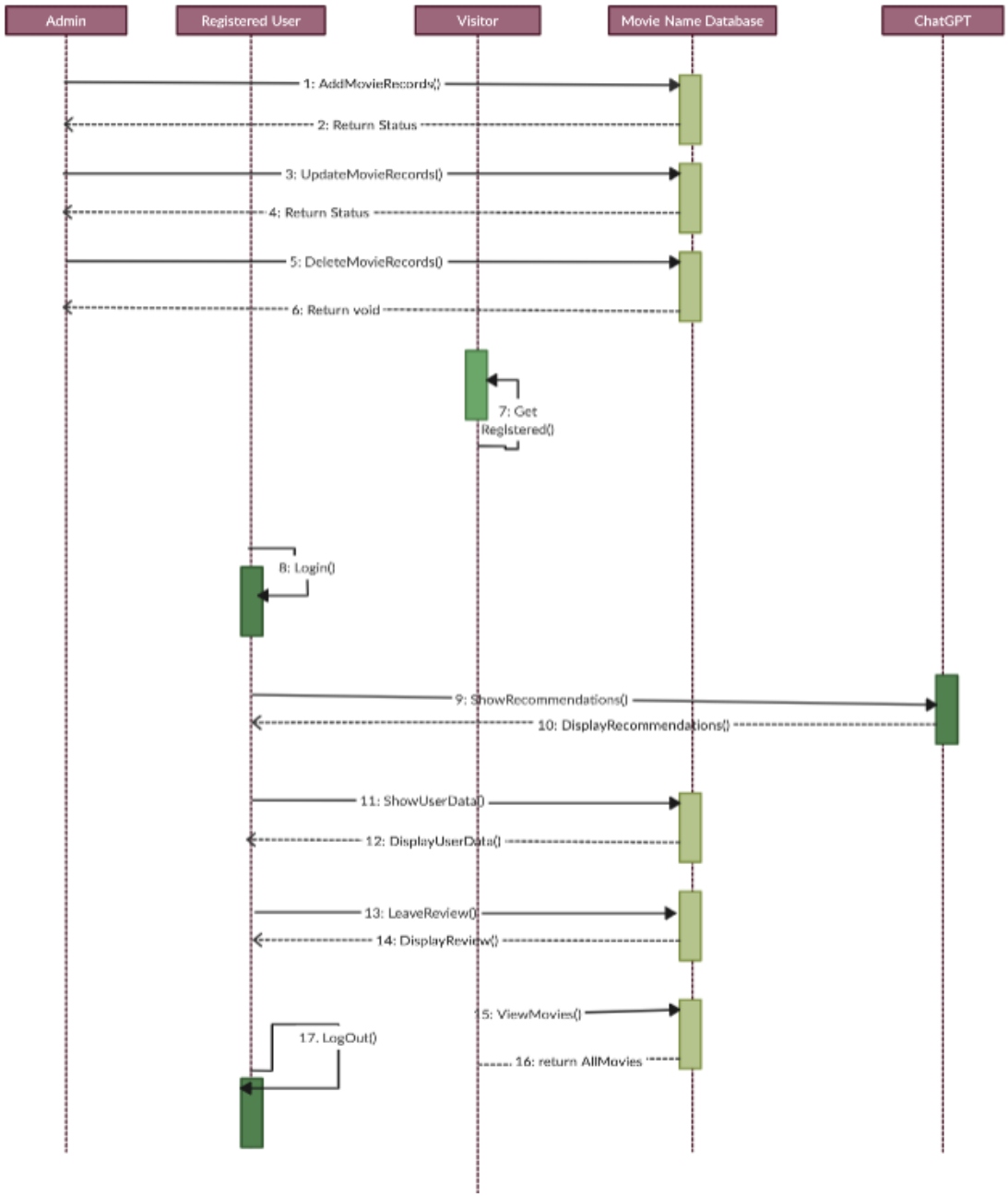
Open-Source Components:

- Django RESTful Framework

5. High- Level UML Diagrams



High-Level UML Class Diagram



High-Level Sequence Diagram

6. Identify actual key risks for your project at this time

Identify only actual and specific risks in your current work such as

- skills risks and mitigation plan
    - Do you have a proper study plan to cover all the necessary technologies?

For the full-stack technologies we plan to utilize in this project, our group made sure to use technologies we have used before in our other Computer Science classes and our own personal projects. This will ensure we complete our milestones as efficiently as possible so we don't have to spend time learning unfamiliar technologies that would hinder our progress. In Milestone 0, we made a list of programming languages, databases and frameworks we will use for the project, such as React, Python and MySQL, all of which we are familiar with to some degree. One concern we do have is that some of us may have forgotten, or could benefit with extra practice on, aspects and concepts of these technologies. In order to ensure we understand and utilize our technologies correctly for the project, our group has provided each other with various resources online, ranging from written documentation to coding bootcamps to even YouTube tutorials. The secret, unofficial technology we also plan to use on the project is this obscure resource called Google, which will help us locate information on how to implement our parts for the project quickly. With the power of Google, we'll be able to quickly refresh our knowledge of our full-stack technologies to create a solid website that both our group and even the general public will appreciate.

- schedule risks
    - Does your team have a team schedule for every member including their detailed task?
    - If change happens, does it update transparently? Does your team use project management tool (e.g. Jira, Trello)

From the beginning of the semester, our group has created schedules for each milestone to effectively plan our tasks. Scrum master, Steve Betts, offered us a schedule on Microsoft Whiteboard where we list our assigned tasks for each milestone for the week, and write down when we have completed them along with our plans for the foreseeable future. We've also written down potential roadblocks ("Blockers") that would hinder our progress on our respective tasks. This way we gain awareness of the time we have to do such tasks and figure out ways to mitigate said risks. Not only will our schedules ensure we're on track to finishing our assigned tasks, but they will also serve us well in our careers in the future, as schedule creation via tools like Microsoft Whiteboard is an important industry practice.

- teamwork risks (any issues related to teamwork);
    - Everybody is on the meeting regularly?
    - Everybody keeps his/her pace? If not, what is your plan to mitigate the risks?
    - legal/content risks (can you obtain content/SW you need legally with proper licensing, copyright).

So far, our group is fine when it comes to teamwork, as we don't have any major risks or issues with our cooperation with each other. We all get along well in terms of group atmosphere, and are dedicated to our work as well. All of our group members do attend the meetings and actively participate, whether together on campus or online. We've met several times in-person outside of class, and on instances where one of our members has another task or can't attend due to commuting issues, we have online meetings on Discord.

For the most part, each of us have stayed on track for our assigned tasks from Milestone 0 to 2. However, we would be dishonest to pretend that we're flawless in terms of time management. One primary issue we have that complicates our project is this giant meteor looming over the sky called homework. As students, we not only have to focus on this class but also balance our time with other classes and outside obligations. Some of us are taking hard classes this semester that we would have to dedicate ourselves to at times in order to pass, such as CSC 415, 600 and 667. If each of us were only taking 648 as our one class this semester, we could probably finish each milestone at least a week before they're due. Alas, we don't have that luxury. So, in order to prevent us from falling behind on our project…

Ideas we could execute in order to prevent us from falling behind on our project…

1. Starting assigned tasks earlier, dedicating at least 1 hour of our day to start on our deadlines
2. Back-end implements their stuff first, then front-end implements their tasks slightly afterwards
3. "Checks and Balances" idea - If a person only completes part of their assigned tasks due to other obligations, another group member or two picks up where they left off, and vice versa. This way we ensure some progress will be made as well.
4. Plan our deadlines a bit earlier than the official due date, so if we end up not completing the milestones on time we can have an extra day or two to finish it
5. (**Not recommended, but ONLY in extreme circumstances**) Pulling an all-nighter session or two to complete a milestone fully.

Finally, our group should be mostly fine with the legality of our project. Perhaps the only concern would be our usage of OpenAI for movie recommendations.

7. Project Management

Each meeting, our scrum lead will have the group members fill out a Microsoft Whiteboard. This Microsoft Whiteboard has a template that allows us to state what we have worked on since we last met, what we plan to work on, and any blockers that we have. This allows each member to transparently update the team on their status within the project. After completing and talking about the whiteboard, the scrum leader creates tickets that are trackable in GitHub Issues.By using GitHub Issues, we are able to compare our Progress Whiteboard and decipher what tasks need extra resources. This allows us to complete our tasks before necessary deadlines and allocate enough time to design, develop, and test.