Computer Science and Engineering
COMP3900 Computer Science Project, 22T2

Movie Monster: Final Report

Name	zID	Role	UNSW Email
Damon (Shengyue Guan)	z5285984	Scrum Master	z5285984@ad.unsw.edu.au
Tam (Khiet Tam Nguyen)	z5313514	Frontend Developer	z5313514@ad.unsw.edu.au
Owen (Xunbo Su)	z5285996	Frontend Developer	z5285996@ad.unsw.edu.au
Joyce (Zhaoyan Liu)	z5271698	Backend Developer	z5271698@ad.unsw.edu.au
Matt (Feng Ji)	z5290365	Backend Developer	z5290365@ad.unsw.edu.au

Submission Date:

Friday 05.08.22

CONTENTS

1. Overview **3**
- 1.1. System Architecture 3
  - 1.1.1. Presentation Layer 3
  - 1.1.2 Data Layer 4
  - 1.1.3. Business Layer 5
- 1.2. Page Structure 6
- 1.3. Project Objectives 7
  - 1.3.1 User Stories 7

2. Functionality Details **12**
- 2.1 Login Page 12
- 2.2 Register Page 13
- 2.3 Movies Page 14
- 2.4 Movie Details Page 15
- 2.5 User Profile 17
- 2.6 Home Page 19
- 2.7 Admin Page 20
- 2.8 Blog pages 21
- 2.9 Quiz Page (Novel) 23

3. Third-party functionalities **25**
- 3.1 Frontend 25
  - 3.1.1. Theme 25
  - 3.1.3. State Management 25
  - 3.1.4. Components 25
- 3.2 Backend 25
  - 3.2.1 Spring Boot 25
  - 3.2.2 Python libraries 26
    - 3.2.2.1 Pandas 26
    - 3.2.2.2 Scikit-learn 26
    - 3.2.2.3 Pymongo 26
    - 3.2.2.4 Alive-progress 26
  - 3.2.3 Maven 26
    - 3.2.3.1 Spring framework 26
    - 3.2.3.2 Projectlombok 26
    - 3.3.3.3 Jsonwebtoken 26
  - 3.2.4 MongoDB 27
- 3.3. IMDb API 27

4. Implementation Challenges **27**
- 4.1 Frontend 27
- 4.2 Backend 27

4.2.1 Recommendation Challenges 27
4.2.2 Configuration Challenges 28
4.2.3 Transferring data from IMDb to Mongodb database 28

5. User Manual 29
5.1. Starting the Backend: 29
5.2. Starting the Frontend: 29

5.3. Deployed Application (Cloud Hosting) 30

6. Reference 31

1. Overview

1.1. System Architecture

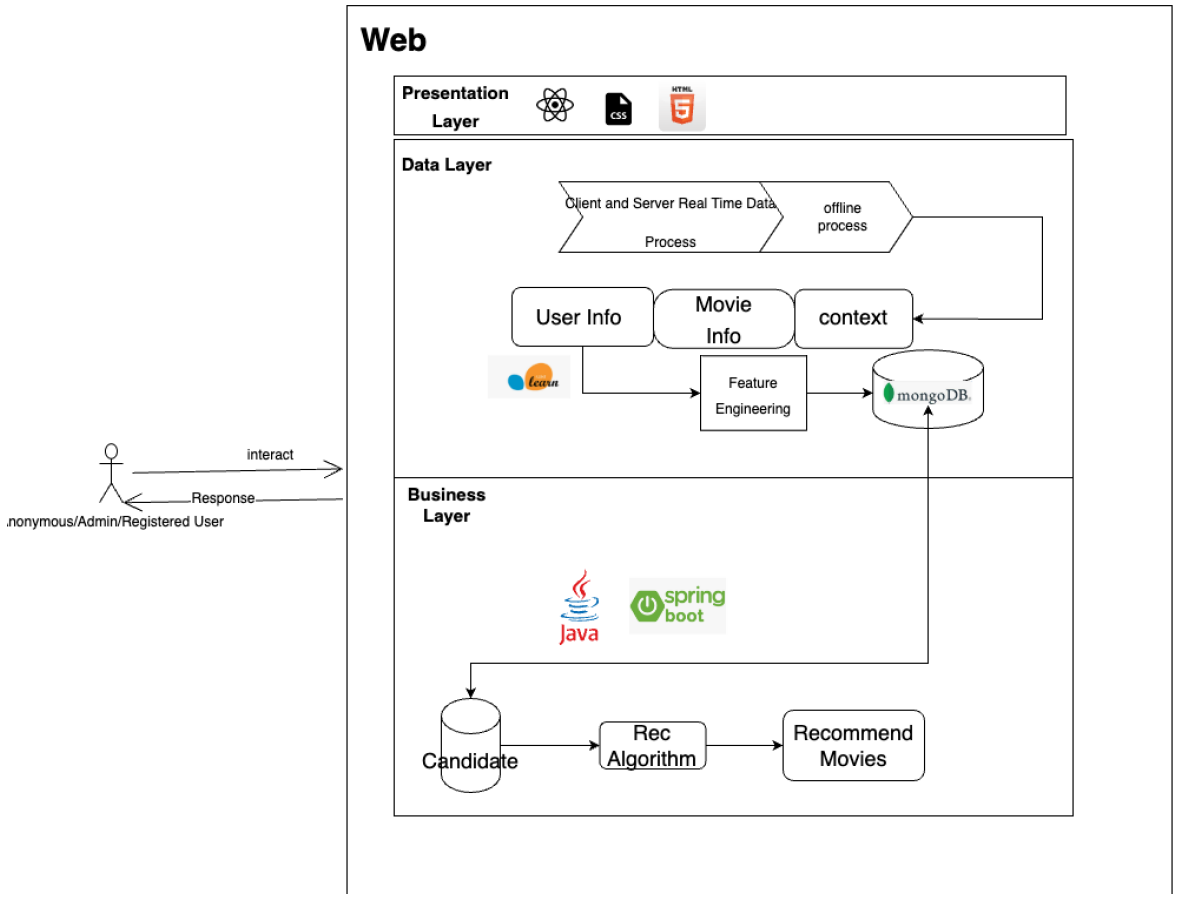


Figure 1.1.0: Layers summary

1.1.1. Presentation Layer

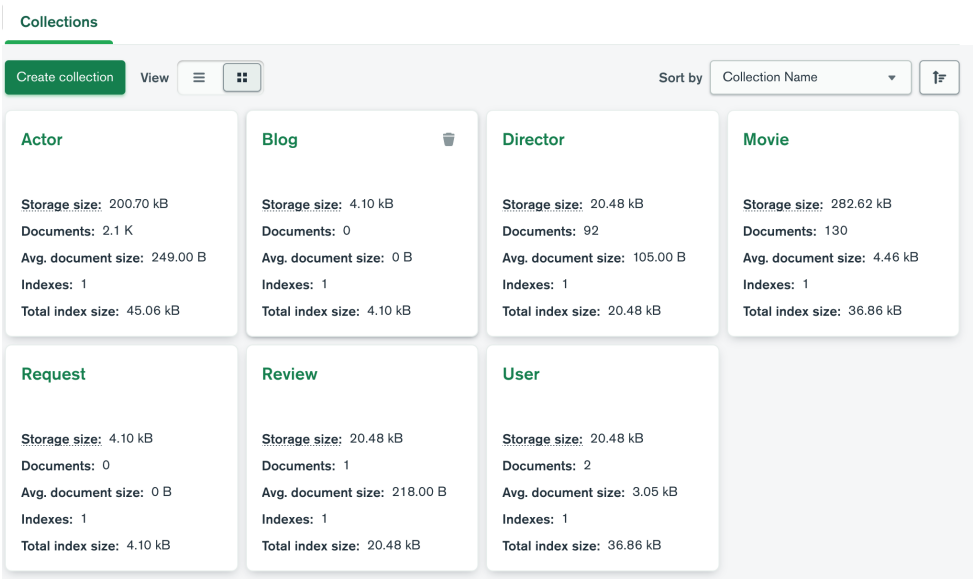
Our frontend is built using the ReactJS framework - a free and open-source library for creating user interfaces through UI components. ReactJS is one framework that enables our website as a Single Page Application, which loads the majority of resources such as HTML, CSS and JavaScript at the beginning of our application's lifespan, which provides a quick and responsive user experience after the initial page load.

We have also chosen to utilise TypeScript, a superset of typed JavaScript to better manage our project while enabling static verification through strong static typing. This results in more robust software and allows for quicker development resulting from less time spent on debugging type issues.

Finally, our React components are styled and themed with Material UI and Bootstrap, both of which provide access to subcomponents that are visually appealing and responsive on both mobiles and desktop sites. A full list of all other frontend packages that were leveraged is contained in the frontend directory's package.json.

1.1.2 Data Layer

MongoDB is our chosen data storage in the backend. Unlike SQL, it is a non-relational document database with JSON-like storage for different collections (tables), which is simpler and more intuitive to manage.



As shown in **Figure 1.1.2.1**, there are 7 collections in our database. With the exception of movies, objects in these collections are assigned globally unique identification strings using UUIDs which can be linked together by MongoDB. As for movies, the identification string is kept the same as IMDb, an online database with information related to films and movies whose API is heavily leveraged in our application for the addition of movies to our database.

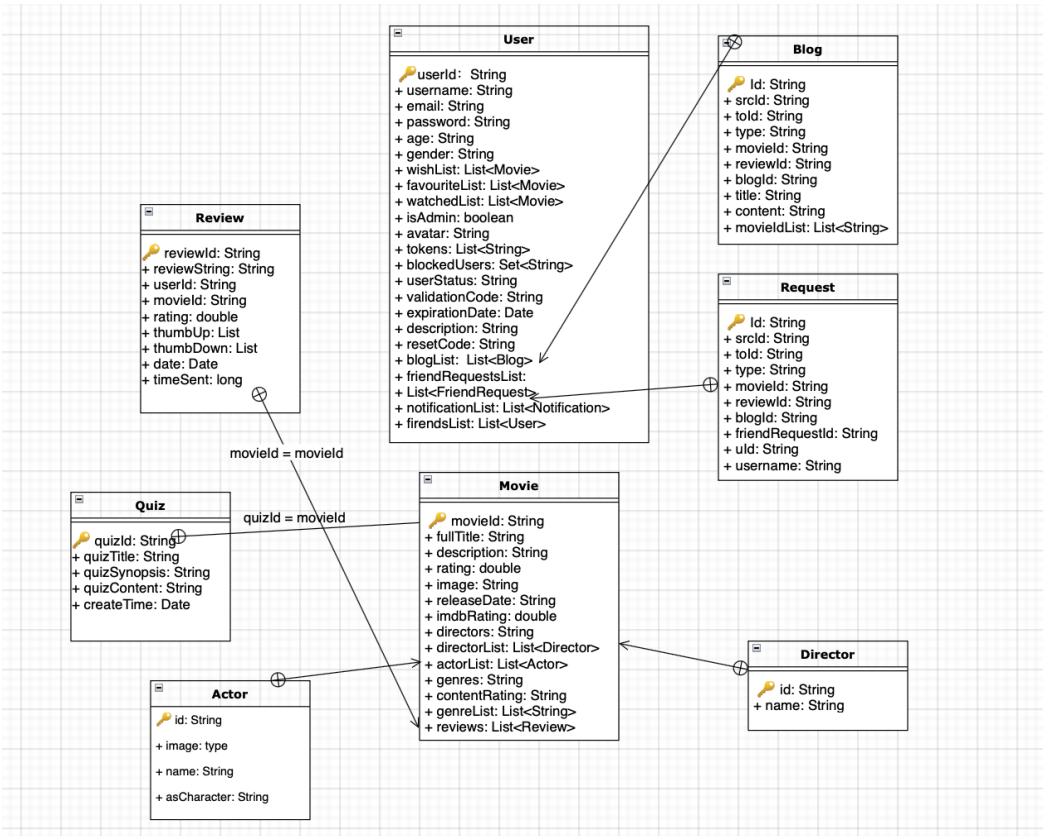


Figure 1.1.2.2: Detailed Collection Properties in MongoDB

1.1.3. Business Layer

Finally our Business Layer is developed in Java and leverages Spring Boot, an open-souce microservice framework for quick development of Java Web Applications.

As encapsulated in **Figure 1.1.2.2**, each collection items contain attributes that our backend will need to interact with. To achieve this, our backend design that adheres to the Object Oriented Programming principle is divided into three main classes, namely:

- View Class
    - This is the lowest level of our backend application
    - It provides the design seen in Figure 1.1.2.2, where MongoDB collection objects are mapped to Java classes for ease of access and consistency.
- Access class
    - This is the core of our Business Layer.
    - It provides the implementation details and logic for the interaction between our Java server and the MongoDB server.
    - This layer handles heavy tasks such as executing the recommendation algorithm and the filtering or sorting of data collections before returning response objects to the Controller class.
- Controller class
    - Handles request validation and parsing as well as valid or error responses
    - This layer allows the frontend and backend to seamlessly interact.

1.2. Page Structure

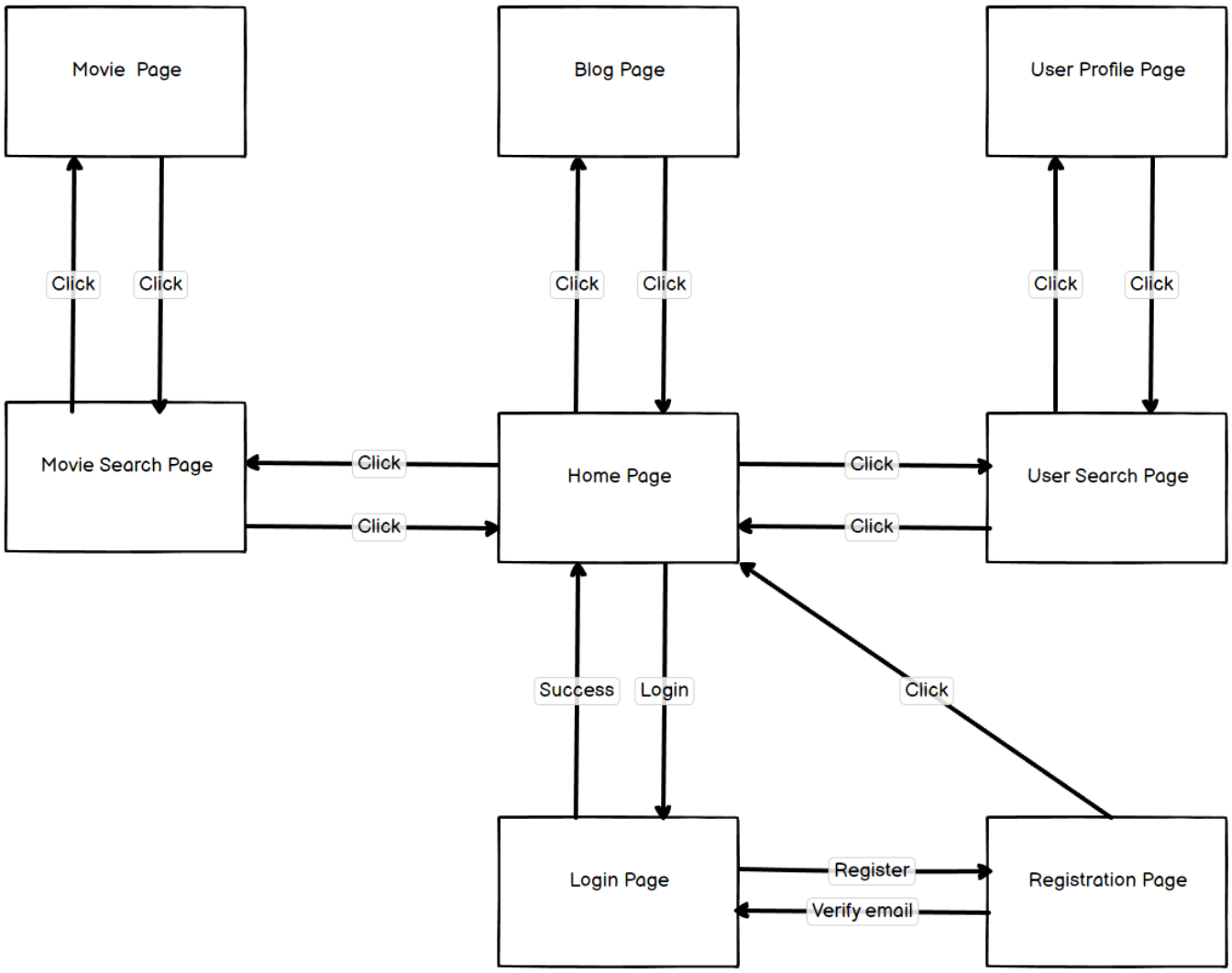


Figure 1.2.1: Page Interaction Flow Diagram

1.3. Project Objectives

Code	Description
UR	Registered **users** can be authenticated and have persistent data
AD	**Admin** users can have elevated privileges and can modify the database
MF	**Movie finders** can efficiently and conveniently browse or search for movies
CP	**Cinephile** can customise their profiles and categorise movies as they wish
RE	Users can receive different forms of **recommendations** from the system
CO	Users can interact with other users in the movie **community** in multiple forms
BL (novel)	Users can **blog** their thoughts about various topics and follow other bloggers
QZ (novel)	Admins can create **quizzes** about movies for users to participate in

1.3.1 User Stories

The table below displays the user stories proposed for the project. The code of each user story is prefixed by the predominant objective that they aim to satisfy.

Some user story codes are bolded to indicate that they were constructed from core functionalities that were proposed in the original Movie Finder System project description.

Code	User Story	Acceptance Criteria
UR1	As a user, I want to have a customised account so that I can store and share information about my movie interests.	● Users can create an account by entering a valid ○ email ○ password ● Users will be prompted to retry if the ○ email is invalid ○ email is already taken ○ password is less than 6 characters ● Once the account is created, the user should be automatically logged in.
UR2	As a user, I want to access my accounts from multiple devices so that I can always sync my personal information.	● Users can log in using a username/registered email and the corresponding password. ● Users can log in on multiple devices/sessions simultaneously. ● If a user fails to enter valid credentials after three attempts, they will be prompted to reset their password.

UR3	As a user, I want to sign out of my accounts when I am done so that no one can read, modify or steal my data.	● Users can log out from the current session. ● Choosing to log out from a single session does not affect other sessions ● Users should have the option to log out of all sessions.
UR4	As a user, I want the ability to reset my password so that I won't be locked out of my account if I forget my credentials.	● Users can reset their password by entering an email address ● If the email address is valid, the system should send an email to the given address with a reset code. ● Once the password has been reset, all active sessions should be made inactive (logged out). ● For security reasons, the user should not know whether the email address they entered is valid.
UR5	As a user, I want to remain logged in when using the same device so that I don't have to enter my credentials every time.	● Once a user is logged in, they should remain so unless another trigger (e.g. sign out or password-reset) occurs.
AD1	As an admin, I want to regulate movie reviews so that users are not negatively affected by inappropriate comments.	● Admin users can edit user reviews ● Admin users can remove user reviews
AD2	As an admin, I want to add new movies to our database so that regular users can have access to up-to-date movie content.	● Admin movies can add new movies to the database by providing an IMDb url.
AD3	As an admin, I want to update movies in our database so that any mistakes pointed out by regular users can be corrected.	● Admin movies can edit all details about a movie that will be displayed on the movie page. ● The website should display when the movie was last edited.
MF1 (novel)	As a movie finder, I want the ability to search for movies that match certain keywords so that I can quickly find movies about particular topics.	● Users can search movies by typing keywords into a search bar. ● Keywords can match a movie's ○ name ○ description ○ genre

MF2	As a movie finder, I want to know all relevant details about a movie so that I can decide whether it is worth watching.	● Users can view a movie's ○ movie id ○ full title ○ image ○ release date ○ rating ○ IMDB rating ○ description, ○ director list ○ actor list ○ genre list ○ reviews
MF3	As a movie finder, I want to filter movies based on different criteria so that I can more quickly find movies that match my interest.	On the browsing page: ● users should be able to filter movies based on: ○ director ○ genre ● users should be able to sort movies by ○ popularity (average rating) ■ movies with no rating will default to a value of 0. ○ names (lexicographically) ■ by default, only sort by name if the movies have equal popularity.
MF4 (novel)	As a movie finder, I want to filter out the movies I've marked so that I don't see the same movies again.	On the browsing page: ● Users can select any combination of their wish list, watched list and favourite list to omit from the display.
MF5	As a movie finder, I want to find a movie without signing up so that I can quickly and conveniently discover something to watch.	Without creating an account, anonymous users ● can search for movies ● can view a movie's details ● cannot add reviews or rate movies
MF6	As a movie finder, I want the ability to instantly view a random movie so that I can discover something new and expand my repertoire.	● Users can instantly view a random movie from the website. ● When a user chooses to reroll a random movie before refreshing the page, previous movies should not reappear.

MF7	As a movie finder, I want to receive immediate search predictions when looking for a particular movie so that I can quickly locate the movie I had in mind.	● When searching for keywords, relevant movie titles should be displayed and updated below the search bar after each change to the search string.
CP1	As a cinephilia, I want to save movies to a list so that I can watch, track and review them at a later date.	● Users can add movies to their wishlist, watched list and favourite list. ● Users can remove movies from their wishlist, watched list and favourite list.
CP2	As a cinephilia, I want to upload an avatar so other users can quickly identify me when using the application.	● Users should be able to upload a profile picture to their account. ● The image should be in JPG or PNG format. ● The size of the image cannot exceed 2 MB.
CP3	As a cinephilia, I want control over who can view my profile so that I can protect my privacy whilst still sharing my movie interest with my targeted audience.	● Users can select whether their profile should be visible to the: ○ public ○ friends ○ private (just themselves) ● Users can search for a list of other users ● Users can view other user's profiles
CP4	As a cinephilia, I want to update my profile so that it reflects my most recent personal information.	● Users should be able to edit their ○ username ○ email ○ password ○ profile picture ● Users must enter the current password if they wish to change their password.
RE1	As a movie watcher, I want to be shown similar movies when viewing a movie's page so that I can select another movie that matches my interest.	● When viewing a movie page, users will receive recommendations on similar movies based on a selection from the movie's ○ review history ○ genre ○ director ○ cast(s)
RE2	As a movie watcher, I want to receive recommendations based on movies I've watched so that I can find movies that are similar to my interest.	● When a user views their profile page, they should receive recommendations of similar movies based on their "watched list"

CO1	As a movie critic, I want to share my thoughts and experience on a movie I've seen so that I can help other users decide if a movie is worth watching.	● Users can leave a review on the movie page ● Reviews should include ○ text ○ rating (from 0 to 5) ● The review title cannot exceed 50 characters ● The review text cannot exceed 500 characters.
CO2	As a movie follower, I want to see other people's movie interests so that I can also watch those movies.	● Users can view other users' wishlists, watched lists and favourite lists when visiting their profile page. ● Users can navigate a targeted user's profile page by clicking on their avatar, which is always displayed when the targeted user makes a comment on the movie page. ● Users cannot view a profile that is set to private
CO3	As a movie follower, I want to block users that I dislike so that I don't have to see their activities.	● Users can block other users by adding them to a "banned" list. ● Users can remove blocked users. ● When a user is blocked, their movie reviews will not be displayed to the current user. ● When a user is blocked, their rating of the movie will not influence the average rating displayed to the current user.
BL1 (novel)	As a blogger, I want to blog about all kinds of movie-related topics so that I can attract a wider audience and recommend the movies I like.	● Users can create blog posts by entering a title and a description. ○ title must be at most 50 characters ○ blog post must be at most 1000 characters ● Blog posts can link to multiple movies.
BL2 (novel)	As a blogger, I want to display all of my blogs in one place so that I can manage and track them more easily.	● Users can view their blogs in their profile ● Users can view other users' blogs if their profile is set to public or set to friends and they are a friend.
BL3 (novel)	As a blogger, I want to transparently edit my blogs so that I can fix mistakes without looking deceptive to the public.	● Users can edit their blogs about movies ● Edited blogs will be time-stamped at the time of edit (i.e. marked by "last edited at TIME").

QZ1 (novel)	As an admin, I want to create quizzes about movies so that users can be engaged and stay longer on the website.	• Admins can create quizzes on particular movies • Users cannot create quizzes • Users can play quizzes about particular movies
QZ2 (novel)	As an admin, I want to modify movie quizzes so that I can make corrections or add new questions when more interesting facts and information are discovered.	• Admins can edit movie quizzes • Admins can remove movie quizzes • Users cannot edit quizzes • Users cannot remove quizzes
QZ3 (novel)	As a user, I want to be informed of my performance in a movie quiz so that I can determine how well I know my favourite movies.	• Users receive a score after partaking in a movie quiz. • Users can compare their scores with other players through a ranking system.

2. Functionality Details

2.1 Login Page

Addresses **UR1** and **UR4**

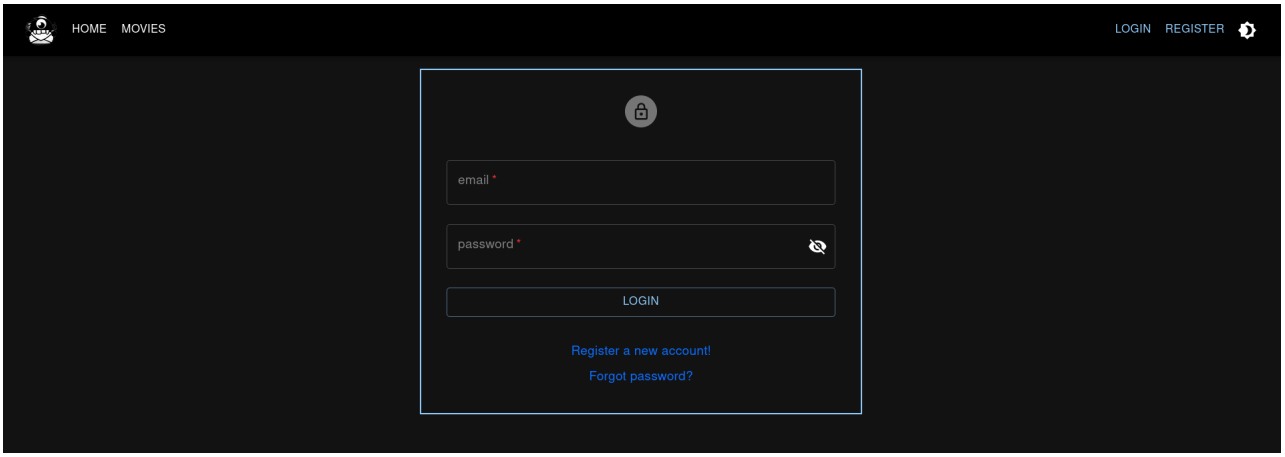


Figure 2.1.1: Login Page

This login page is accessed through the “LOGIN” button at the top right corner of the navigation bar. It will only appear for guest (anonymous) users. Below are the conditions and respective results on the login page.

Condition	**Result**
Email or Password is incorrect or invalid	A relevant error message will appear
Account has not been email-validated	A relevant error message will appear

Click “Register a new account!”	Navigates to the registration page
Click “Forgot a password?”	Navigates to a new page so that the user could reset their password.
Login successfully	A welcome message pops up and navigates the user to the previous page if one exists, or otherwise the home page

2.2 Register Page

Addresses **UR1**

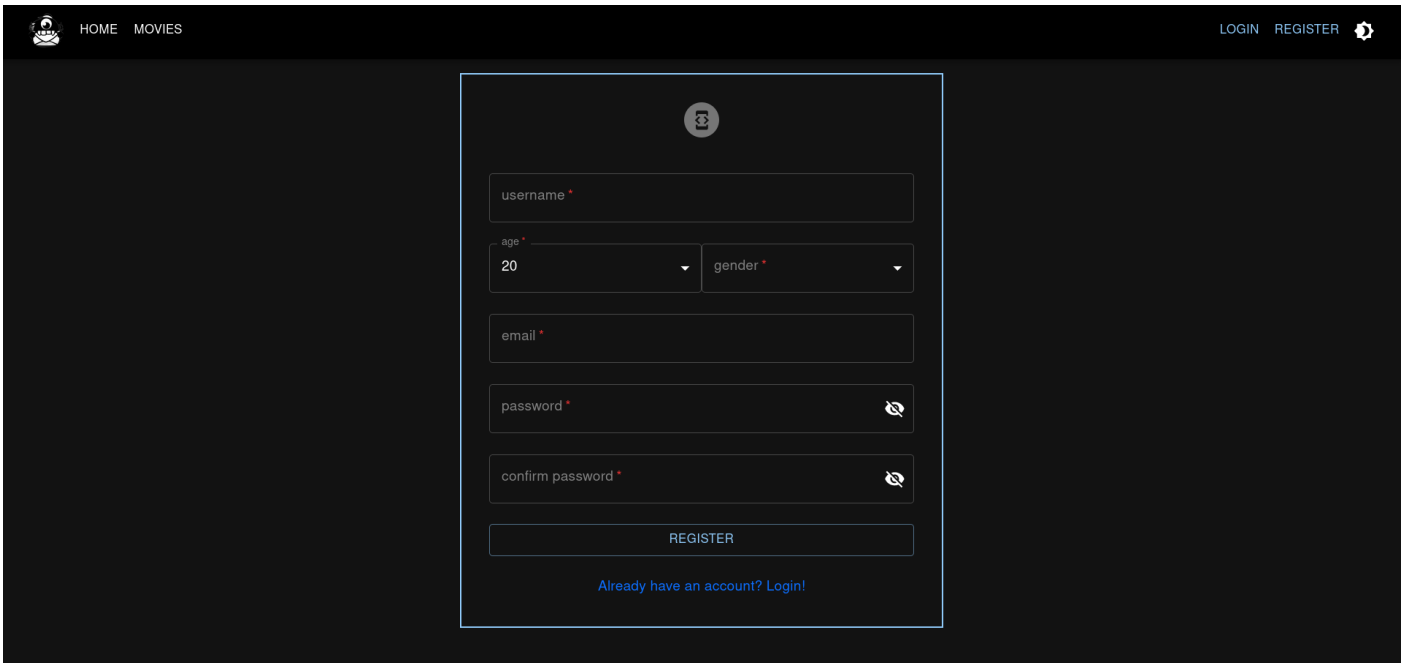


Figure 2.2.1: Register Page

Condition	**Result**
Email is invalid or has been taken	Display a relevant error message.
Password length is less than 6 characters	Display a relevant error message.
Click the “Register” button	A verification email will be sent to the user containing a link that will activate their account

2.3 Movies Page

Addresses **MF1, MF3, MF4, MF5** and **MF7**

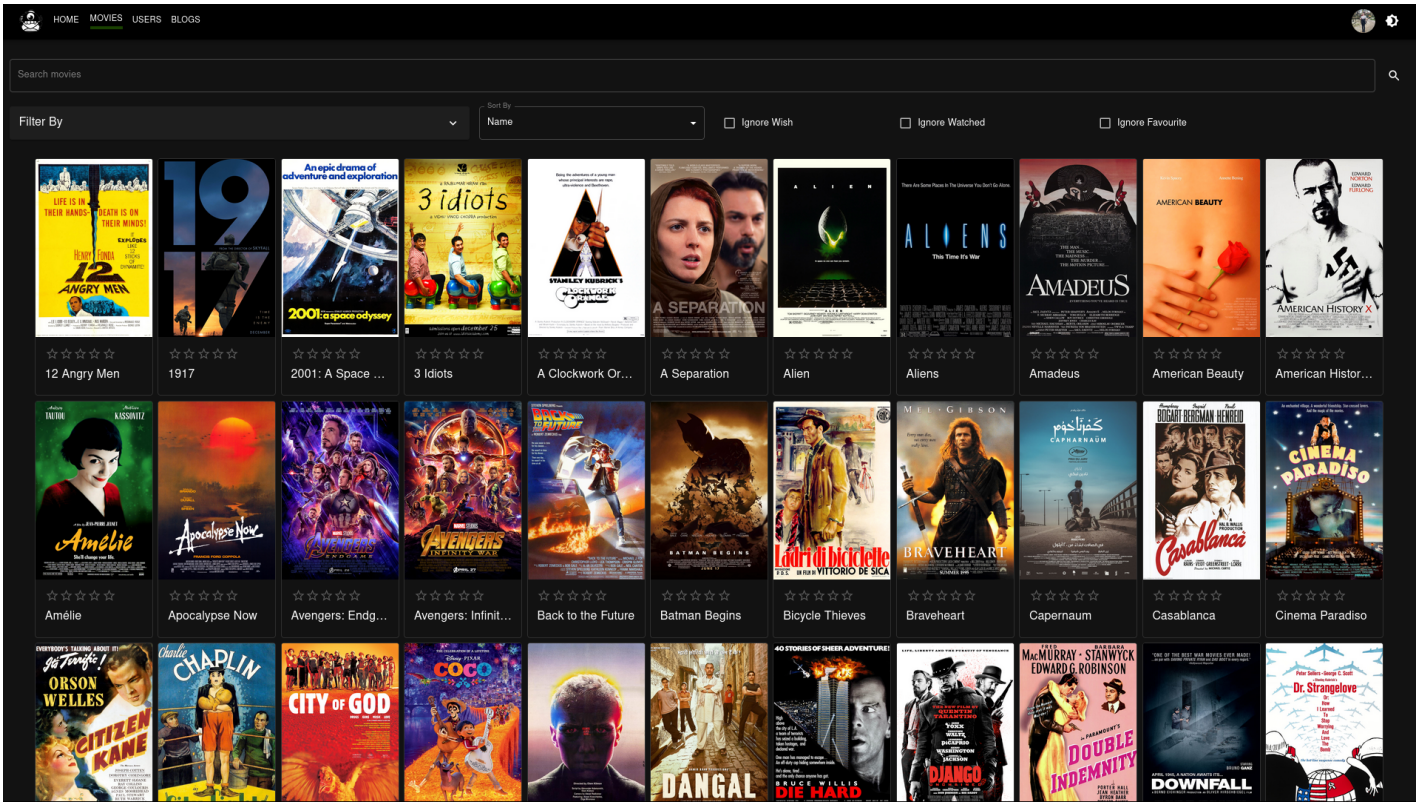


Figure 2.3.1: Movies Page

This page can be accessed by both registered users or anonymous (guest) users. On this page

- users can search for movies by keyword
- when users are typing in the search bar, auto predictions will appear for movie titles
- users can filter movies by genre and/or directors
- users can sort movies by either name or rating first

For registered users,

- Movies that these users have added to their wish/watched/favourite can be ignored

The possible conditions and respective results are as follow:

Condition	Result
Click “ Filter By”	New selection will pop up below for director/genre
Tick “Sorted by”	Movies will be sorted according to the user selection
Tick “Ignore Wish”	Movies that are in users wish list will be ignored
Tick “Ignore Watch”	Movies that are in users watch list will be ignored
Tick “Ignore Favourite”	Movies that are in users favourite list will be ignored

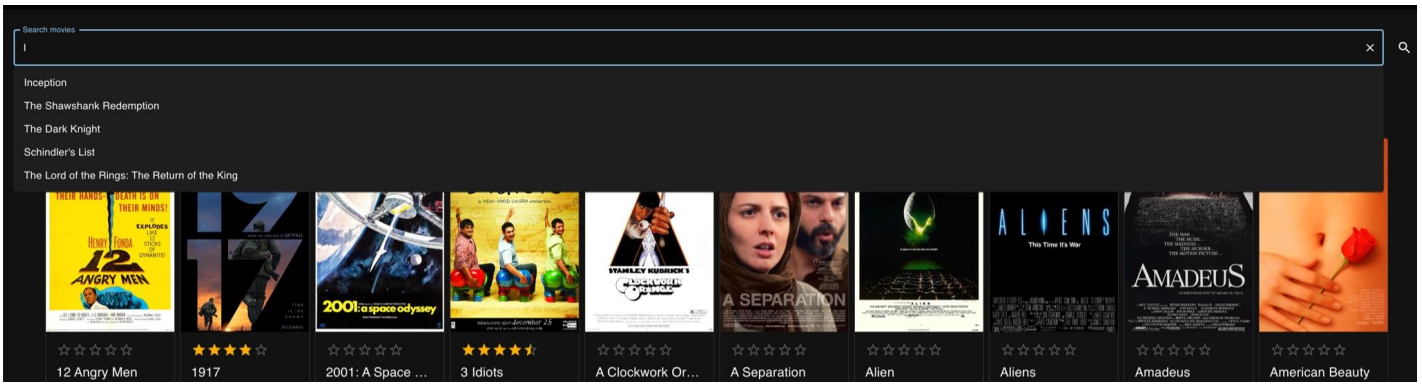


Figure 2.3.2: Movie search and autocomplete predictions

When clicking search movie bar, typing keywords, the relevant movie titles should be displayed and updated below the search bar after each change to the search string.

2.4 Movie Details Page

Addresses **MF2, AD3, CO1, CP1, RE1** and **AD1**

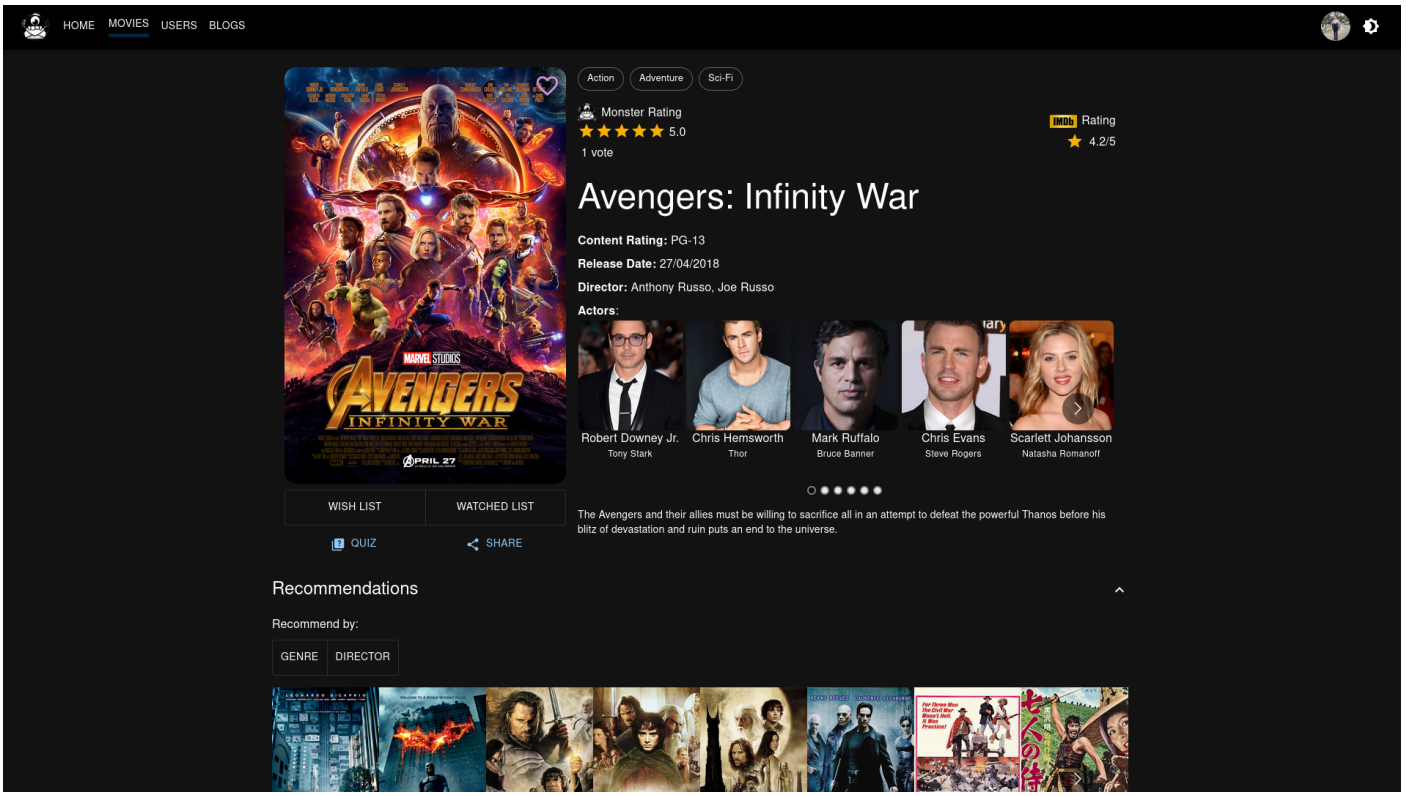


Figure 2.4.1: Movie Detail Page

The movie details page contains information such as the actor list, our website site rating, IMDb rating and the movie's description. Users can add this movie to their wishlist, watchlist or favourite list. Users can also leave a review containing their ratings and comment.

Recommendations for the current movie are displayed at the bottom, whereby users can select to be recommended movies based on genres and/or directors. Note that watched-list recommendations are displayed for logged-in users on the home page.

The possible conditions for the movie details page and their respective results are as follows:

Condition	Result
Click the “ WISHLIST” button	This movie will either be added to or removed from the user’s wishlist.
Click the “WATCHED LIST” button	This movie will either be added to or removed from the user’s watched list.
Click the heart icon at the top-right corner of the movie poster	This movie will either be added to or removed from the user’s favourite list.
Choose “GENRE” as recommendation criteria	recommendations for movies with similar genres will be displayed.
Choose “DIRECTOR” as the recommendation criteria	recommendations for movies with similar directors will be displayed.
Add reviews and click submit button	A review will be added to this movie and the user can see the latest view under the reviews section. The movie’s number of reviews and ratings will be updated accordingly.

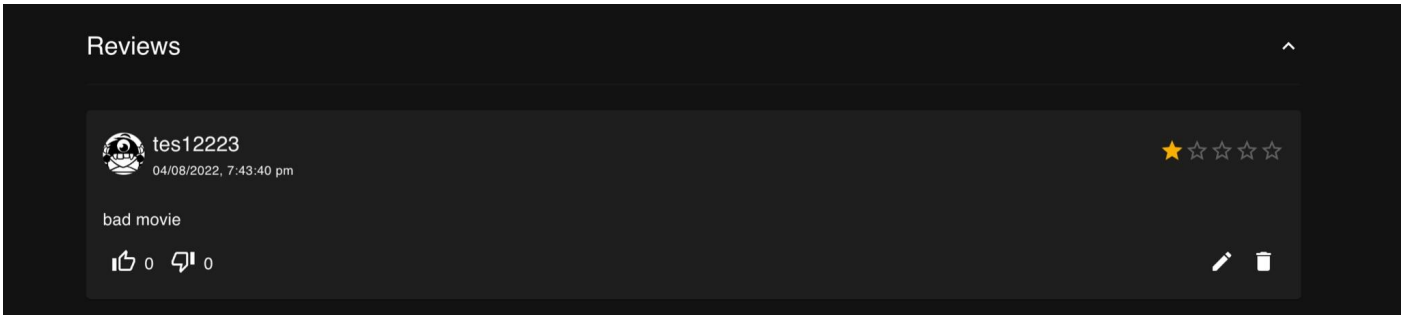


Figure 2.4.2: Movie Review Example

Condition	Result
Click the “Bin” icon.	This review will be deleted.
Click the “Pencil” icon	This review will be edited if you are the admin or author of this review.
Click the “like” icon	User will like or unlike the review.
Click the “dislike” icon	User will dislike or undislike the review.

2.5 User Profile

Addresses **CP1, CP2, CP3, UR5, CP4, CO2, CO3**

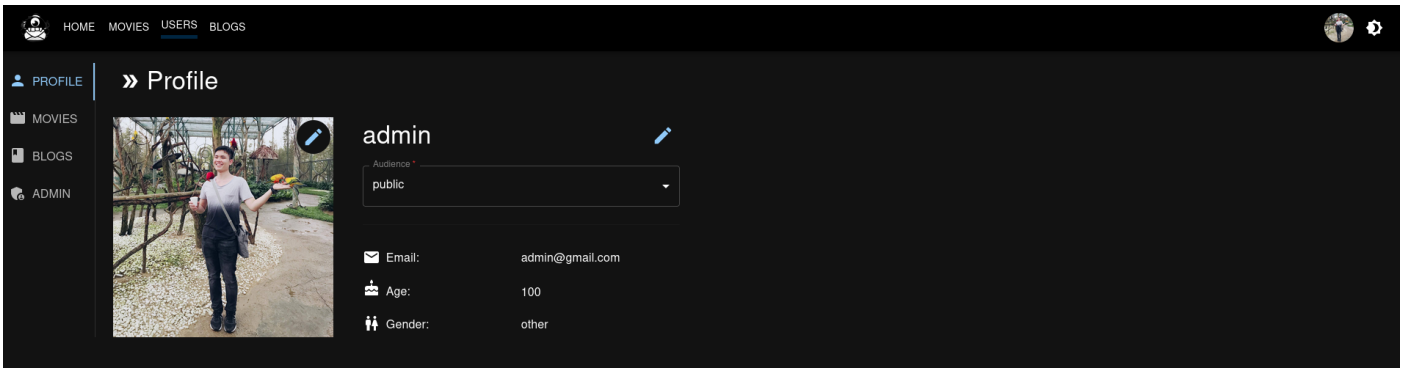


Figure 2.5.1: User Profile Tab Default View

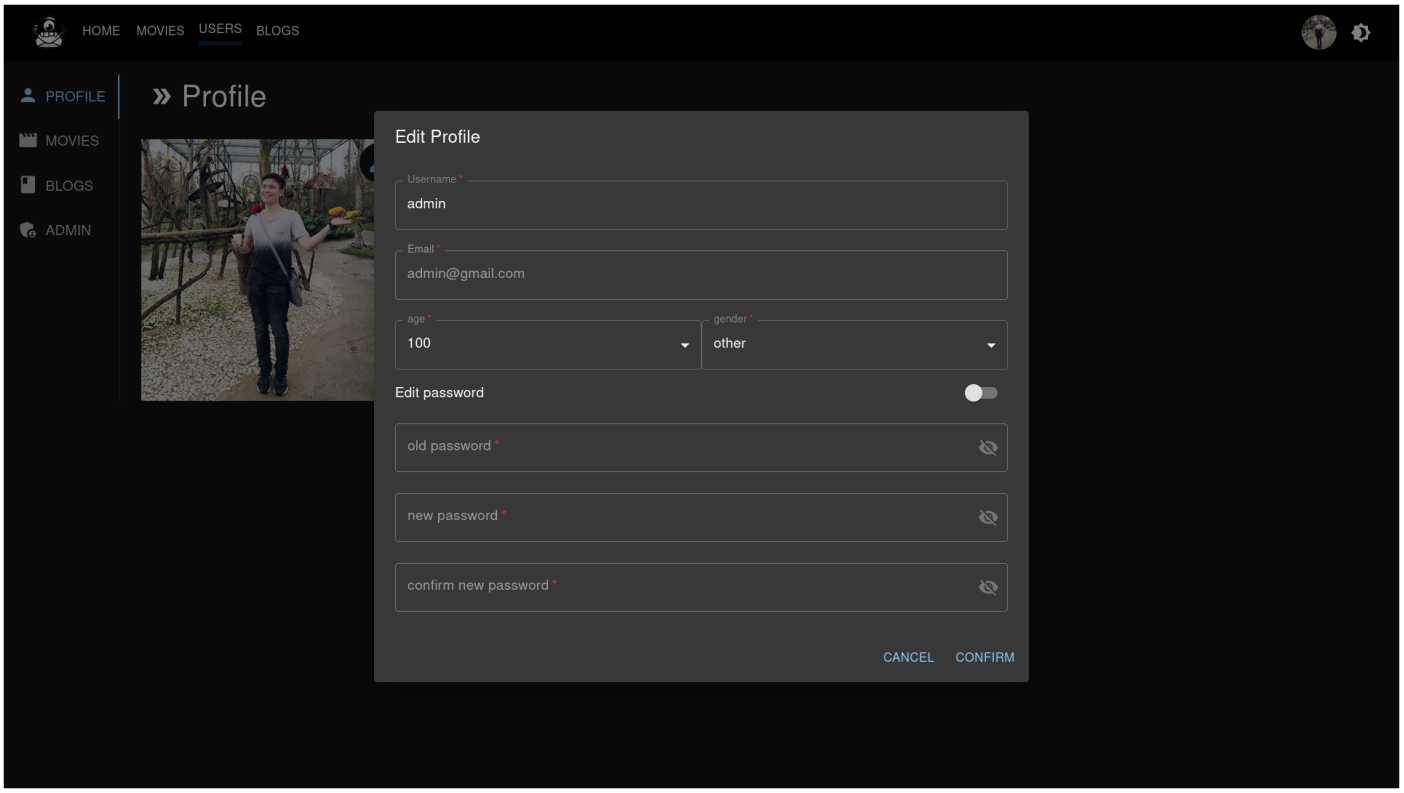


Figure 2.5.2: User Profile Tab Edit Profile

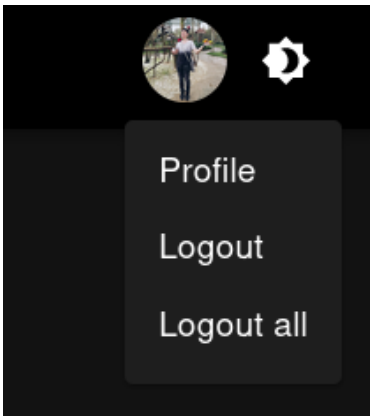


Figure 2.5.3: Log out

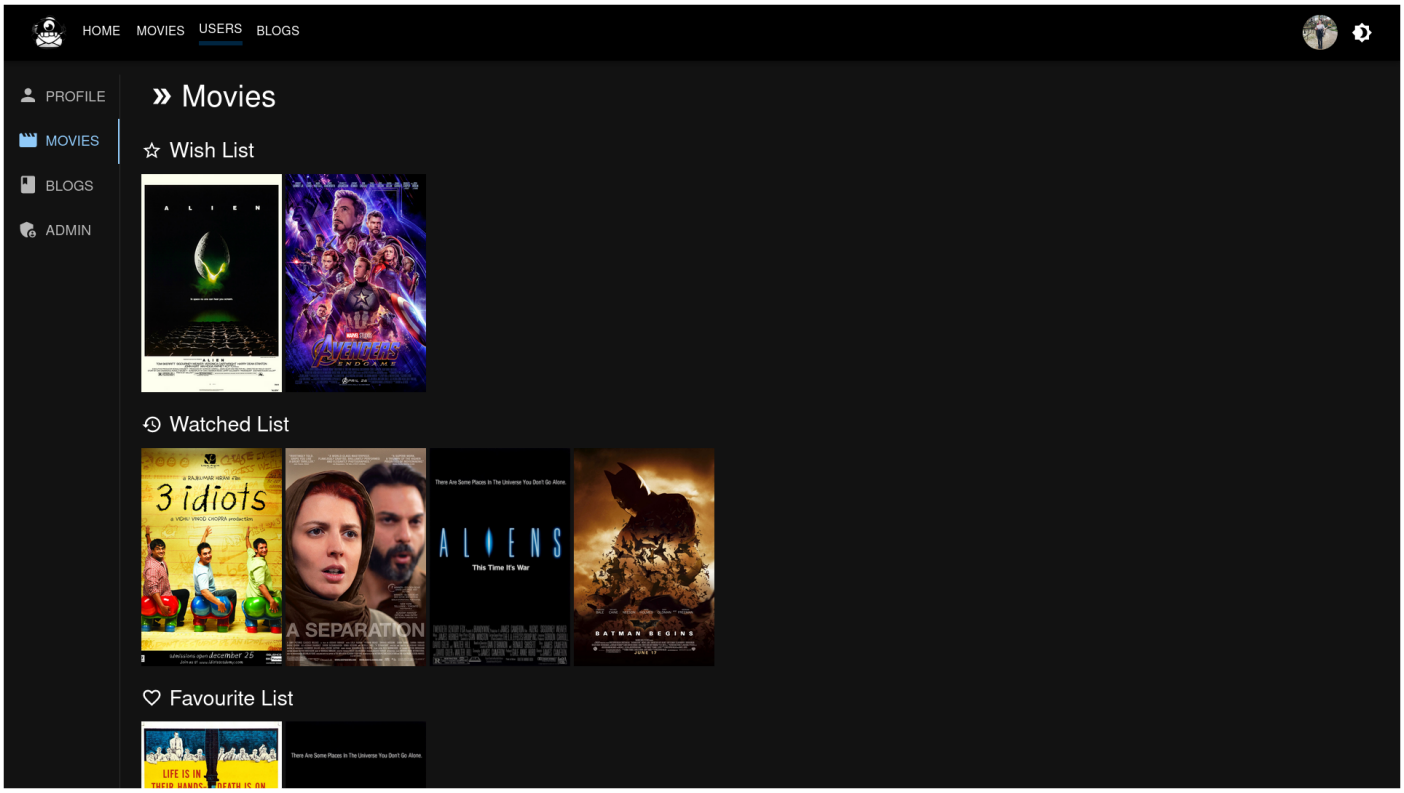


Figure 2.5.4: User Movies Tab

This page is used to display a user profile and three lists for movies.

While globally accessible on all pages, we will also address the log-out functionality here. The possible conditions and respective results are as follows:

Condition	**Result**
Click “profile” button	Navigates user to the profile tab
Click “movies” button	Navigates user to the tab showing their lists
Click the edit button	Profile modal will pop up for editing
In Edit Modal, click “Confirm”	User profile detail will be updated accordingly
Click Audience and select private or public	Updates the viewable audience of the profile
Click “User” icon and choose to log out	Current user will be logged out and the page will be redirected to login page
Click “User” icon and choose to log out all	User all devices will be logged out and the page will be redirected to login page
View other user profile and Click on “ban” button	A message will pop up and the ban button will change to unban button

2.6 Home Page

Addresses **RE2**

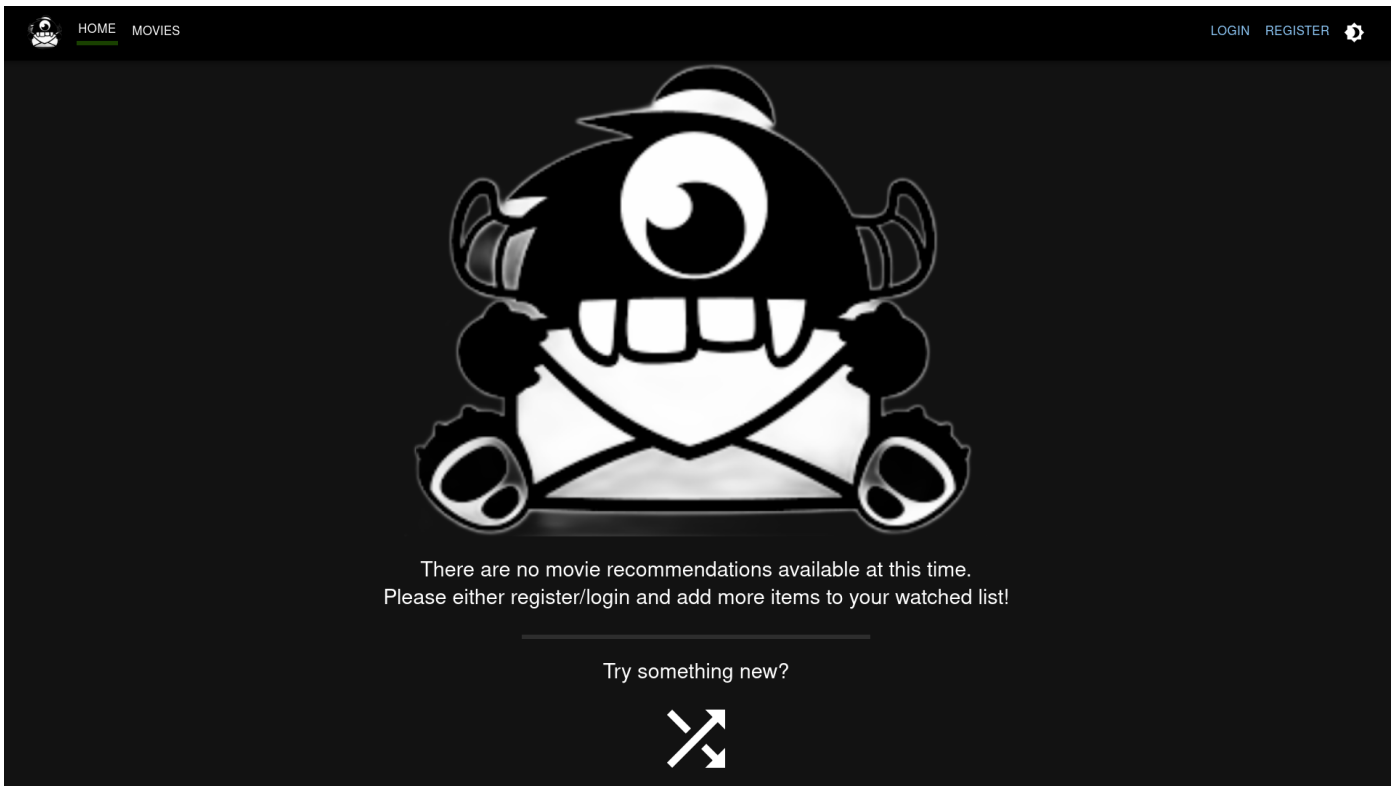


Figure 2.6.1: Home Page for Guest Users

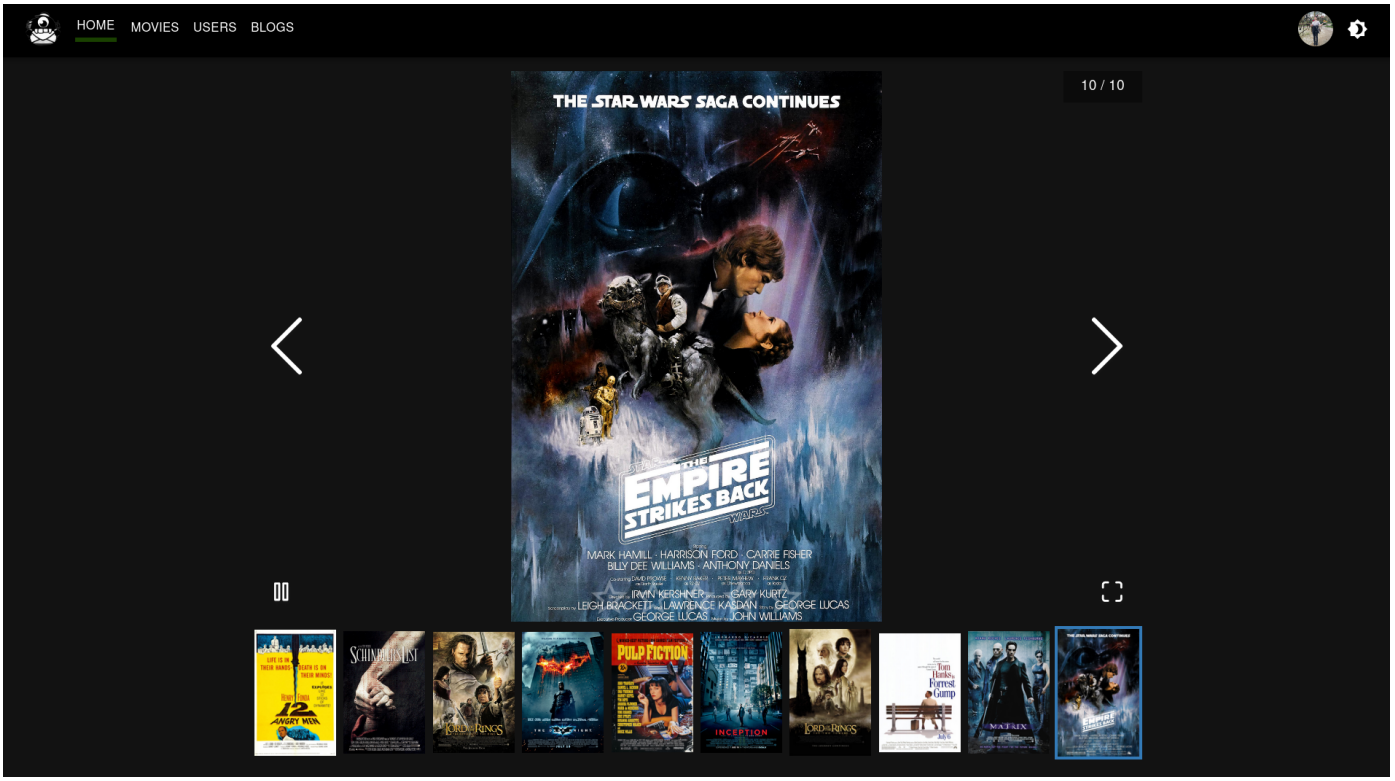


Figure 2.6.2. Home Page for Registered Users containing Recommendations based on Watched List

Condition	Result
Click the “Shuffle” Icon at the bottom of the screen under “Try Something New”.	The user will be navigated to a random movie page
In the Registered User Home Page, a user clicks on the enlarged movie poster	User will be navigated to the enlarged recommended movie page

2.7 Admin Page

Addresses **AD2**

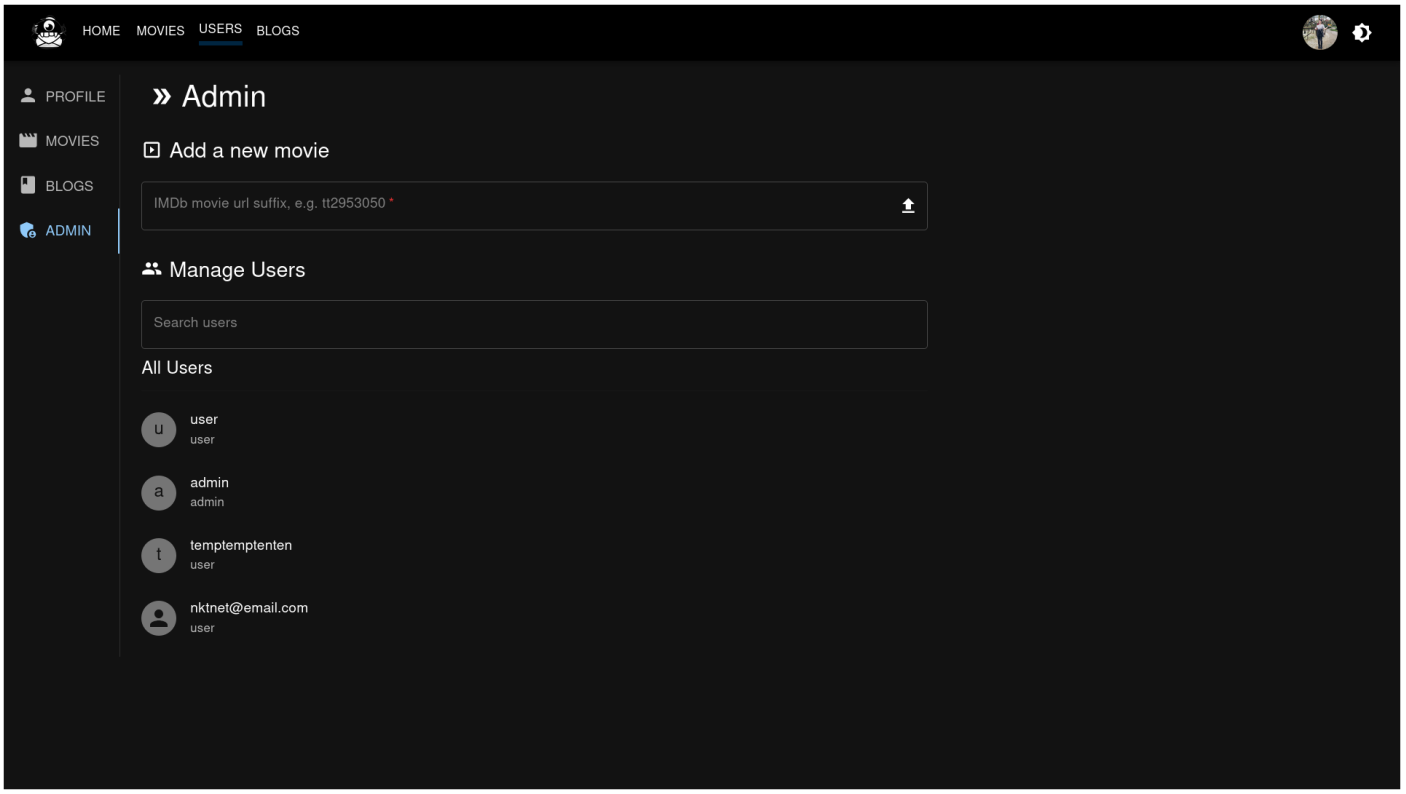


Figure 2.7.1: Admin Page

This page is used for administrators. It contains Manage User and Add Movie. Administrators can add new movies to the database by providing an IMDb Id, which can be retrieved from any IMDb movie page.

2.8 Blog pages

Addresses **BL1, BL2, BL3** and **BL4**

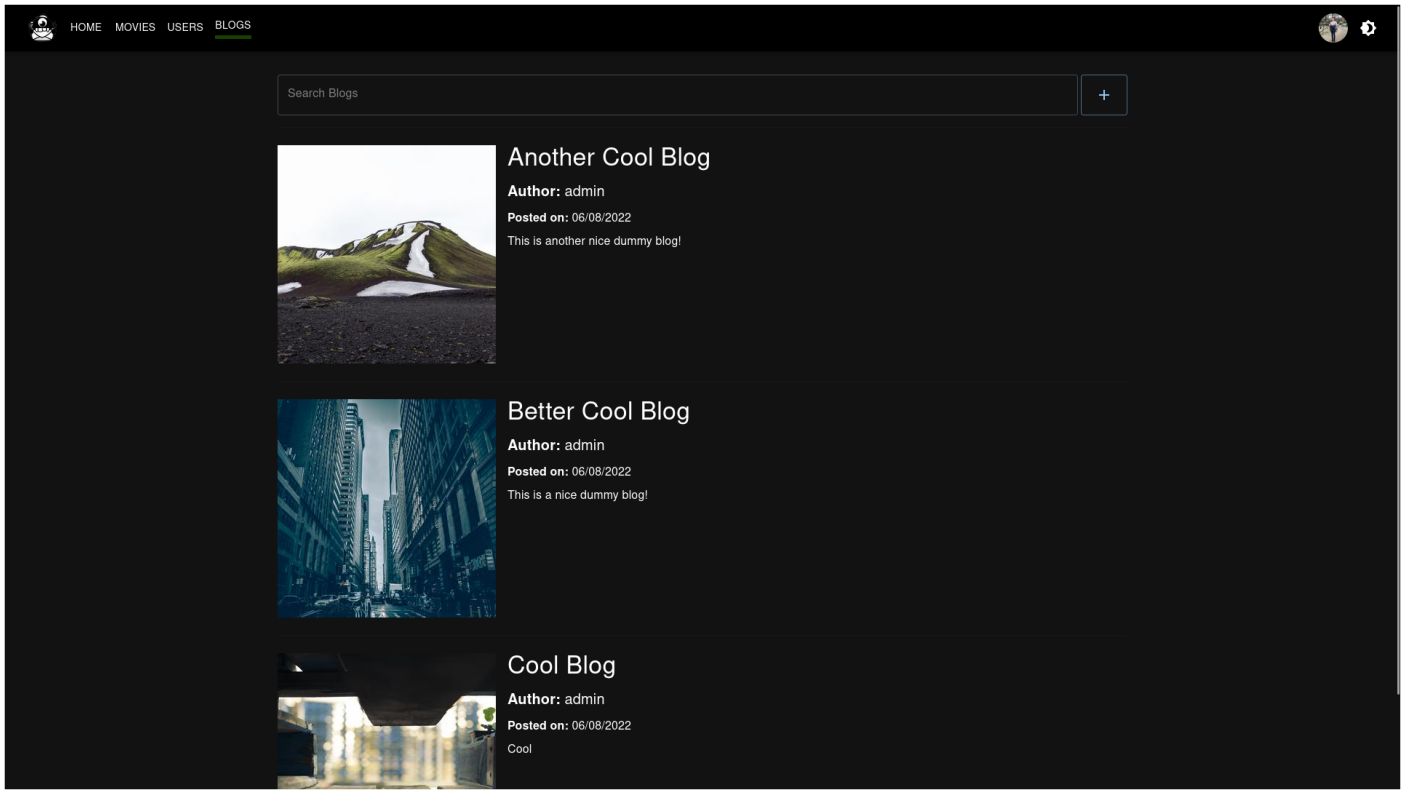


Figure 2.8.1: Blogs List Page

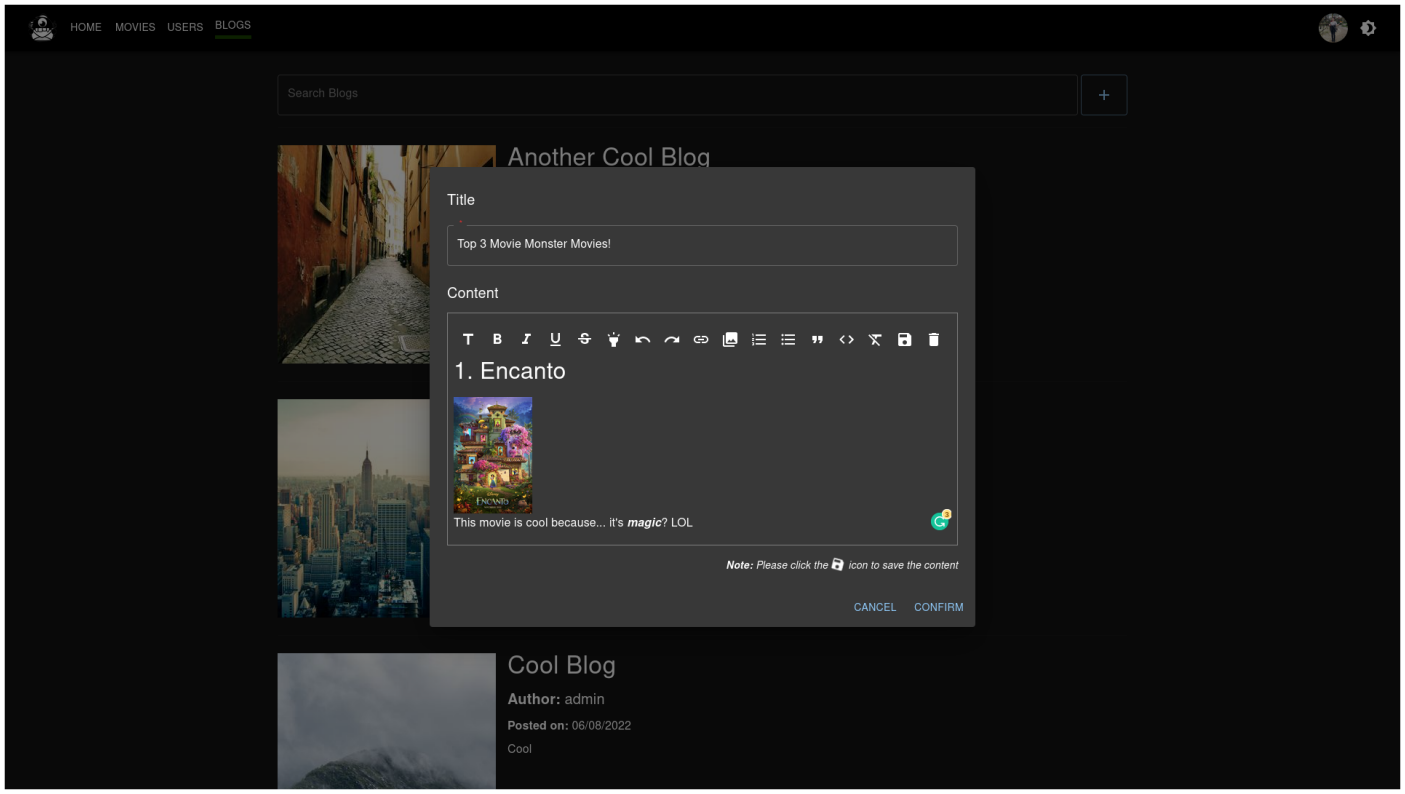


Figure 2.8.2: Blog Create Modal

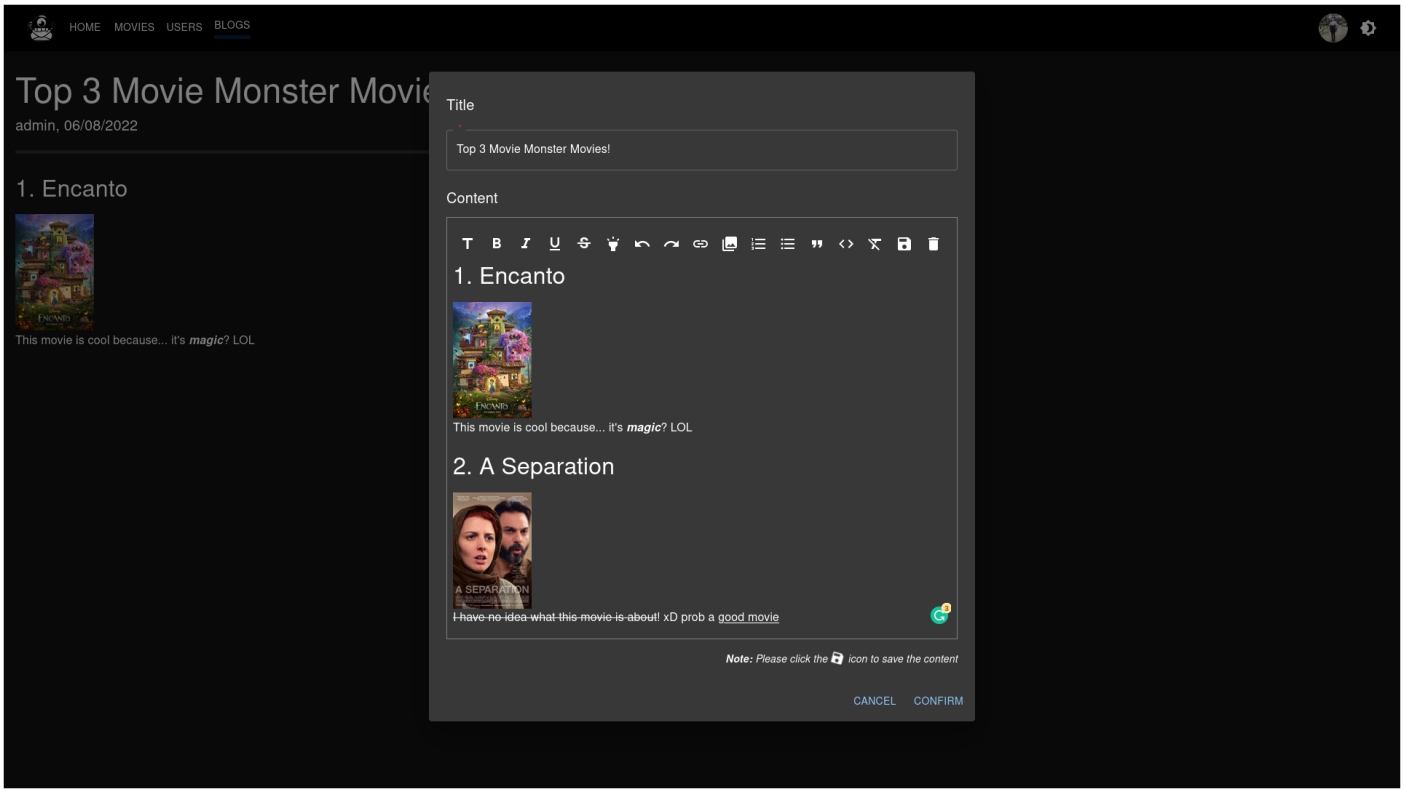


Figure 2.8.3: Blog Edit Modal

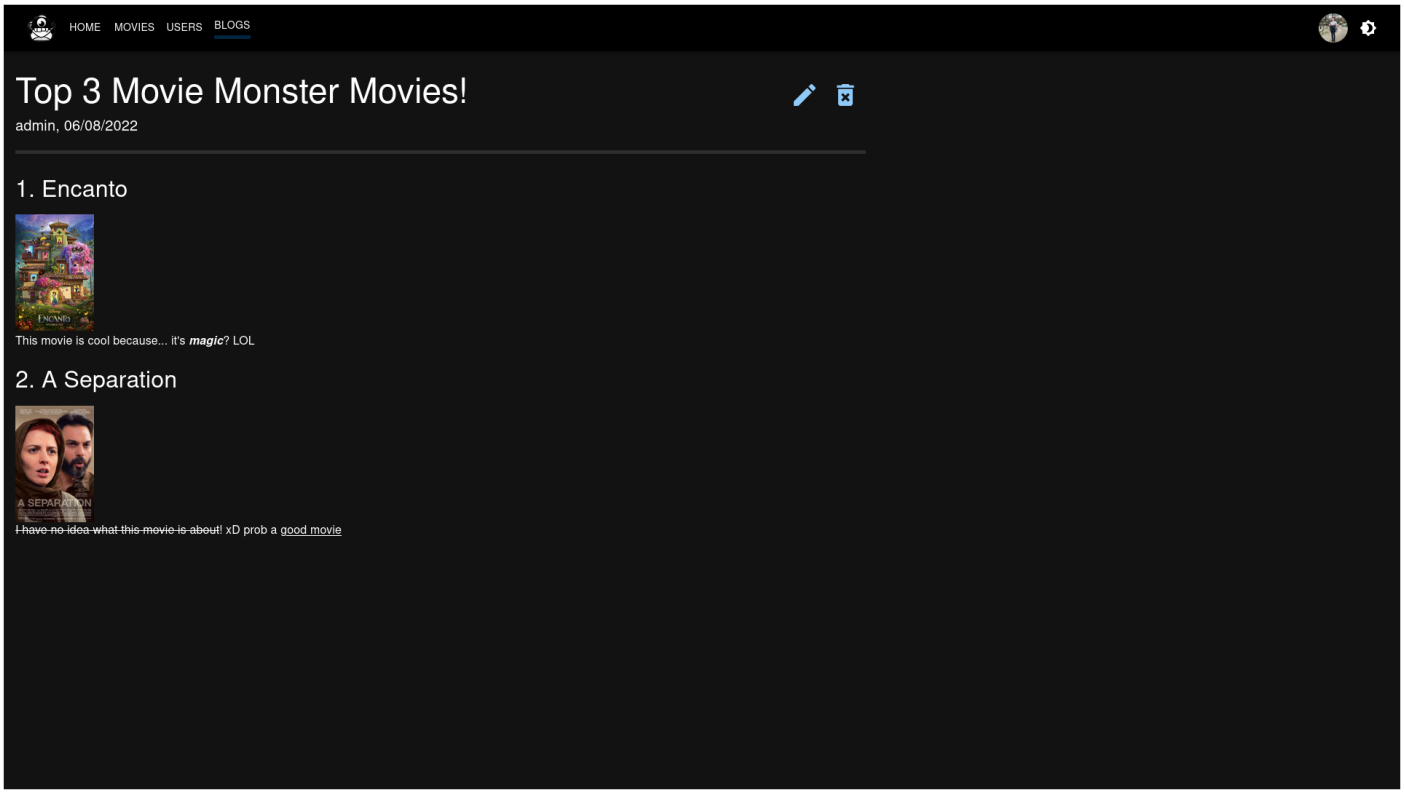


Figure 2.8.4: Blog Details Page

2.9 Quiz Page (Novel)

Addresses **QZ1**, **QZ2** and **QZ3**

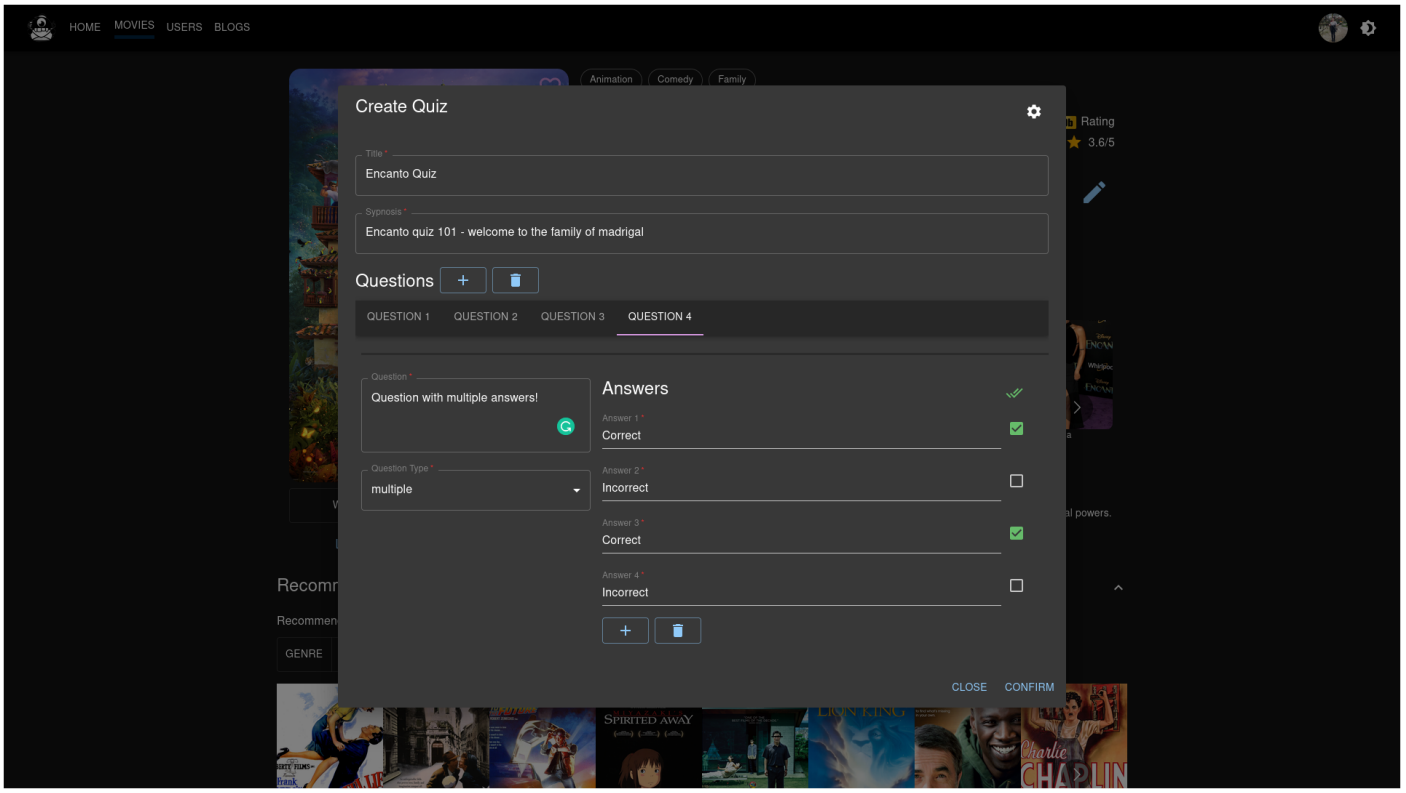


Figure 2.9.1: Quiz Create Modal

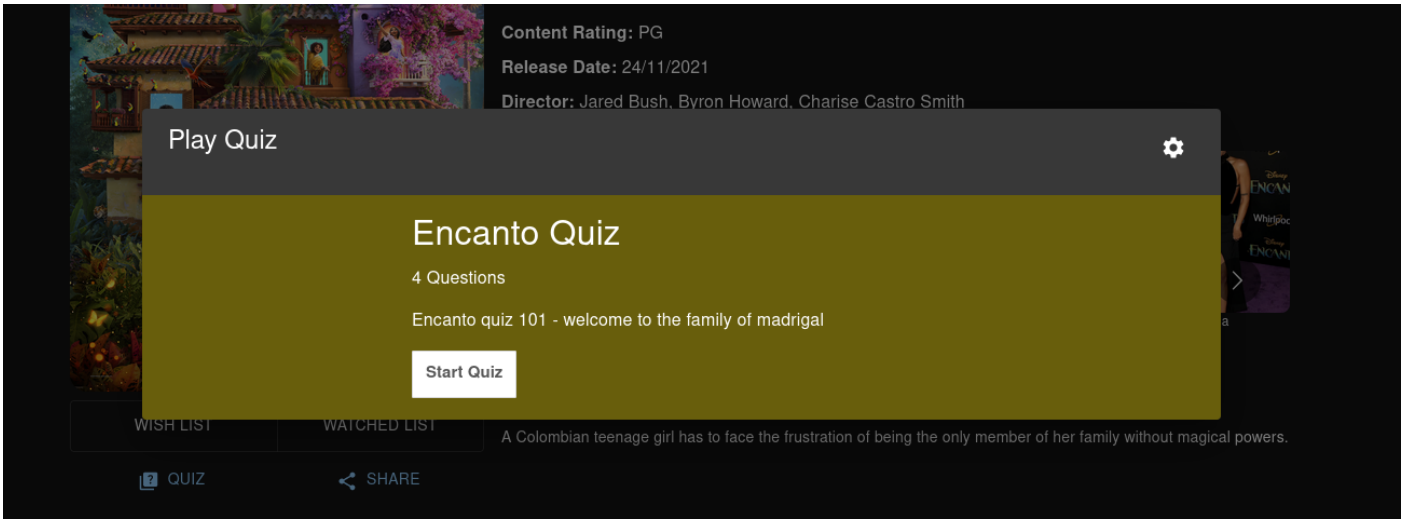


Figure 2.9.2: Quiz Play Initial State

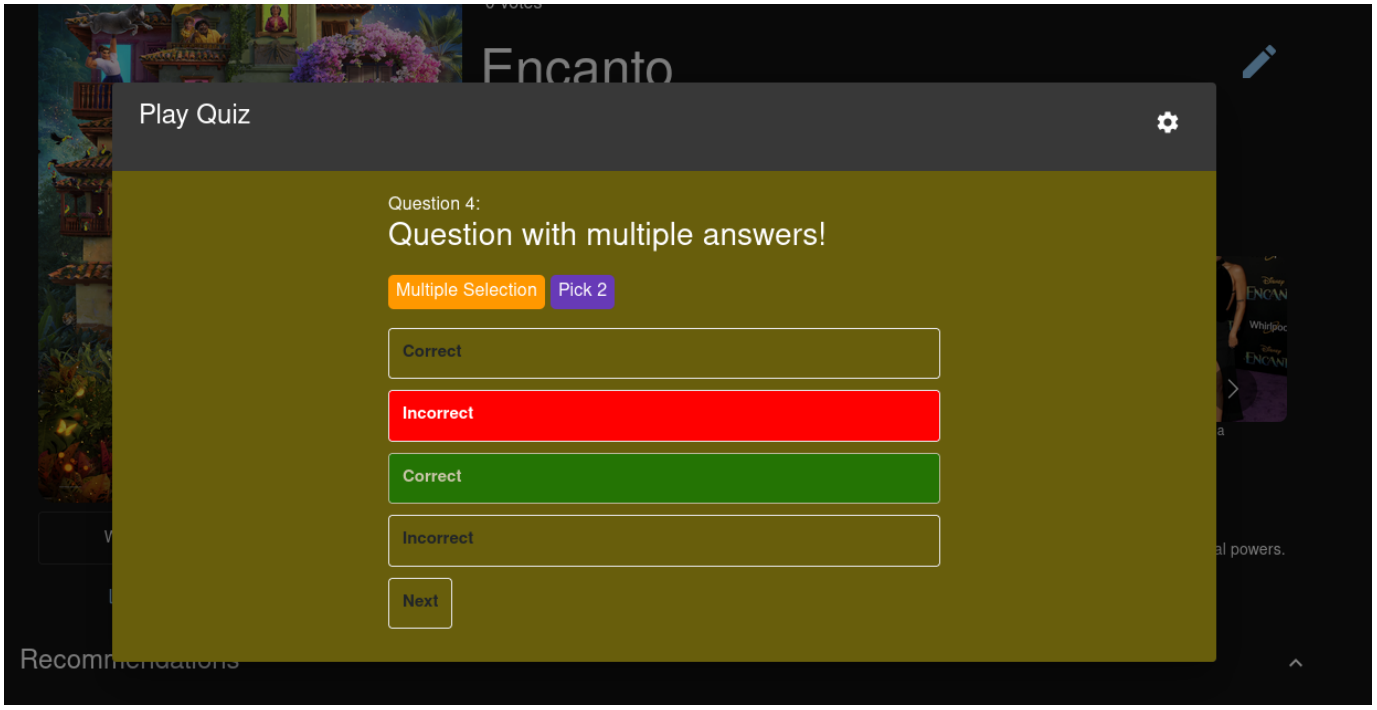


Figure 2.9.3: Quiz Play Question State

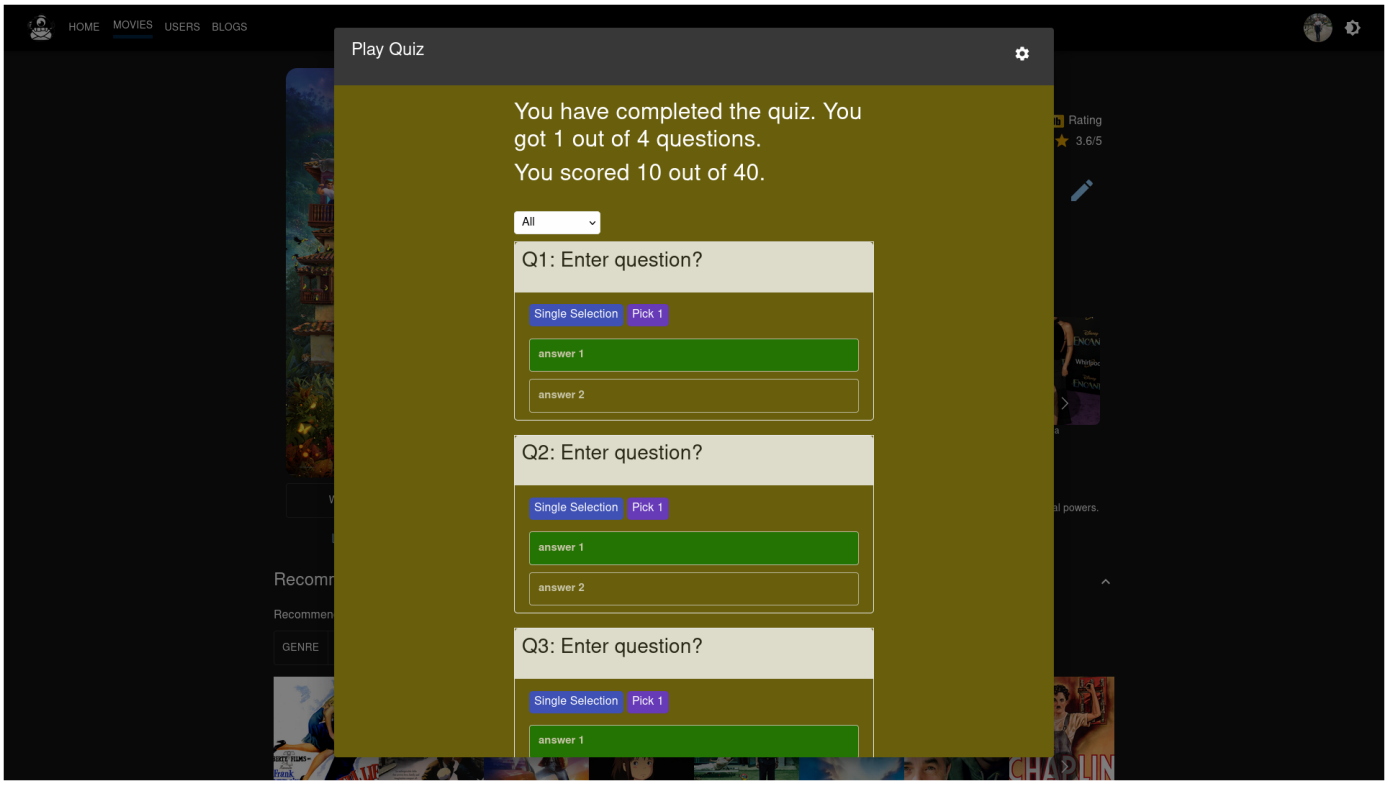


Figure 2.9.2: Quiz Play Result State

3. Third-party functionalities

3.1 Frontend

All components on the frontend are either self-written or free and open-source, so there were no licensing issues that we faced during our website development.

3.1.1. Theme

As mentioned previously, our frontend is themed with Material UI and Bootstrap. Material UI was mainly used for styled components that were both responsive and adaptable to the current theme, as well as providing a consistent modern feel to our site. Bootstrap, however, was mainly utilised for style-overriding and adjustments for components from other third parties or self-written.

3.1.3. State Management

Redux was chosen to manage and centralise our front-end application state - this was done to provide a clear separation between the page layout states from the global states (such as the current user token, which should be accessible on all pages). The interaction between front-end and backend was also abstracted away in Redux so that one frontend developer can focus on the user interface while the other addresses the requests and responses that the user may send to the backend.

3.1.4. Components

Below are a list of major components that were used to improve the visual aspect of our site. All packages listed will contain their name as appeared on the npm registry:

- **react-image-gallery**: For home page watch list recommendations
- **react-infinite-scroll-component:** Used in movie list page to enhance the user experience by automatically fetching new movies when they've scrolled to the bottom of the page
- **react-multi-carousel**: For displaying wish/watched/favourite lists and movie recommendations based on genre and/or director
- **react-quiz-component-3900:** forked and published from react-quiz-component to fix existing bugs and enables compatibility with TypeScript
- **mui-rte:** rich text editor that is compatible with Material UI themes, used in our blogs
- **react-swipeable-views**: for quiz questions and allow users to swipe left or right on mobile devices, thus improving our site's responsiveness and support for multiple devices

3.2 Backend

3.2.1 Spring Boot

Spring Boot is an open-source Java-based framework for creating a microservice. It was developed by the Pivotal team to build standalone and production-ready Spring applications (Spring Boot - Introduction, 2022). The framework was chosen as the back-end members were familiar with using Java. Spring Boot is easy to understand and develop Spring applications and can increase productivity and save development time

3.2.2 Python libraries

3.2.2.1 Pandas

Pandas is a tool for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format. In this project, pandas transpose data from data sources to python for easy data processing (“Pandas”, 2022).

3.2.2.2 Scikit-learn

Sci-kit-learn is a Python module for machine learning built on top of SciPy and is distributed under the 3-Clause BSD licence. In this project, sci-kit-learn is used to implement recommendation systems (Sci-kit-learn 1.1 Documentation, 2022).

3.2.2.3 Pymongo

PyMongo is a Python distribution containing tools for working with MongoDB and is the recommended way to work with MongoDB from Python (Python Documentation, 2022). In this project, pymongo connects MongoDB and python.

3.2.2.4 Alive-progress

Alive-progress is a new kind of Progress Bar, with real-time throughput, ETA, and very cool animations (alive-progress, 2022).

3.2.3 Maven

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project (Maven - Wikipedia, 2022). In this project, we use some dependencies from maven.

3.2.3.1 Spring framework

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments (Spring Framework, 2022).

3.2.3.2 Projectlombok

Lombok can simplify Java code in the form of simple annotations, making it more efficient for developers.

3.3.3.3 Jsonwebtoken

JSON Web Token (JWT) is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key (JSON Web Token - Wikipedia, 2022).

3.2.4 MongoDB

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public Licence (SSPL) which is deemed non-free by several distributions ("MongoDB - Wikipedia", 2022).

3.3. IMDb API

IMDb's Application Programming Interface was leveraged extensively in our system for the purpose of adding new movies to our database, where a subset of information is extracted and stored in MongoDB. This approach was chosen to remove the reliance on the site admins to manually add movies which would have been neither convenient nor robust. By relying on an external trusted source of movie data, our developers were able to focus more on the design and other distinguishing aspects of our Movie Monster system.

4. Implementation Challenges

4.1 Frontend

The main challenges faced in the frontend were the sheer number of pages and screens that needed to be designed and tweaked to fit the needs of our system. With only two developers working, we needed to divide and conquer in a way that is most efficient. Initially, the work was divided based on features, which were suboptimal as it resulted in inconsistent designs and overlapping or redundant code being produced, which were reflected in our Retrospective A. This was improved in Sprint 2, where one front-end developer focused solely on the website design and user interactions, while the other serves as the bridge that linked the frontend to the backend, resulting in less intrusive and conflicting workflows and better cooperation and cohesion.

Another challenge that the front-end faced was the information that needed to be learned and used from numerous packages and their documentation. This was especially difficult due to the short time available for both research and implementation but was somewhat overcome by pair programming and sharing past knowledge between our two frontend developers. In hindsight, perhaps more research should have been conducted on which libraries and packages would meet our needs to remove the reliance on trial and error and wasted time in testing libraries which were eventually excluded from our final submission.

4.2 Backend

4.2.1 Recommendation Challenges

The First Challenge we come across is how to design an algorithm. Even though we know that there is an existing algorithm related to the recommendation, it's still hard for us to use it. The biggest problem, it's difficult to modify third-party implementations based on our business design. Furthermore, It is literally impossible for us to design a recommendation algorithm from scratch in such a short period of time.

After a discussion with Matt and a huge amount of time searching online, we finally decided to use TF-IDF. TF-IDF is used to discover document similarities. The reasons why we use it come as

follows. First of all, we only collect users' watchlists, for every element inside the user's watchlist, we find that the most relevant part is a movie plot. Although movie genres can be seen as another gauge of that, movie genres are not strong enough to represent one user's interest. Specifically, if we look into one user watchlist, we may know that this user may be into movies filled with gunshots and car racing by looking into the plots. We could know nothing more than "Action" if we only care about movie genres.

Another reason is that, by using plots, we can treat every movie plot as one "document" and our whole movie database as a huge collection of documents. Then TF-IDF can handle everything. We can use it to spot documents with high similarities and use them as a result of recommendations.

4.2.2 Configuration Challenges

Another challenge rises when we finish implementing the algorithm. Our backend is running on Java Code, but the algorithm code is on Python. We need to find a way to run Python code inside the Java program and return Python program output to Java.

Finally, we tackle this problem by, first, using command lines to execute the python files, and Second, building a stream between two programs so that the main program can capture another program's output. One trick we use here is that the python file does not need to output the correct format. Instead, it only needs to return a string of movie ids so that we can handle the format in the main program. The reason for that is, that Stream seems only to allow them to communicate in String other than different types. Using Java Spring boot to handle the format is much easier.

4.2.3 Transferring data from IMDb to Mongodb database

We use information from the movies on the IMDb website, but we cannot directly import JSON files directly into the database, we need to filter the useful data and not all. The solution was that we code in a file to optimise and add each movie data to the database, and we could run this code whenever we want to refresh the data. The data of the director, actor, and movie will be imported into the database.

5. User Manual

5.1. Starting the Backend:

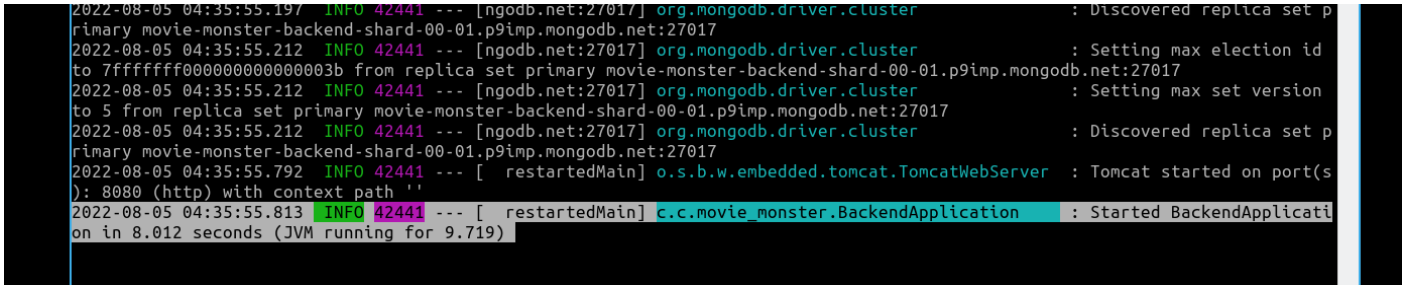
In the first of two terminals needed, clone the project repository to the directory ~/capstone-project-3900-f16a-impregnable.

Next, use the commands:

```
$ cd ~/capstone-project-3900-f16a-impregnable/backend
$ bash run_backend.sh 8080 --clean
```

The --clean option is necessary to update Lubuntu and install all the required packages for the backend when running the server for the first time.

Wait for the installation to finish and the backend server to start (10-15 min for the first time due to package updates and installation). This is indicated by a line similar to the highlighted text below being outputted in the terminal:



5.2. Starting the Frontend:

Once the backend is running, run the frontend server by entering the next set of commands.

```
$ cd ~/capstone-project-3900-f16a-impregnable/frontend
$ bash run_frontend.sh 3000 --clean
```

After 2-3 min, the website should be launched in a Browser (E.g. Firefox). You may see a white screen on startup - it will take an additional 1-2 min for the frontend to transpile and for the site to fully load for the first time. The frontend URL will be

- http://localhost:3000/capstone-project-3900-f16a-impregnable/

Section 2 of our report covers the features available as well as how a user can interact with the page. An available admin account can be accessed with the following credentials:

- Username: admin@gmail.com
- Password: 123456

5.3. Deployed Application (Cloud Hosting)

While this was not a requirement of the project, we have chosen to deploy our application to gather feedback from beta testers (users/friends who aren't one of our 5 developers).

If you are unable to start the application locally, you can access the cloud application through

- https://unsw-cse-comp3900-9900-22t2.github.io/capstone-project-3900-f16a-impregnable/

For the sake of completeness, the backend and MongoDB server was deployed at

- Backend: https://movie-monster-backend.herokuapp.com
- MongoDB: mongodb+srv://movie-monster-backend.p9imp.mongodb.net

Do note that it may take 30 seconds for the backend to boot up from idle after your first interaction with the frontend.

6. Reference

Alive-progress. (2022). Retrieved 4 August 2022, from https://pypi.org/project/alive-progress/#description

JSON Web Token - Wikipedia. (2022). Retrieved 4 August 2022, from https://en.wikipedia.org/wiki/JSON_Web_Token

Maven - Wikipedia. (2022). Retrieved 4 August 2022, fromhttps://en.wikipedia.org/wiki/Apache_Maven

MongoDB - Wikipedia. (2022). Retrieved 4 August 2022, from https://en.wikipedia.org/wiki/MongoDB

Pandas (2022). Retrieved 4 August 2022, from pandas-dev/pandas: Pandas 1.4.3 | Zenodo

Python Documentation. (2022). Retrieved 4 August 2022, from 3.10.6 Documentation (python.org)

Scikit-learn Documentation. (2022). Retrieved 4 August 2022, from About us — scikit-learn 1.1.1 documentation

Spring Boot - Introduction. (2022). Retrieved 4 August 2022, from https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm

Spring Framework. (2022). Retrieved 4 August 2022, from https://spring.io/projects/spring-framework/