MovieMaster Project Report

Names:

Sopan Nair: sopannair

Aiden Devine: aide8765

William Hardee: WilliamHardeeIsCool

Sameer Kulkarni: sameerk777

Kayhaan Rashiq: KayhaanR

Nicholas Kopiwoda: Achoowy

Project Description:

MovieMaster is a centralized platform for movie lovers who want a one-stop-shop for finding their next Movie. This website pulls ratings and reviews from credible movie review pages such as IMDb, TMDB, and MetaCritic. It allows users to add ratings, reviews, “Like” movies, and use our ingenious proprietary features. Users must register an account and log in to gain access to our suite of innovative features. For instance, the “For You” page uses “Liked movies" to compute movie suggestions using MovieMaster’s state-of-the-art Recommendation Model. Making recommendations tailored to a user’s individual tastes was a core goal for the MovieMaster website. The “Flix” page allows MovieMaster users to watch an uninterrupted stream of hot movie trailers with friends and family, to help decide what movie to watch together. This feature was inspired by popular platforms such as Tik Tok, Instagram Reels, and Youtube Shorts. We wanted to deliver an addictive scrolling experience unique to MovieMaster through the “Flix” page. We believe this strategy will help garner a younger audience who is fond of the “scrolling” media consumption format. Our overall mission with MovieMaster is to provide movie buffs with one single platform to search for, review, discover, and rate movies with their friends and family on a movie night.

Project Tracker: Github Project Board

Video: Youtube Demo

VSC (Link to GitHub Repo): GitHub Repository

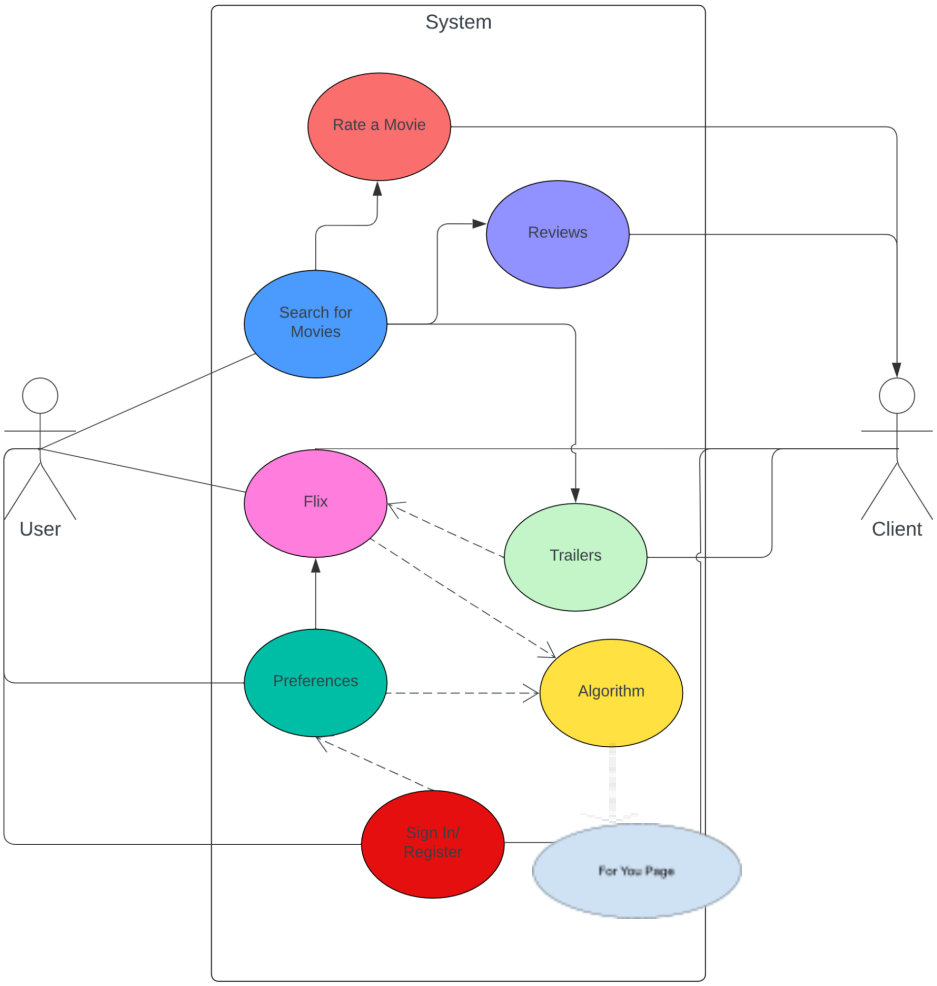
Contributions:

- Aiden:
    - Full wireframe through Figma
    - Full implementation of Flix page and trailers from TMDB:
        - Initially implemented with embedded YouTube playlist
            - Did not allow for movie information without YouTube API
        - Final version used OMDB to pull movie information (mainly implemented by William) and TMDB to pull YouTube trailer links attached to each movie (my addition)
            - Associated videos per movie were pulled using a second TMDB call and then filtered through a confirmation of video.site being 'YouTube' and video.type being 'Trailer' in order to only insert YouTube trailers into the movies table of DB
        - Movie info pulled through SQL query in GET function for /flix and altered so that stringified data works with single and double quotes
        - Implemented change trailer button which chooses a random video from parsed flix_data, pulls its info, and displays trailer, name, description, year, and director on the page (function is called upon page start)
            - Calls changeTrailer again if no trailer included for a movie
        - Implemented 'Like' feature into flix page: adds movie to liked movies and redirects to flix page, which displays a new trailer
    - Link on register page to login page if already have account
    - Some occasional tweaks with redirect routes and css styling
    - Release notes 4-5, 4-19
- Sopan:
    - Wrote documentation for app description
    - Wrote function to call OMDB API with access key
    - Wrote initial fetchMovieData function to insert movie data from API into SQL DB
    - Modified SQL DB fields to correspond with OMDB fields
    - Worked with Kayhaan to render top ranked movies dynamically on Home Page carousel
    - Worked with Sameer to implement Search Bar functionality with drop-down suggestions
    - Release notes 4-12
- William:

        - Implemented function that pulls from both TMDB and OMDB to populate the database with movies
        - Implemented function that pulled from TMDB to populate the external reviews
        - Mapped movies to genres
        - Implemented login, register, and logout pages hashing the passwords and doing appropriate error checking
        - Implemented the movies details page
        - Implemented like and unlike express routes as well as the route that adds a review
- Sameer:
    - Implemented entire profile page and all of its features
        - Select and change avatar feature
        - Displaying user's reviews as well as liked movies as they are updated
    - Worked with Sopan to implement the search bar, and I added some convenience features to it (enter and escape key functionality)
    - Styling across the website
    - Made and styled the navigation bar
- Kayhaan:
    - Designed the UI and general style of the home page along with color scheme of the website
    - The homepage includes:
        - A couple carousels in order of most watched movies every week
        - The navbar designed by Sameer and Sopan
        - The Featured Movie section displaying the most watched movie across all the APIs we pulled
    - Worked with Sopan to render top ranked movies from the APIs and input them into the Home Page carousels in the order of most popular movies
    - General website styling
- Nicholas:
    - Designed our project db and wrote create.sql (db init code)
    - Recommending Movies:
        - functions to quantify the likelihood a specific user will like a movie based on feature similarity with their previously liked movies.
        - function to pull N top recommended movies for a specific user, also including the probability the user will like the movie.

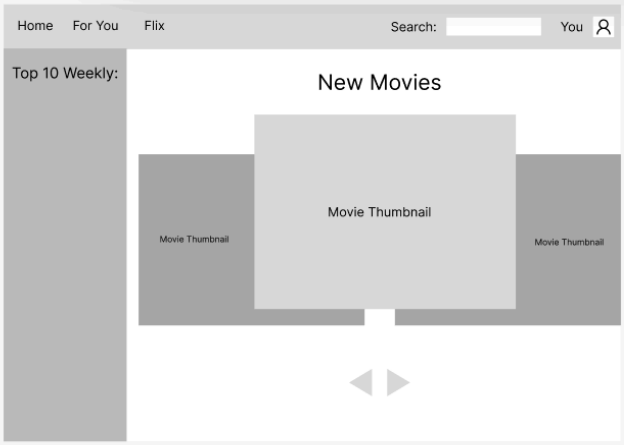
    - API call to pull a user's top N recommended movies to test functionality.
    - (Function usage is clearly defined in predictScript.js for your convenience).
- ‘For You’ Page:
    - App route to render the ‘For You’ page with movies or a prompt for the user to like movies before recommendations can be made.
    - Page includes top 20 recommended movies for a user.
    - Each movie is displayed with the movie image, name, and probability of the user liking it.
    - Clicking on the movie card takes the user to the movie details page.
- I also made at least two jokes, thus improving team morale.

Use Case Diagram:

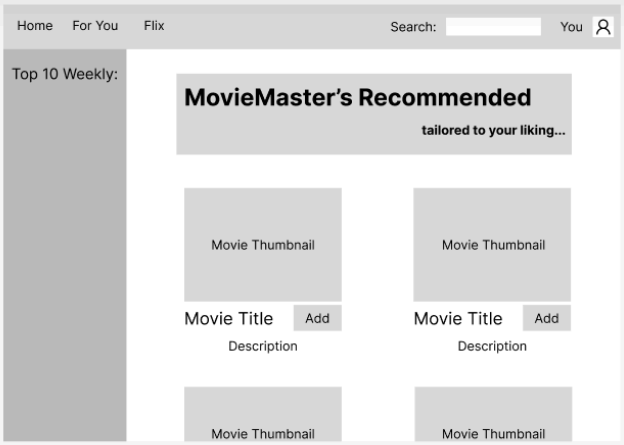


Wireframes: [Figma]

Wireframe - 1

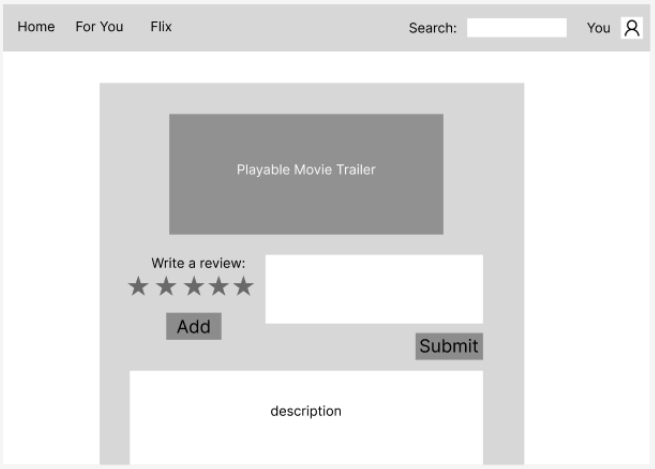


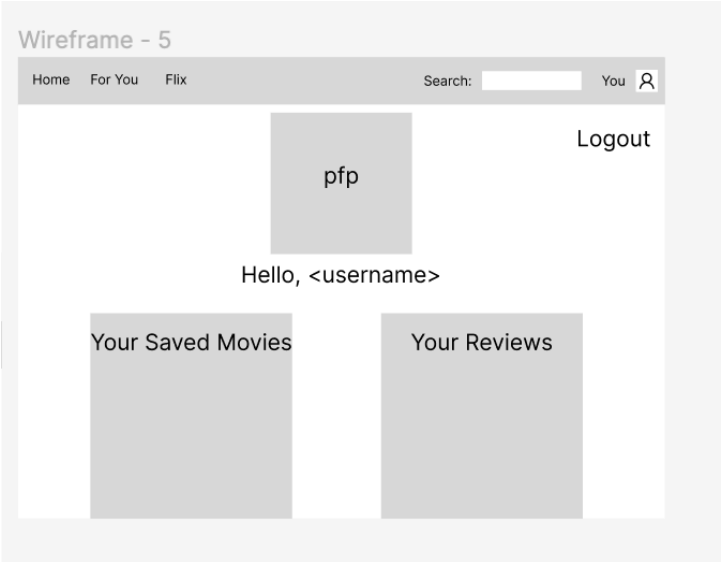
Wireframe - 2



Wireframe - 3

Wireframe - 4





Test Results (Lab 11):

1. **UAT for Login:**
    a. **Test 1 - User was instructed to type in credentials “test” and “password” on the login form**
        i. **User took little to no time to find login form as it is front and center**
        ii. **User typed “test” correctly but “password” incorrectly, prompting a redirect to login page and a message of “Wrong Password”**
        iii. **User then typed correct credentials and was redirected to the home page, which displayed correctly**
        iv. **User then clicked logout on top right of navbar, this took a couple seconds for the user to find**
    b. **Test 2 - User was instructed to type in wrong credentials for login**
        i. **Due to “Test 1,” user completed test quickly and efficiently**
        ii. **User typed “test” and garbage text for the password, prompting a redirect to Login and a message of “Wrong Password,” as expected**
    c. **Test 3 - User was instructed to type in garbage values for username and password fields**
        i. **User typed “username” and “12345” in the fields, prompting a redirect to Login and a message of “Username Not Found”**
    d. **All test results were same as expected**
2. **UAT for Register:**

a. **Test 1 - User was instructed to type the credentials used earlier for Login test**
    i. **User easily navigated to Register page from Login page through "No account: Register" at the bottom of the Login form**
    ii. **User typed "test" and "password" in the register fields and submitted, prompting a redirect to the register page and a message of "Username already exists," as expected**
b. **Test 2 - User was instructed to enter new valid credentials for a new user account**
    i. **Already on the register page, user typed values "username" and "12345" before submitting, then being redirected to the login page**
    ii. **User then typed same information into the login form before submitting, which then redirected them to the home page, as expected**
c. **Test 3 - User was instructed to input nothing to the username and password fields, which should prompt a message**
    i. **Having logged in, the user logged out, was redirected to the login page, and after navigating to the register page, clicked the submit button without typing in either field**
    ii. **User was redirected to Register and a message was displayed: "Username is too short"**
    iii. **User was then asked to enter a username of 3 characters with a valid password, to which they were redirected to the Register page and a message was displayed: "Username is too short"**
    iv. **User was then asked to enter a valid username but 3 character password, to which they were redirected to the Register page and a message was displayed: "Password is too short"**
d. **All test results were same as expected**

3. **UAT for likedMovies - User was instructed to login with above credentials, choose a movie on the home page, like the movie, and view it in the user page**
    a. **Login process was smooth as user already knew credentials**
    b. **Once on home page, user clicked large red "Check Out" button front and center next to *The Godfather Part II* thumbnail, which navigated them to the movie's page**
    c. **User then read through the information displayed about the movie before seeing the 'Like' button displayed in the right corner of the information box**
    d. **User then clicked like button, which redirected them to the same movie page but with the 'Like' button now displaying as 'Unlike'**

e. **When trying to navigate to user page, user accidentally clicked Logout button, which then required them to login again**
    i. **Future scope: logout confirmation form**
f. **Once on user page, user first selected one of the five avatars and clicked "Change Avatar" before looking to the "Your Liked Movies:" section and clicking *The Godfather Part II***
g. **User was redirected to the movie page again, to which they then clicked 'Unlike'**
h. **User then navigated back to the user page (correctly this time) and the liked movies list was now blank, showing a successful test**

4. **UAT for For You - User was instructed to like a handful of movies, navigate to "For You" page, and like a recommended movie**
    a. **User was already logged in and therefore clicked the MOVIEMASTER title on the navbar, which redirected them to the home page**
        i. **Glad to see this is an intuitive act for users to click the website title to navigate back to home**
    b. **User liked movies *Dune: Part Two, Avengers: Endgame,* and *Pulp Fiction***
        i. **After liking first movie, user clicked back arrow on the browser, which redirected back to the movie page due to the 'Like' function redirecting upon clicking**
            1. **Future scope: implement dynamic button change for 'Like'**
    c. **User then clicked "For You" button on navbar, redirecting them to the "For You" page**
    d. **User chose the third movie listed, *The Shawshank Redemption*, which had a 68% recommendation score**
    e. **User then liked the movie and clicked the "You" button, navigating back to the user page and showing all four movies in the liked movies list**
    f. **Test Success**

Deployment:

1. cd ProjectSourceCode
2. Docker compose up -d
3. Open http://localhost:3000/