Chau Lucky Nguyen | A10742893 • Kenneth Vu | A08680828 • Ryan Weng | A10713613 // CSE 134B

HW 5: Final Execution and Optimizations

GITHUB REPO & WEBSITE URL

Github: https://github.com/ryanweng24/cse134b
Website: https://moviedex-99d1a.firebaseapp.com/
README: README.md in HW5 directory of the github repository

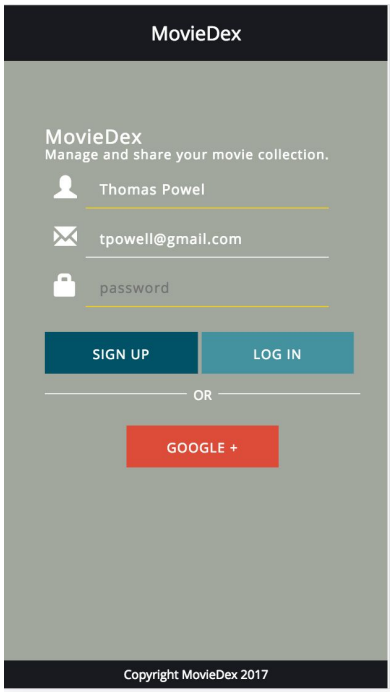
APPLICATION DESCRIPTION & FLOW

MovieDex is a simple web application that aids movie-loving individuals in collecting and sharing their favorite movies. Users are able to access the application directly online at: https://moviedex-99d1a.firebaseapp.com/

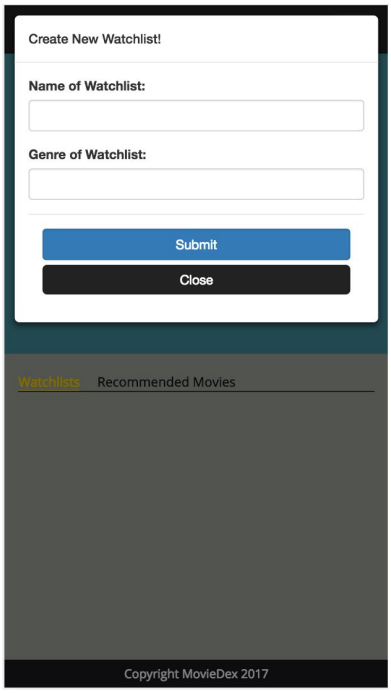
Specific Features that MovieDex offers:

- Users can sign up for an account by inputting their username, email, and password. Users can log in with a Google account through Google authentication or they can log in if they already signed up for an account.
- Once users log in, they are redirected to their profile page. Users can click on the "Create New Watchlist" button, which will display a modal for users to specify the name and genre of their watchlist. The user's profile is populated with all the watchlists that the user created. Users can edit and delete watchlists from their profile page similarly.
- Users can add their favorite movies to their watchlists by selecting from a set of pre-stored movies. Users can click on "Add Movie" on a watchlist, which will display a modal for users to choose which movie to add from the dropdown with pre-stored movies. From there, MovieDex will automatically store and display the poster for that movie.

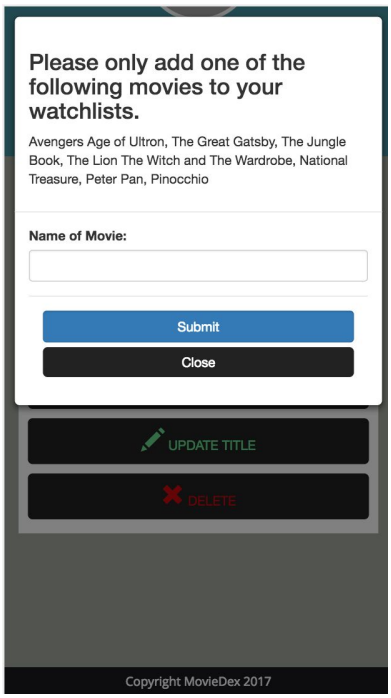
DEMO



Log in



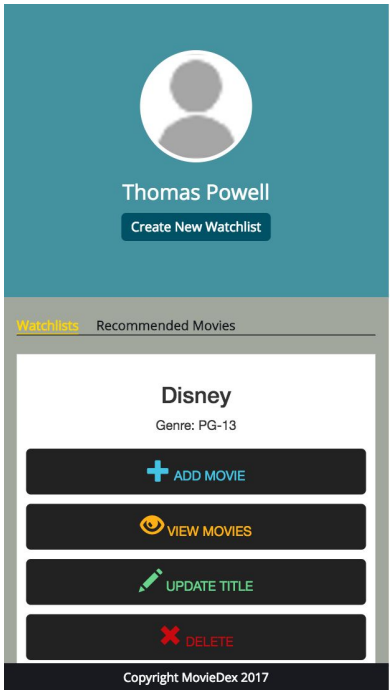
Create watchlist



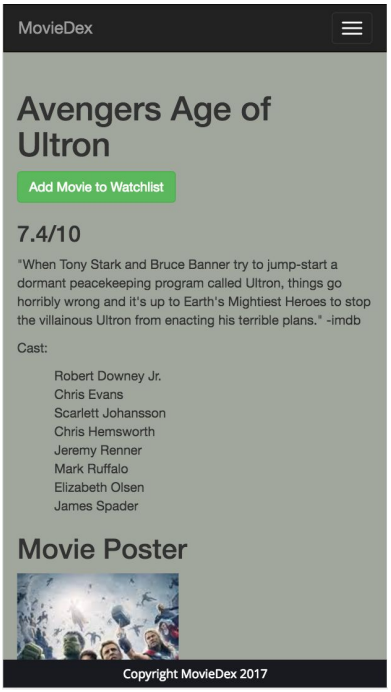
Add movie

- Users can view what movies are in their watchlist by clicking on “View Movies” on a watchlist, which will display all the posters of the movies in the watchlist.
- Users can edit the watchlist by clicking on “Update Title” on a watchlist, which will display a modal for the user to change the name.
- Users can delete a watchlist by clicking on “Delete.”

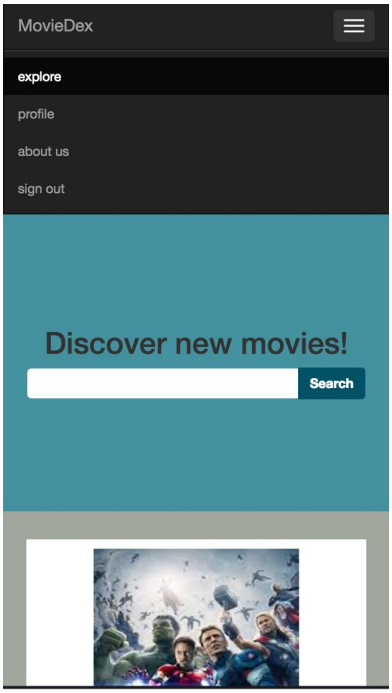
DEMO



Delete



See Movie Description



Basic Navigation

- Users can navigate to the explore page by clicking on the “explore” button on the navigation bar, where they can view new movies as well as search for movies (unfortunately we do not have a database full of movies to search through).
- On the explore page, users can click on any of the movies to take them to a new page with full details about the movie such as rating, synopsis, and cast.
- On each of the movie description pages, users can click on the “Add Movie to Watchlist” button, which will display all the watchlists that the user has so that the user can choose which watchlist to the add the movie to (not implemented).
- Users can navigate to our team page by clicking on the “about us” button on the navigation bar, where they can learn more about our team.
- Users can sign out by clicking on the “sign out” button on the navigation bar.

CODE ARCHITECTURE

We used Firebase as our backend data storage to implement the CRUD features of our application. Though we did not have a choice in this, our team did develop our own strategy in how we would use Firebase. We chose to store information about all users, all watchlists, current user, and current user’s watchlists on Firebase, so that we would be able to display the correct data for a specific user.

We chose to use Bootstrap as our CSS framework rather than using vanilla CSS or another framework because two out of our three teammates have had previous experience with Bootstrap and found the framework to be very convenient to use especially when making the site mobile responsive. Even though Bootstrap is a bloated framework, we believed that the tradeoff was worth the ease in building an application in a limited amount of time.

We used Vue.js and JQuery over plain vanilla JavaScript with goals to make accessing data easier due to the defined methods in Vue.js and JQuery. Vue.js encapsulates the DOM allowing for easier to read and less complex code as compared vanilla JavaScript. We use Vue.js primarily for our CRUD implementations. We used JQuery to quickly select and change HTML elements because the functions provided in JQuery resulted in an easier learning curve for the entire team. Similarly to Bootstrap, even though JQuery may be bloated, it was easier for us to quickly access elements and use pre-built functions to save time.

The JavaScript and CSS are separated into different files for reusability and efficiency purposes. Very often, HTML files will use the same CSS and JavaScript code segments. The separation of the files allows for the JavaScript and CSS files to be stored locally within the cache of the user's computer in order to reduce the loading time of each page. Separating the files also promotes reusability as each html file can refer to the styling and script page they need via the use of tags.

With our current architecture, the movie selection on the explore page is hard-coded due to a challenge we faced when attempting to add the watchlist creating feature. Each movie currently has its own HTML file as we currently do not know how to create a dynamic movie page accounting for all movies. Implementing such a feature would require the use of a template framework such as Handlebars, but again we were unable to complete this due to time constraints.

DEVELOPMENT DIFFICULTIES

We had some trouble in meeting our expectations in designing the functionalities of the application that we originally seeked from HW1, such as allowing users to search for movies, add individual movies to their personal watchlist, and deleting movies from a watchlist.

The team members found Vue to have a difficult learning curve, and our members often asked on Slack and went to office hours for help from tutors. Because of this, it was difficult for us to implement all the functionalities and decided to focus on the main functionality of creating watchlists and adding movies to that specific watchlist.

The most challenging issue, besides not knowing Vue too well, was correlating watchlist to their users and adding movies to watchlists. As of right now, when a movie is added to a watchlist, the movie gets added to all watchlists pertaining to that user. Although we looped through the entire user's movies, it was difficult to grab the specific key (stored in Vue) that was associated with the watchlist. We believe it was either due to Bootstrap's modal that was not being dynamically created on each time the user clicked on "Add Movie" or our lack of knowledge of Firebase. So instead the modal was only created on the first click.

In our original specification, we also wanted to implement a feature for users to view each other's watchlists and like them to store in their own collection of watchlists. However, this feature proved to be difficult to implement as we were unfamiliar with working with the Firebase JSON database as opposed to working with a SQL-esque database.

KNOWN BUGS & FUTURE GOALS

Due to the limitations of the course, we were unable to complete all the functionalities that we set out to implement in HW1. The following represents the known bugs in our application as well as the features that still need to be implemented.

Bugs

- Adding a movie to a watchlist will add that movie to all watchlists that belong to the user rather than just that specific watchlist

Future Goals

- Implement "Add Movie to Watchlist" button on the movie description pages so that you can add one movie to any watchlist
- Implement the search bar on the explore page so that users can search and choose any movie in the database
- Implement more read features for the watchlists (can view title of all movies in watchlist and have watchlists have their own page and description)
- Implement more update features for the watchlists (can add any number of movies by title and can update genre)
- Implement functionality for users to search and like watchlists from other users.

- Broaden the number of movies that can be added to user's watchlists

USER IMPACT

***We used a Nexus 5, with a throttling of regular 3G.**

File Name	HW4 load time	HW4 byte count	HW5 load time	HW5 byte count
index.html	1.77 s	250 KB	2.33s	290KB
explore.html	1.86 s	250 KB	2.63s	363KB
profile.html	5.13 s	779 KB	3.44s	385KB
movie.html (includes files such as narniaMovie.html, nationalTreasureMovie.html, etc)	1.31 s	116 KB	2.4 - 3.0s	~260KB
watchlist.html	1.80 s	225 KB	x	x
team.html	1.41 s	164 KB	2.27s	264KB