Advanced JavaScript

Continuous Assessment 1 – React Web App Development

Table of Contents

[Introduction 3](#_Toc453695)

[User Stories 4](#_Toc453696)

[Architecture 5](#_Toc453697)

[Application Functionality 6](#_Toc453698)

[React 6](#_Toc453699)

[Functionality 6](#_Toc453700)

[Overview 6](#_Toc453701)

[Features 7](#_Toc453702)

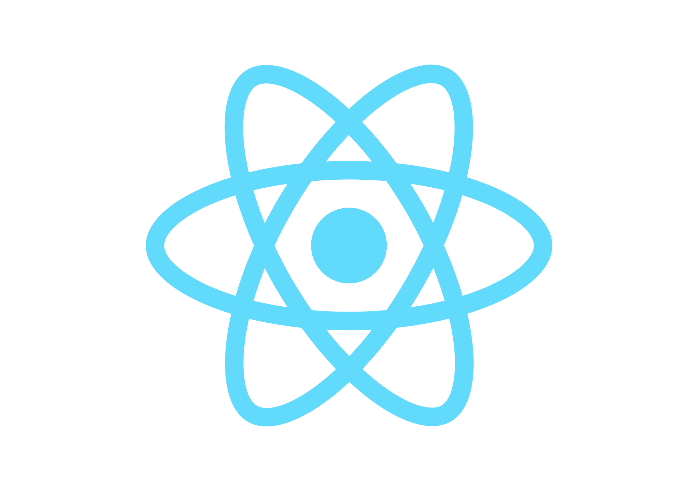
[Conclusion 8](#_Toc453703)

[References 8](#_Toc453704)

# Introduction

For this assignment the task was to create a web app using the React front-end JavaScript library. In addition to the React library, the application had to utilise an API to consume data.

This application uses the React JavaScript library and connects to a movie database to retrieve data and present it to the user. The database contains thousands of films each with their own attributes and values. The application has a filter system allowing the user to search for films by their title.



# User Stories

**User stories** are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system [1]. User stories use the following format:

*As a* ***< type of user >,*** *I want* ***< some goal >*** *so that* ***< some reason >***

I created the following the user stories for the application.

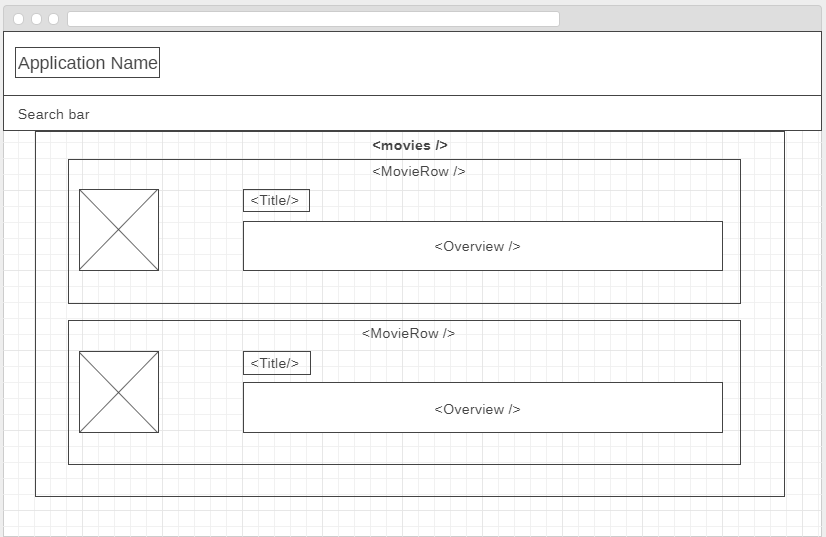
* As a user interested in movies, I want to find out more about them in the one place so that I don’t have to search all over the internet.
* As a film critic, I want to find a large directory of films so that I can review them.
* As a Netflix user, I want to find movie reviews so that I can watch well-received films.

# Architecture

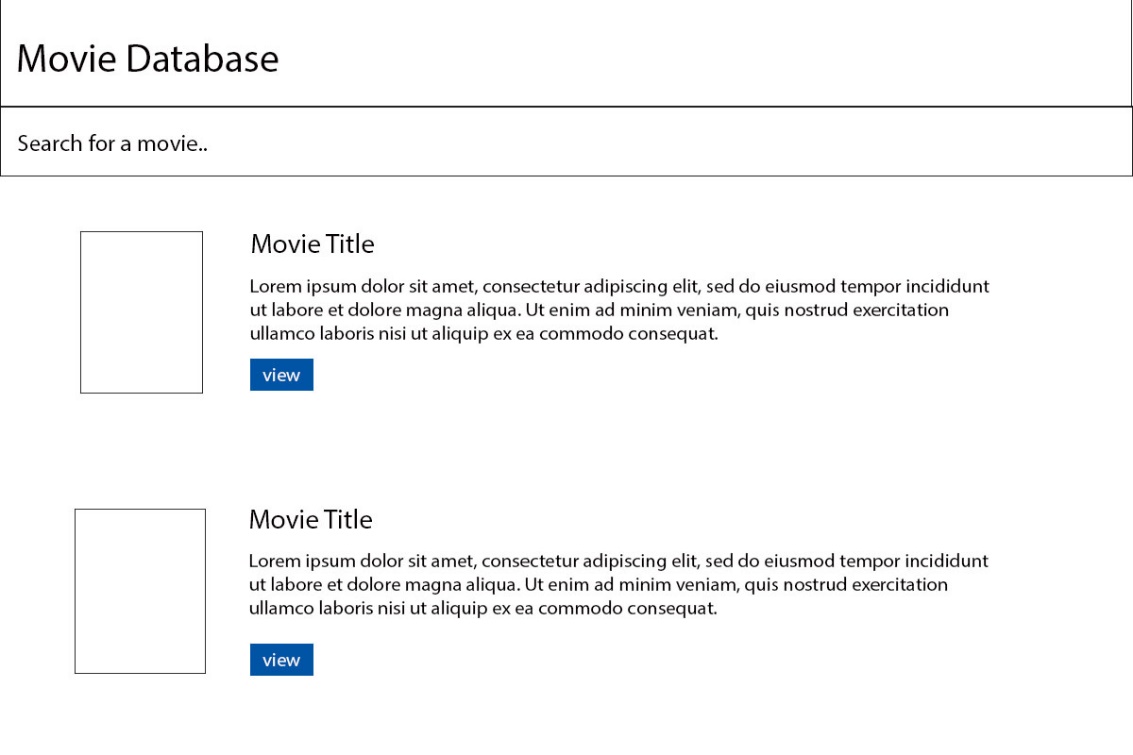
React uses components in order to display different UI elements. This application has one component called MovieRow.js which is responsible for displaying the relevant attributes for each movie object, i.e. movie title, overview and poster.

As seen in application’s wireframe below, the movie poster in shown on the left (stored in a separate table data element) with the title and overview displayed on the right.

*Figure 2.0 – Application wireframe showing React component*



*Figure 2.1 – Application UI*



# Application Functionality

## React

React is a JavaScript library for building user interfaces (UI) in applications [2]. Developed by Facebook, it is one of the most popular JavaScript libraries for UIs.

React applications are built using components which are essentially a part of the UI. When building applications with React, it creates a number of individual and reusable components which are used to build complex user interfaces.

React applications will have at least one component known as the root component. The root component represents the entire application and contains the other child components; these child components inherit from their parent component. A React app could be described as a tree of components, with each component representing a part of UI.

A component is generally implemented as a JavaScript class with two main parts; a state and render method.

* The state is the data we want to display when the component is rendered
* The render method is responsible for describing what the UI should look like.

Once the render method runs, it creates a React Element which is a simple JavaScript object that maps the DOM element. Although it’s not technically a “real DOM element”, it represents that DOM element in memory which can be referred to as the Virtual DOM.

When you change the state of a component, we get a new react element. React will then compare this new element and its children with the previous one, it figures out what has changed and then updates a part of the real DOM to keep it in synch with the Virtual DOM. The Virtual DOM is one of the main benefits to using React as it allows for an overall improved application performance.

## Functionality

### Overview

The application works by searching through a movie database API and retrieving information about movies. Included with each movie object is a view button which brings the user to the movie URL for that movie.

The API is called The Movie Database and requires users to create an account in order to obtain an API key for retrieving data from the API.

The API returns the movie title, the movie overview and the movie poster in an array. When entering a film title into the search bar, the application pings the Movie DB API, returns a large amount of data and then this data is rendered into a table list view. The table list view is composed of one table row and two table data elements.

As mentioned earlier, React applications utilise states. This application has two states and React allows you to alternate between both states with ease:

* finished loading state - when all movies are returned after completing a search
* empty state - no content in search bar and therefore no results

The search bar is displayed in the top right corner of the navbar and the relevant code is stored in a file an App.js file which contains the root class called “App”, extending to the Component class. Within the App class is an AJAX call to the API which uses an API URL link, a unique API key and a variable called “searchTerm” which contains the user input from the search bar. If the app is successful in obtaining the relevant data, then it outputs a success message to the console.

### Features

**Components** – The application utilises several components in the UI:

* Movie Poster
* Movie Title
* Movie Overview

Components allow for UI elements to be split into independent and reusable parts. A component will accept a props (properties) object argument and as a result, will return a React element (i.e. the desired UI output). Components are advantageous in that they allow for reusability and reduce the need for excess amounts of repetitive code.

**States –** The application has two states:

* Finished loading state
* Empty state

These states are utilised in reference to the search box in the application’s navbar. The finished loading state occurs after the user searches for a film and the app has returned the data, i.e. completing a search. The empty state is active when the user has not entered any input into the search and box and as a result, there is no data returned. States are beneficial in that they allow you to control what happens at different stages of interaction with the application.

**Props –** Props (short for properties or parameters) allow for the customisation of components. This in turn makes components more versatile as some components can contain props that other components don’t have.

In this application, props are used to for four movie components; id, title, overview and poster.

*Figure 3.0 – Props React Code*



**CSS (Basic Styling) –** The application is styled using an internal CSS file, App.css. Elements in the application are styled using their classes and ids and are styled for both interactive functionality and appearance purposes.

An example of an interactive CSS feature in the app is the enlargement of the movie poster on hover, which is done using the transform tool (changes scale size of image). Another feature is the fixed navbar which stays in place while the page is scrolled down.

An attempt was made to incorporate a scroll to top button, and although it generated a visual of the button, it resulted in errors which could not be fixed.

# Conclusion

The result of this process was a React web application that incorporates a movie database API, allows users to explore that API via the filter system and also view information about each film.

I was pleased with the overall result of the process and learnt an invaluable amount of information about the React framework and JavaScript frameworks as a whole.

# References

[1] Mountain Goat Software, User Stories

<https://www.mountaingoatsoftware.com/agile/user-stories>

[2] React, Wikipedia

<https://en.wikipedia.org/wiki/React_(JavaScript_library)>