CFG Degree Project Report

WANDERFLIX

GROUP 4

Jessica Gilbert,
Ozge Ozersahin,
Carey Yuen and
Nicola Ward

November 2022

Contents (Road map of the report)

1. Introduction

We are aiming to create a React application that users can interact with to explore the world through the medium of Movies and Television. We will ask the user to enter a destination of their choosing and the app will return Movies and TV Shows that are related to that destination. The architecture of the application will be a three-tier client-server model (see fig in section 3.2.)

1.1. Aims and objectives of the project

This project's goal was to use React to construct a comprehensive, usable, and functional online application. Our goal was to create a travel-based movie recommendation website application named Wanderflix which will allow users to search for movies based on a specific film location criteria.

In order to meet our aims and objectives we set out with the mission that our app must allow users to:

- Allows the user to enter a destination and it will return all movies, shows, documentaries shot/set there
- Save and unsave personalised movies watchlist
- Discover travel pages that allow users to discover some popular travel destinations for travel and related movies to watch

Along with these goals and objectives, we also had to work to integrate and expand skills gained during the CFG course, incorporating both Front and Back-end technologies such as SQL, React, JavaScript, HTML, and CSS, as well utilising other libraries where applicable.

2. Background

Wanderflix was inspired and created from our group's shared passion for travel and exploring different destinations around the globe. We wanted to create something that will allow the user to search for and browse a specially curated selection of movies and TV shows related to their desired destination.

We have designed Wanderflix to ask the user to sign in or sign up to use the service to its full capacity. This will allow them to save movies or shows to their 'Watchlist' and also provide a star-rating for the content they watch. Watchlist items and ratings will be stored in our Database, allowing the customer to view these again next time they sign in. Users can browse a selection of movies for their chosen destination which will have been filtered and returned using a 'keyword' search from 'The Movie DB' API. This will return any content where the movie description(blurb) features the destination that the user has searched for.

3. Specifications and Design

3.1. Requirements Technical and Non-Technical

3.1.1. Technical Requirements

Key system features:

- The system must allow for the user to enter a destination of their choosing into the search bar.
- The system must create and send a request to the TMDB API with the given search term.
- The system must present the results from the TMDB API to the user.
- The system must allow the user to sign-up and sign-in to the application.
- The user will have the option to save movies from the results to their Watchlist (the user must be signed up and signed-in to do this.)
- The system will store the users saved movies into the Watchlist database.
- The user will be able to rate the movies that they have watched.
- The system must be able to save the user ratings to the database.

3.1.2 Non-Technical Requirements

Usability

- The website should be intuitive and easy to use.
- The user should easily be able to search for movies and access their Watchlist.

Accessibility

- All images must have alt HTML tags for those using screen readers.

Security

- Users must enter the correct username and password in order to access their watchlist.
- We will store only necessary data to the database in order for the user to retrieve their saved watchlist and ratings.
- All of the personal user data that we store must be stored securely.

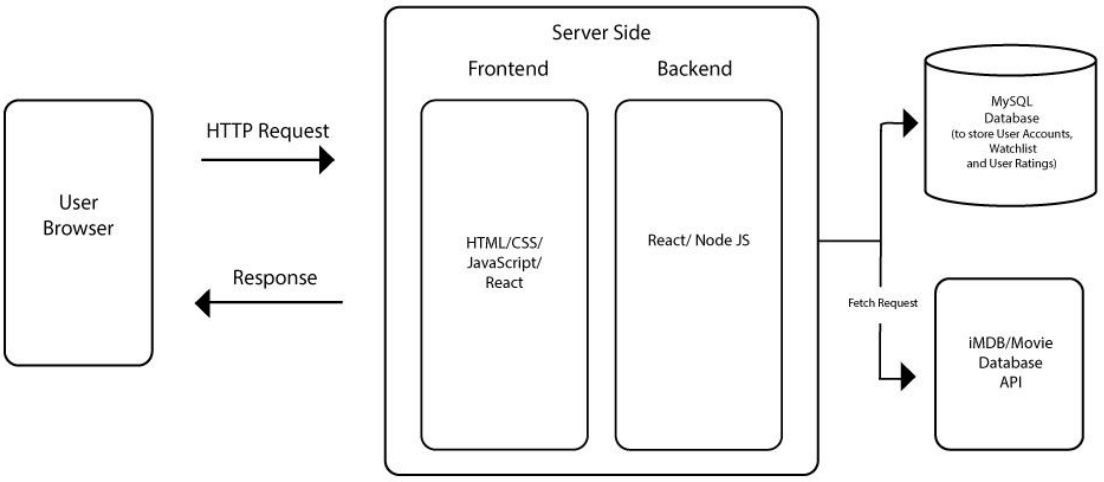
Documentation

- We will ensure that we use well documented code so that if another developer wishes to add/pull from the GitHub repository they are able to understand the various components of the site.

3.2 Design and Architecture

3.2.1. Three Tier client- server model

After discussing different options for the architecture of the project, we decided that the best way to successfully incorporate all of the features would be to use a three tier client-server model, using HTML/CSS/JavaScript within the React framework for the front-end and Node JS/React for the Backend of the program. The server would then connect to both TMDB Movie Database to fetch requests and our created MySQL database to store user information, watchlists and ratings.



3.2.2. Project Wireframe

After researching design heuristics, we developed a simple Wireframe for the project, mapping out where the key components of the site should be. We particularly focused on implementing a minimalist design aesthetic, only including on each page what we feel is necessary for the user to interact successfully with the program. Instead of using lengthy text-based instructions we have used visual clues such as action buttons to inform the user how to use the application

3.2.3 Project Storyboard & Design Principles

When designing the application we kept Nielsen's design heuristics at the forefront:

Visibility of system status

- We have designed the navigation bar to keep the user informed of where they are on the website by having the text of the link that is currently active displayed in green instead of white. The navigation bar appears on every page of the website so that the user can refer back to it at each stage.

Match between system and the real world

- To make sure that we are not over-complicating things for the user we have avoided overly technical expressions throughout, using clear and concise buttons and action points. We have also tried to use familiar icons such as the 'heart' icon next to the Wishlist link. The heart icon is a common feature in websites that allow users to save items to a 'Wishlist' or 'Favourites'.

User control and freedom

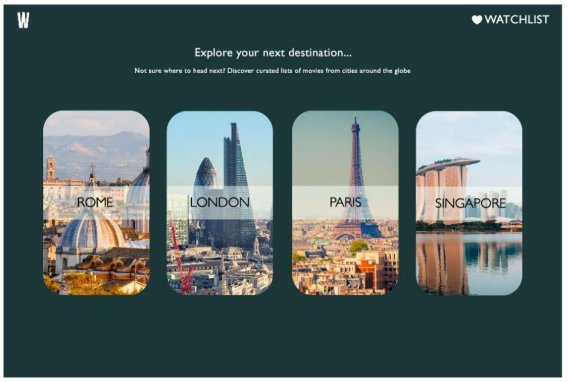
- Users sometimes want to start their journey over, or exit the page they are on, we have made this simple by providing a link back to the home page through our 'W' logo in the top-left corner of the screen. This is a familiar place for the home link to be found.

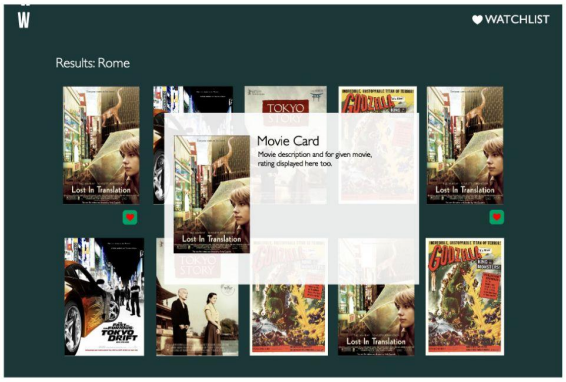
Error prevention

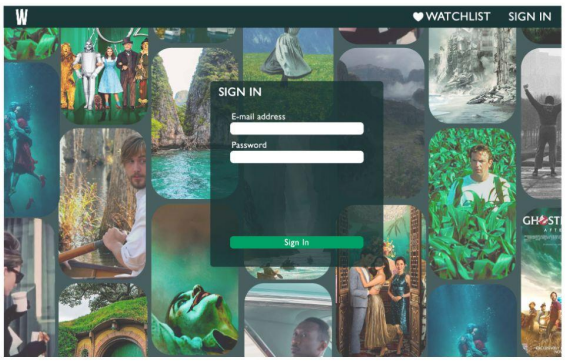
- To prevent errors before they happen, we looked at adding in features in the Sign-Up section, such as flagging that a password is not long enough or letting the user know that a field has been left blank.
- We prevented users that are already registered from re-registering. We also wrote in code that would avoid users adding the same movie twice to their watchlist.

Aesthetic and minimalist design

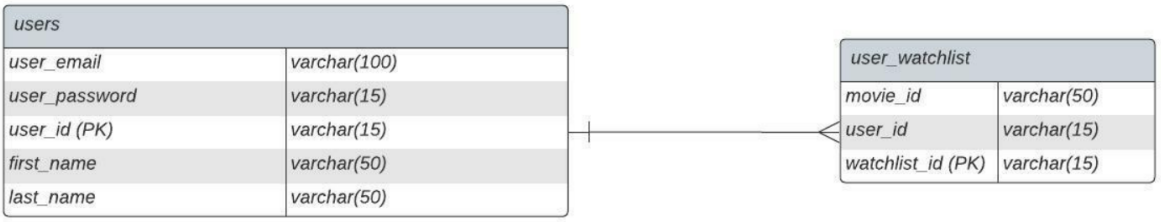
- We have implemented and utilised a strong but minimalist colour palette and design aesthetic throughout, only including on each page what we feel is necessary for the user to interact successfully with the program. We also chose to use a dark background with white text making it clear for the user to read the text. Instead of using lengthy text-based instructions we have used visual clues such as action buttons to inform the user how to use the application.







3.2.4 Database Schema Design



Our DB has two tables which are "users" and "user_watchlist". When users sign-in or sign up we store their information in the users table. When users add/remove movies from their watchlist we also keep this information in the user_watchlist table.

All DB information connects to the front end with APIs. These APIs Reference are:

- Post /sign-in
- Post /signup
- Post /watchlist
- Delete /watchlist
- Get /watchlist/{user_id}

4.1 Tools and Libraries

Throughout the project we utilised various tools and libraries, listed below;

Tools

- VS Code
- GitHub
- Postman
- Chrome Developer Tools

Libraries

- React
- React Router
- Express
- MySQL2
- Node JS
- Axios
- Jest
- Babel

4.2 Implementation Process

In order to create the application we started by researching various movie API's and decided that the best one to gain the information we required was the 'The Movie Database' API (TMDB). We learnt how to make calls to the API and fetch various parts of the Movie information and how to display it to the user in a clear and cohesive way. A search function was then created in order to connect to the API and let the user search for the destination of their choice.

Whilst developing this main feature we also worked on creating site navigation using React Hooks and creating other functional components such as the Sign-Up, Sign-In and Watchlist pages. The back-end database schema was created to hold user profile information and to allow users to sign-in and save their favourite movies. We created various back-end-points using JS Node that connected with the front-end components in order to make these features fully functional for the user.

Once we had successfully developed the main feature of the application that allows the user to search a destination of their choice and save items to a Watchlist, we decided to further the project by adding in an extra feature, allowing the user to use a 'Discover' page to browse, pre-selected movie collections from destinations around the world. We developed this by again utilising various React Hooks and JavaScript object mapping.

We gained knowledge about teamwork, task delegation, and setting realistic goals. By setting up a team repository, sharing files, establishing several GitHub branches, and merging them, we all took on the challenge of learning how GitHub operates. Being a cohesive team, being helpful and collaborative, and adhering to the Agile methodology was one of our team's finest accomplishments and we were able to communicate effectively together via regular meetings and daily updates. Everyone on our team learned a lot from working on this project, which was a great experience.

4.3 - Agile Development and Team Member Roles

We used an Agile method to develop, which was based on the notion of iterative development. As a remote team, we utilised tools like GitHub (where we would save the code and use Version Control), Slack (for constant communication and idea sharing), and Trello to convey our goals and code changes (to manage our projects and tasks). This provided accountability, clarity about what needed to be done, and allowed us to keep track of both our individual and group goals.

In summary, our four sprints took the rough approach of:

Sprint/Week 1 - Project set up including React app, GitHub, Slack, Trello, etc., learning React & planning design & architecture of overall site.

Sprint/Week 2 - Developing UI & UX design, wireframes, and overall site navigation. Creating mock data for MySQL.

Sprint/Week 3 - Developing core list functionality including connecting to MySQL, website navigation, features components, and styling. Building API endpoints with express.js.

Sprint/Week 4 - Testing, finalising core list functionality, project documentation, presentation PowerPoint, and final features styling.

Group 4 used a Trello board to assign, manage and review tasks within the project. The following headings were used to track tasks:

- Not Started
- In Progress
- Under Review
- Completed

You can view the Trello Board here https://trello.com/b/T1lvTCaP.

Each team member was responsible for assigning themselves a task and moving it across the board to completion. These were completed in week-long sprints and reviewed in the middle of each week to discuss any issues or barriers in the way of completion. These could be reassigned at any point throughout the project if limited progress was being made or support from another member was required.

Flexibility and adaptability were important considerations for us, and we made sure to use different technologies to overcome obstacles (e.g., for integrating with other libraries in React: React Router to navigate to different webpages; React Bootstrap Modal to reuse and integrate UI components, such as the pop-up window on the result page). We honed our teamwork skills by conducting frequent code reviews on GitHub in a flexible and encouraging manner. We used an Agile methodology and this was essential to the project's success.

Code Reviews

The GitHub repository was set up to require a code review from other collaborators before each Pull Request could be merged to 'main'. This meant that each component was reviewed and tested by other team members before being committed. In doing so, this ensured that each member's code contributions worked in unison with existing code and avoided any major changes prior to project completion.

Refactoring

At the end of the project, the group was involved in code refactoring. Each file was reviewed to ensure that formatting was consistent using the 'prettier' plugin. Any unnecessary lines of code were removed and styling was reviewed. Variable and function names were also renamed, where required, to ensure code was readable.

As an example all the back-end codes were just in one file which was mysql.js. To make it more understandable and to make the code clearer, we made structural improvements and then separated it into different files; we have controller, config and route files in the back-end. Herewith refactoring made it easier for testing and changing the code.

4.4. Implementation challenges

Throughout the project, we encountered a few difficulties with the development of the application. A particular challenge that we encountered was how to store and pass variables, state and props between routes within React. We overcame this problem by researching, testing and utilising various React Hooks, such as useEffect, useState and useNavigate.

One of the challenges we faced when developing the backend was unit test integration. We used Jest for testing Express.js and Jest was unable to validate test cases because of JavaScript syntax standards. To overcome this we had to use Babel to convert and arrange its configuration, which was a bit time consuming, however the issues were fixed and testing was completed for the Express.js API's. Another challenge we faced was dealing with code conflicts, because everyone is working on the same project and sometimes within the same file, conflicts arose, however we overcame this issue by investigating the latest implemented codes.

5. Testing and Evaluation

5. Functional and user testing

After completing the project, we completed both functional and usability testing to ensure the functions behaved in the intended way and that the user was able to successfully use the app for its required purpose (to search for and browse movies related to different destinations).

Usability and user testing were carried out to assess the navigational flow, content and page layout that the user would encounter. The completed app was given to a variety of independent users for them to test the features and then provide feedback. Areas of focus were to determine whether the UI was intuitive (e.g did the user know what to expect when clicking the heart button and was the response clear), whether they ran into any bugs or how they reacted to being presented with buttons or requests for an input.

We have also implemented various unit tests, testing some of the main individual functions and components in order to guarantee that they are operating properly. We have written unit tests for the main Search bar to check that it renders correctly and that the user can see the input on the screen when typing into the bar. Unit-tests for the Navigation bar were written in order to check that the links navigate to the correct pages and that the logo renders correctly. We also wrote and implemented unit-tests for other important parts of the site such as the sign-in and out pages.

For the backend testing, we also made tests for the sign-in, sign-up, and watchlist pages. We checked that the inputs that came from the front end can reach the database and work without any problems and all of the tests passed.

5.2. System limitations

If we had more time we could add in a feature that allows the user to come-back to the watchlist after watching the film and give the movie a rating, storing this into the DB for their future reference.

For any long-term project, we would need to develop a group methodology with the aim of reducing coding conflicts on GitHub when multiple team members are working on the same code.

6 . Conclusion

The result of the team project was the successful development of a React application that allows the user to search for Movie recommendations based on their own choice of destination, allowing users to discover new places to travel through the medium of film. The application also allows users to browse through pre-curated lists of Movies in order to discover new movies and places. We successfully connected all aspects of the Front and Back end components creating a fully-functioning application.