FlickPick

SENG 401 Final Project Report
Group 25

(https://github.com/Jeff5bn/SENG401_Group25/tree/main)

2024/03/27

Jeffrey Bartel

Allen Qu

Kevin XIE

Raymond Lam

Logan Nightingale

Jiawei He

Josh Froese

Requirement Traceability Matrix:

Requirment id	Requirment description	Functionality	Development	Testing	Notes	priority	Due Date
1	**Frontend** User inputting if selected movie is liked or disliked	Functional	Completed	Test Case 20, 21, 22, and GUI testing	Front-end page was finsihed on Febuary 13	High	Febuary 24
2	**Backend** User inputting if selected movie is liked or disliked	Functional	Completed	Test Case 7, 8, 9, 20, 21, 22	Backend api was finished on Feburay 23	High	Febuary 24
3	**Frontend** User displayed movies that are recommended based on data collected	Functional	Completed	Test Case 11, 12, 13, 14, 15, 23, 24, and GUI testing	Front-end page prototype was finsihed on feburary 23	High	March 4
4	**Backend** User displayed movies that are recommended based on data	Functional	Completed	Test Case 1, 2, 3, 4, 23, 24	Backend api was finsihed on March 3	High	March 4
5	Database of Movies	Functional	Completed	Test Case 11	Was made and pre-processed on Feb 13	Highest	Febuary 14
6	User creation and Login	Functional	Completed	Test Case 16, 17, 18, 19	Front-end had login finished on Mar 05, Backend finsihed on March 18	Mid	March 4, was extended to March 20th
7	password hashing	non-functional	Completed	Test Case 5, 6	Backend finsihed on March 18	Mid	March 20th
8	cloud deployment	non-functional	Terminated	None	Tired using Heroku, suggestion by TA. Had it working by March 21th but no free tier so it was stopped	Mid	March 27th
9	Having Frontend and Backend connected	Functional	Completed	GUI Testing and API Testing	Started and had it working for local prototype on March 3, Updated for new login changes on March 18	High	March 20th
10	Having UI scale to differnet screen sizes	non-functional	In-progress	None	Finsihed on main swiping page on March 25th, Unable to fiinish on recommed movies	Low	March 27th
11	Having swiping controls	non-functional	Completed	None	Had buttons implemented on Feburary 13, Finisihed kinda on main swiping page on March 25th	Low	March 27th

Requirement description and implementation:

1. Frontend User input:

This requirement is the main page of the application. This is where the user is presented with a movie and decides if they like, dislike, or haven't seen it before. For our application, this requirement is very essential as with this data of the user we can then run an algorithm to recommend movies for the user. We are making this a functional requirement of high priority.

This requirement was completed before check-in #1. We used the react library in JavaScript to complete this as well as the rest of the backend. This main page did go through a few different looks and revisions to include all the buttons that are present in the project right now.

2. Backend User input:

This requirement is the APIs needed to handle the requests the user gives to dislike and like different movies, as well as giving a list of movies from the database to present to the user. Like requirement 1 it is very essential as with this data of the user we can then run an algorithm to recommend movies for the user. Making this a functional requirement of high priority.

This requirement was finished before check-in #2. Our backend uses a Django Python framework. For the URLs in this requirement (popular_movies, like-movie, and dislike-movie) descriptions of what they expect and return can be seen in the API link descriptions.pdf.

3. Frontend User recommender:

This requirement is the recommendations page for the application. After we have gathered the data on a user's liked and disliked movies, we need a place for what movies we recommend the user should watch based on their preference. This requirement is also essential to the application because it can be seen as the 2nd part of what our project does. Thus making it functional and a high priority.

This requirement was finished before check-in 2. Just like the main page, this recommender page was also revised to include all the buttons and features that are present in the project right now.

4. Backend User recommender:

This requirement is the algorithm that recommends movies as well as the API calls to get that information. The algorithm we used is based on using the genres of the liked and disliked movies to create weights for each one. Then we input a new movie with its genres and the formula will give us a score. The API call does this with all the movies and then orders them based on the score, sending the top 20 movies for the front end to handle. This is a functional and high-priority requirement.

This requirement was finished before check-in 2. Look up API call recommend-movies to see what it should return. This feature was updated to remove movies already seen by the individual being returned in the response.

5. Database of movies:

This requirement is the creation of a database that has movies with genres and other details. We prioritized highly as it was necessary for both the data collection and recommendation parts of the application.

This requirement was finished before check-in 1. We used data from TMDB (The Movie Database). The Movie Database (TMDB) (themoviedb.org)

6. User login and creation:

This requirement is the management of users for logging into and creating. This would allow multiple different accounts to be in use for the application. This was a medium priority as it wasn't as important as requirements 1,2,3,4. However, this was also a functional requirement.

This requirement was completed before check-in 3. We planned to have it finished before check-in 2, however, that didn't happen. The backend APIs created were create-user and log-in.

7. Password Hashing:

This requirement is having passwords for users stored hashed in the database. This is a need for security concerns but that makes it a non-functional requirement. Since it wasn't functioning this requirement was given a medium priority.

This requirement was finished before check-in 3.

8. Cloud deployment:

This requirement is the deployment of the application to the cloud. The professor made this seem optional but encouraged me to do this requirement. This was non-functional to how the application works itself but it makes it a lot more accessible. We gave this requirement a medium priority as it was non-functional.

This requirement still needs to be completed in time. After check-in 3 we asked the TA for advice and were recommended for the use of Heroku (https://www.heroku.com/). Heroku has changed the policy and got rid of the free tier for cloud deployment, making it not appealing to use. We did get it running on Heroku deployment but didn't want to continue spending to keep it running so we closed it.

9. Connecting Frontend and Backend:

This requirement is connecting the frontend with the backend API URLs. This was a requirement as having separate frontend and backend allowed two different groups of developers to work independently. However after both groups had developed things, additional work was required to connect the two. Making this requirement functional and a high priority for the project.

This requirement was finished before check-in 2. We anticipated that we would finish this before check-in 3. This was done to test and show what was working in the check-ins sooner.

10. GUI scaling:

This requirement is the front end being able to display on more narrow screens like a phone. This kept in mind the more Tinder phone application mindset we had for our project. Since this is just for the assets of the front end, it is a non-functional requirement with a low priority since we demo the project on a computer.

This requirement was half completed. For the main page, it should be implemented but for the recommendations page, we did not get it completed. This was implemented in the CSS of the pages.

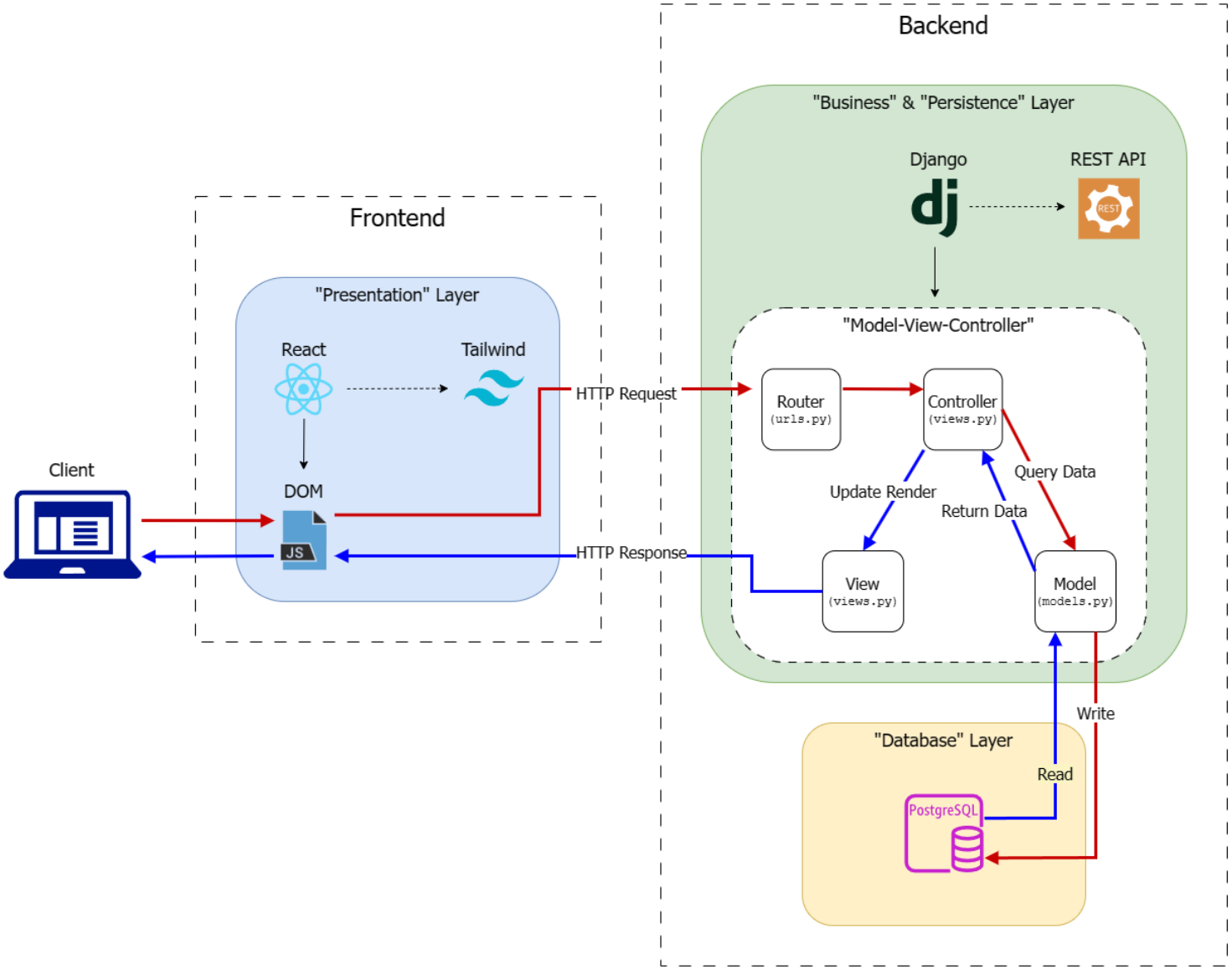
11. Swipe controls:

This requirement is the frontend controls allowing for swiping for liking or disliking a movie. This also kept in mind the phone application mindset we had for our project. Since it just controls it is a non-functional requirement and we gave it a low priority since we demoed the project on a computer.

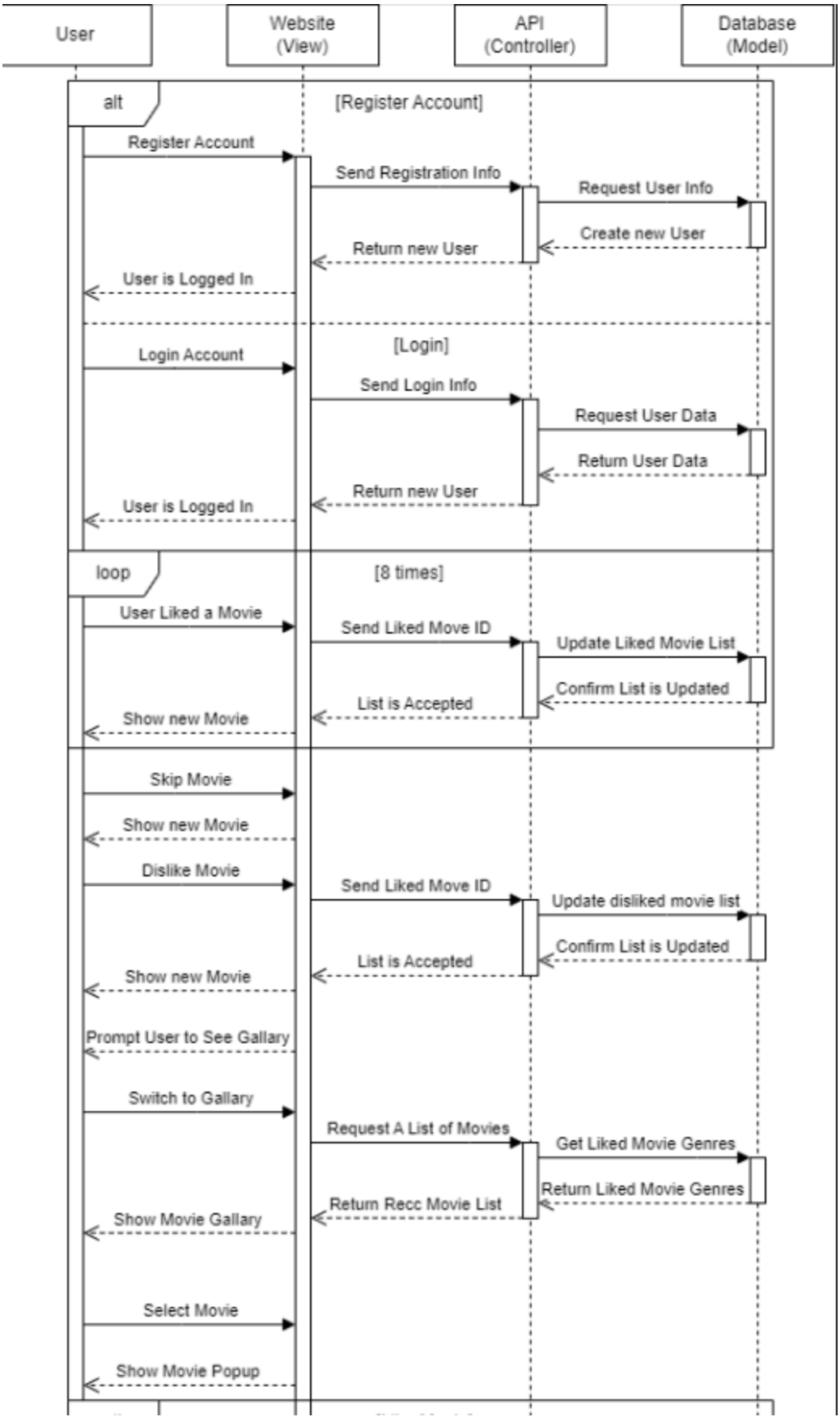
This requirement was finished after check-in 3. It is shown in the video demo. This was done in MovieCards.js with some functions to track mouse movement while holding down a click.

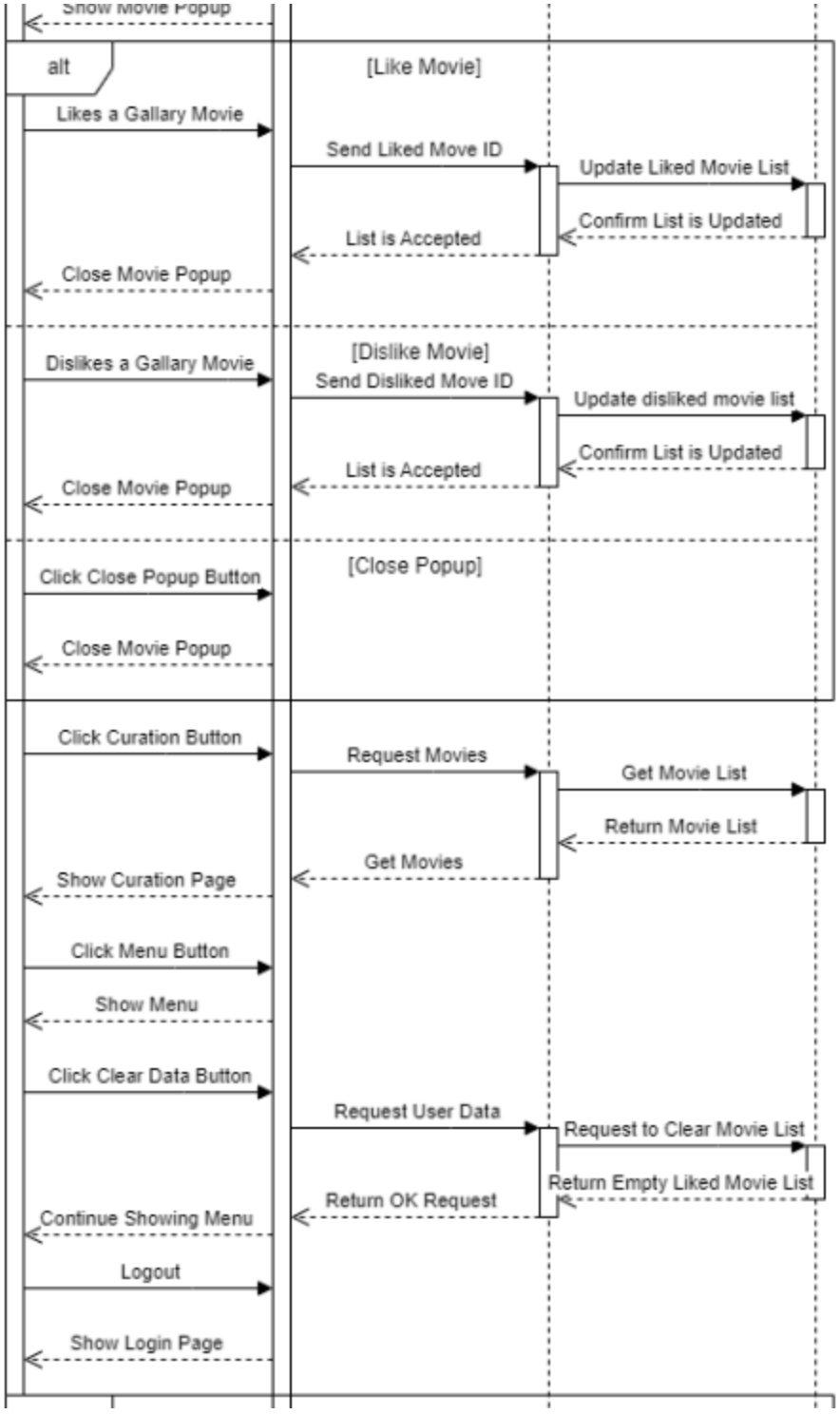
Diagrams:

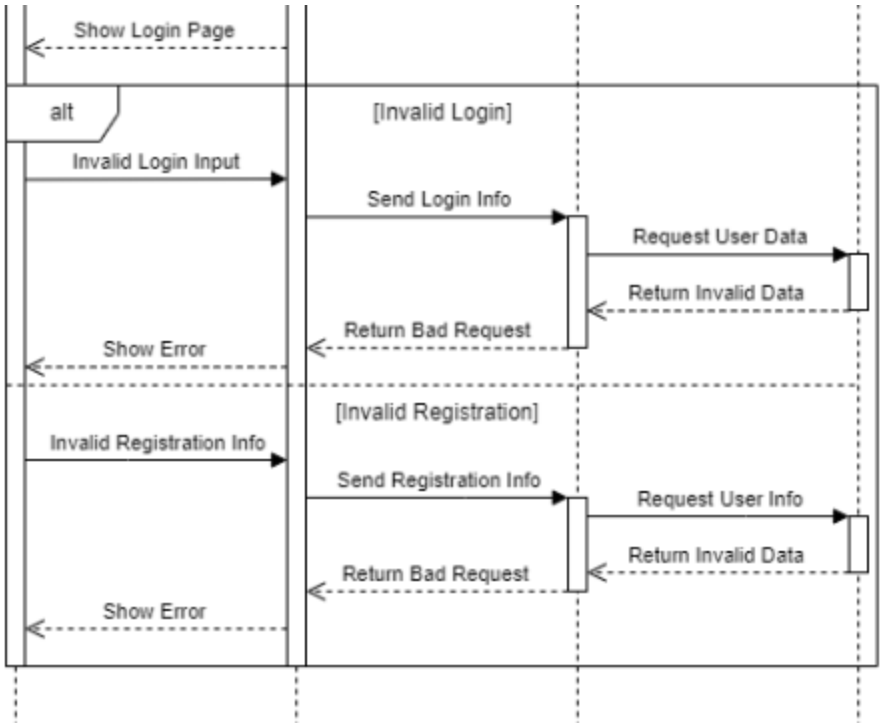
System architecture:



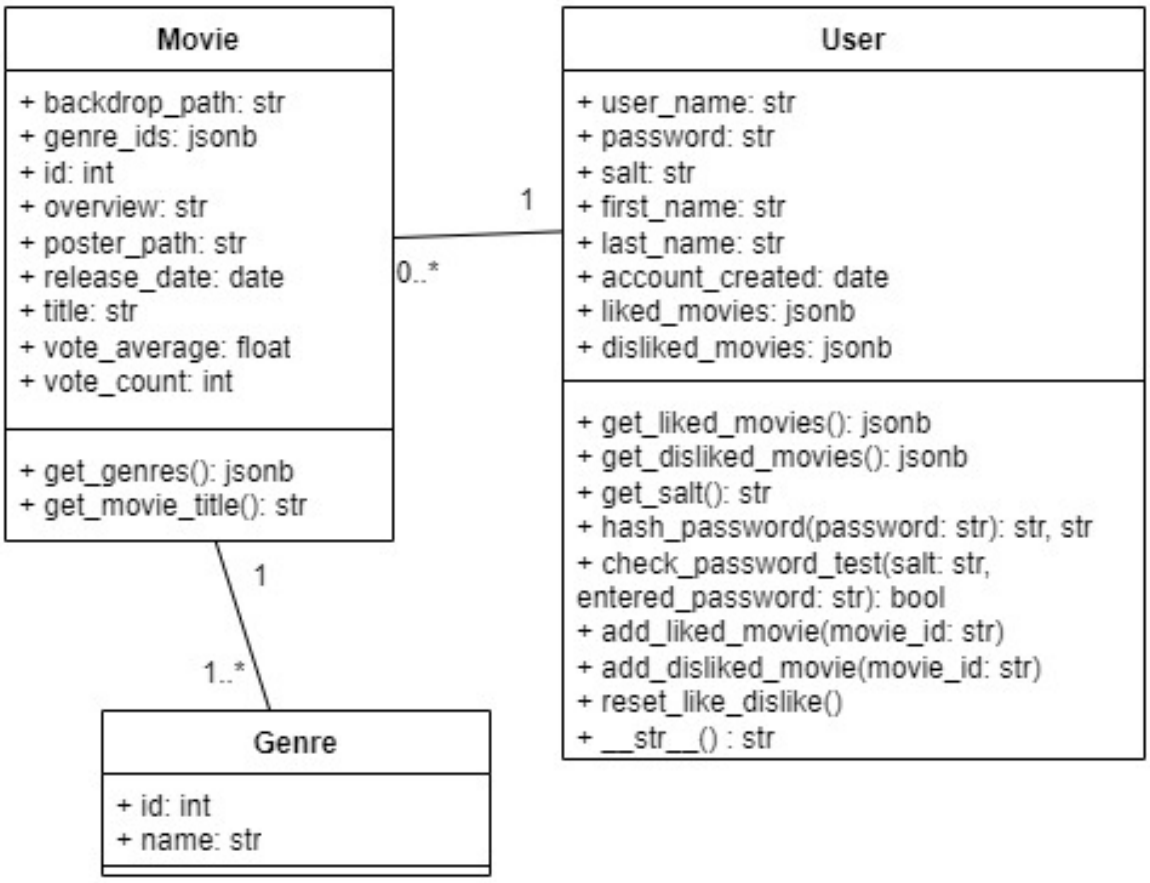
Sequence:


User
Website (View)
API (Controller)
Database (Model)
alt
[Register Account]
Register Account
Send Registration Info
Request User Info
Create new User
Return new User
User is Logged In
[Login]
Login Account
Send Login Info
Request User Data
Return User Data
Return new User
User is Logged In
loop
[8 times]
User Liked a Movie
Send Liked Move ID
Update Liked Movie List
Confirm List is Updated
List is Accepted
Show new Movie
Skip Movie
Show new Movie
Dislike Movie
Send Liked Move ID
Update disliked movie list
Confirm List is Updated
List is Accepted
Show new Movie
Prompt User to See Gallary
Switch to Gallary
Request A List of Movies
Get Liked Movie Genres
Return Liked Movie Genres
Return Recc Movie List
Show Movie Gallary
Select Movie
Show Movie Popup


Show Movie Popup
alt
[Like Movie]
Likes a Gallary Movie
Send Liked Move ID
Update Liked Movie List
Confirm List is Updated
List is Accepted
Close Movie Popup
[Dislike Movie]
Dislikes a Gallary Movie
Send Disliked Move ID
Update disliked movie list
Confirm List is Updated
List is Accepted
Close Movie Popup
[Close Popup]
Click Close Popup Button
Close Movie Popup
Click Curation Button
Request Movies
Get Movie List
Return Movie List
Get Movies
Show Curation Page
Click Menu Button
Show Menu
Click Clear Data Button
Request User Data
Request to Clear Movie List
Return Empty Liked Movie List
Return OK Request
Continue Showing Menu
Logout
Show Login Page


Show Login Page
alt
[Invalid Login]
Invalid Login Input
Send Login Info
Request User Data
Return Invalid Data
Return Bad Request
Show Error
[Invalid Registration]
Invalid Registration Info
Send Registration Info
Request User Info
Return Invalid Data
Return Bad Request
Show Error

Class:


Movie
+ backdrop_path: str
+ genre_ids: jsonb
+ id: int
+ overview: str
+ poster_path: str
+ release_date: date
+ title: str
+ vote_average: float
+ vote_count: int
+ get_genres(): jsonb
+ get_movie_title(): str
User
+ user_name: str
+ password: str
+ salt: str
+ first_name: str
+ last_name: str
+ account_created: date
+ liked_movies: jsonb
+ disliked_movies: jsonb
+ get_liked_movies(): jsonb
+ get_disliked_movies(): jsonb
+ get_salt(): str
+ hash_password(password: str): str, str
+ check_password_test(salt: str, entered_password: str): bool
+ add_liked_movie(movie_id: str)
+ add_disliked_movie(movie_id: str)
+ reset_like_dislike()
+ __str__() : str
1
0..*
1
1..*
Genre
+ id: int
+ name: str