**Deliverable 2 - Greatest Group Ever**

# **Software Architecture**

## **External data sources**

There are a variety of external data sources that are employed in this project. These data sources are accessed by using a variety of APIs and web-scraping tools. We have accessed these data sources in order to provide more information to the user regarding the movies, allowing them to make a more informed choice about whether they would like to view the movie. The combination of APIS and webscraping has been integrated as part of the backend.

1. **APIs**
   1. **Youtube Official API:** Will be used to get the trailer and price for the requested movie.
   2. **OMDB API**: Will be used to obtain general information for movies such as the title, cast, runtime, synopsis and user reviews.
   3. **iTunes API:** Will be used to retrieve the price information to view the movie on ITunes Store.
   4. **Hulu API**: Will be used to retrieve the price information to view the movie on Hulu.
   5. **Google API:** Will be used to log in the user.
2. **Web scraping**

Web-Scraping is used in order to obtain the price for watching the given movie on various platforms such as Google Play, Youtube and Vudu. The sites that were web-scraped were:

* 1. Google Play store
  2. Youtube.com search page
  3. Vudu.com

## **Software component diagram and external data sources**

We made two software component diagrams. The first is a more general diagram that showcases the different external sources we are using, whereas the second diagram is more in detail and shows exactly how each component interacts with each other including the frontend, backend and different APIs.

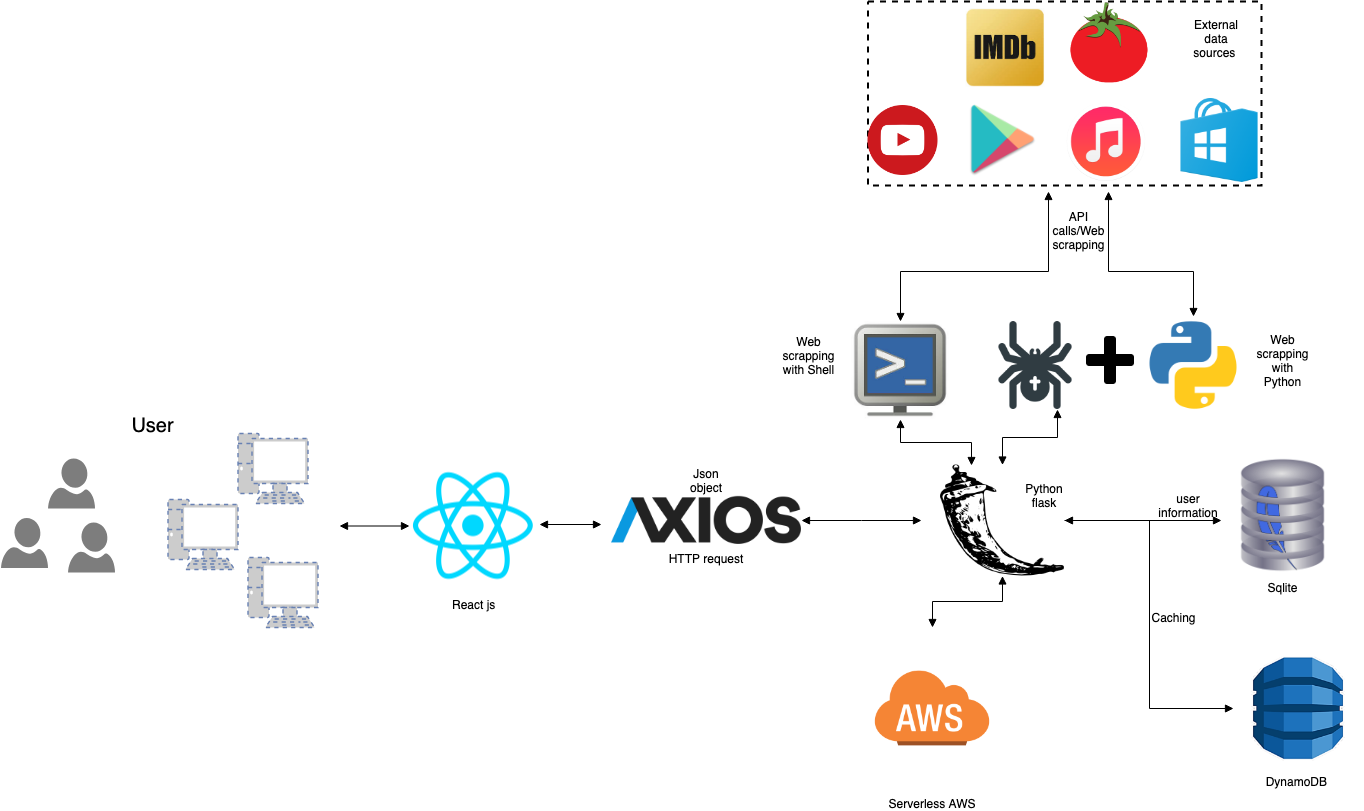
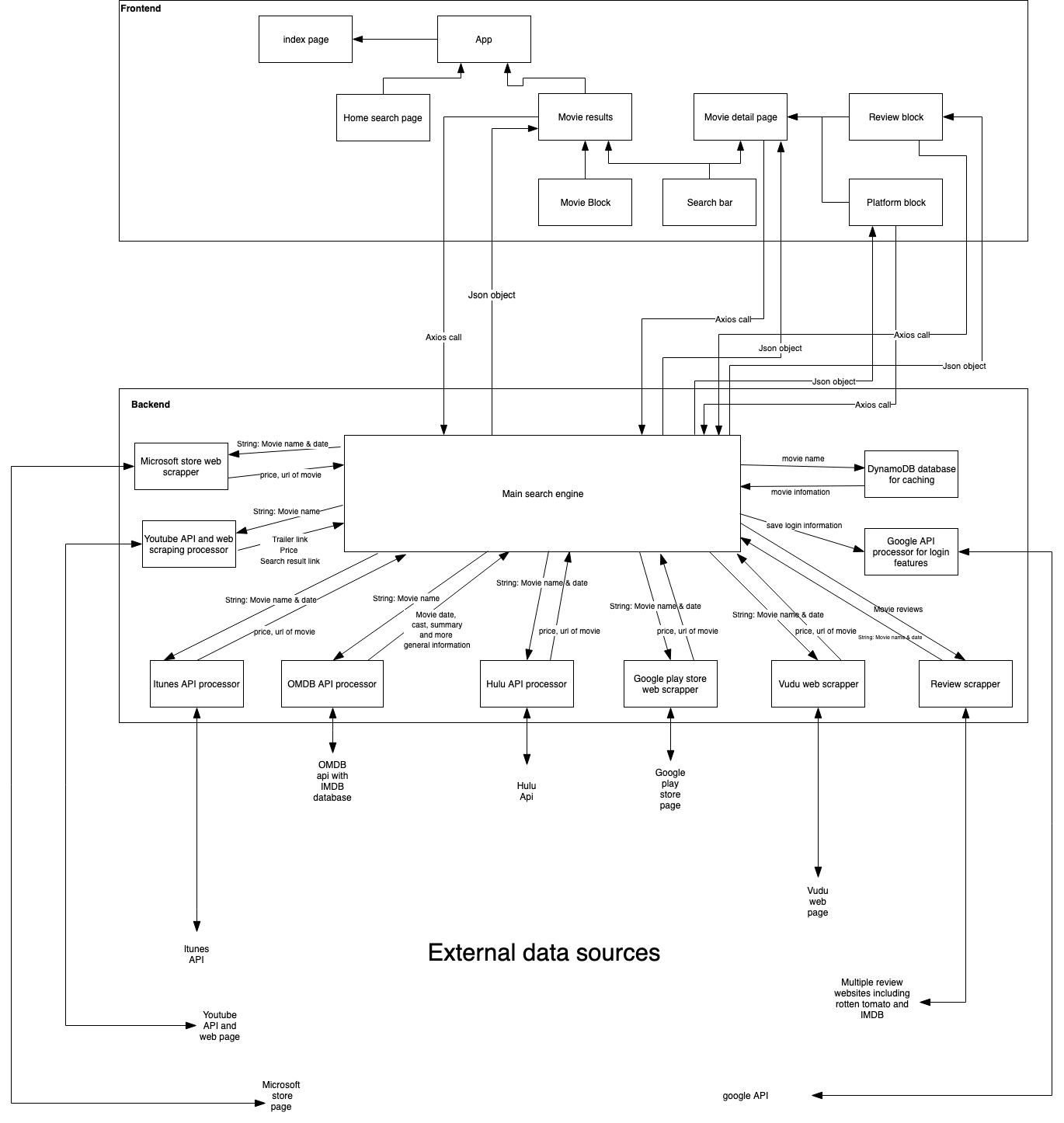


Figure 1.1: General Software Component Diagram

Figure 1.2: Detailed Software Component Diagram

## **Choice of languages and related web stacks**

**Frontend**

For the frontend, we are employing Javascript along with the corresponding React JS library. For the layout and styling, we will be using HTML/CSS along with the Material Design CSS library, in order to improve the UI and UX of our website.

**Middleware**

Axios middlewares are used by the frontend react to communicate with backend through HTTP requests.

**Backend**

We decide to use Flask and Python in conjunction with the Restful API to build up our backend so that we can convert Python class data to JSON. This is needed because our frontend, that is build with React needs JSON to update component data for the movie.

All the API calls to external data sources will be implemented in Python.

Most of the web scraping will be done with Python employing the Beautiful Soup library. However, the Youtube Web scraping will be done using Shell and Python.

## **Platform**

The final system can run on any operating system. This includes both PC, tablets and mobiles. We are planning on making the web app as user friendly as possible, employing a responsive user design style, so that it can be viewed with easy on any screen size or platform.

Furthermore, all the required packages are listed in the JSON file

## **Benefit of architectural choice**

* **HTML and CSS with Material UI and Material Design libraries**: For the frontend of our website, we agreed that it would be imperative for the layout to be simple, clean and aesthetic in order to make ensure that we had a good UX. This especially applies for users that are not used to navigating a movie database website. Thus, with the obvious choices of HTML5 and CSS3, we decided to use the Material Design CSS library in order to easily implement the styling for the website. Furthermore, we employed Material UI for some components as it works extremely well with React.
* **React JS**: Since we also have a lot of components, we decided that it would be better if we employed React JS along with the above components. React is an open-source JS library that allows for easily scalable websites to be built by creating reusable UI components.
* **Axios**: Axios can be simply imported into JavaScript to make http request with the backend API. This will simplify the procedure to update the state of the frontend component according to the response from backend.
* **Python flask with RESTful API** : Flask is an microframework for building up backend endpoints, which can be simply imported without the requirement of any tools or library. It is extremely useful for us to build our backend server with it. We pass JSON objects to our frontend in order to dynamically load components, however, there are no built-in python features that allow us to interchange between Python classes and JSON objects. For this reason, we decided to use RESTful API library for efficiency.
* **Beautiful Soup with Python and Shell for Web Scraping**: Since some online streaming platforms do not provide API directly for their video information, we decide to integrate some web scraping feature into our system. Beautiful Soup and Shell allow us to scan through various HTML tags on a given website for specific price information (if available) and display it to the user.
* At the prototype stage of the project, the whole program will be mainly based on MacOs or Linux system. Meanwhile, we are looking into moving the web stack on to an serverless framework with AWS services, in order to achieve functionality and reduce the maintenance cost on the server side over the longer term

# 

# **Initial software design**

**List of prioritised features**

Upon a series of meetings, we decided that we need to prioritise of features based on their relevance. The following requirements will be prioritised, using the MoSCoW notation. They are divided based upon their categories.

* (a) Must Have
* (b) = Should have,
* (c) = Could have
* (d) Won’t have

**1.Search Engine**

1.1 Search by movie title: **(a)**

1.2 Preference Filter (such as: in theatre, upcoming): **(b)**

**2. Display movie information**

View all related information of the movie on the movie info page (title, poster, etc.):

2.1 Display the title and poster of the movie: **(a)**

2.2 Display the cast and director of a movie: **(a)**

2.3 Display a synopsis of the movie: **(a)**

2.4 A clearly stated release date of a movie: **(a)**

2.5 Show an accurate release date for an upcoming movie or a date of release for an existing movie: **(b)**

2.1.5 State the viewing time of a movie: **(b)**

2.1.6 Display the age restrictions of a movie: **(b)**

2.1.7 Show the link to the Wikipedia page for the movie: **(a)**

2.1.8 Show the promotional trailers of a movie: **(b)**

2.1.9 Provide a link to Youtube so that the user can watch more trailer clips of a movie: **(b)**

2.1.10 Display a list of the top three reviews of a movie: **(b)**

2.1.11 Display a link to the streaming platforms on which the movie is available to watch: **(a)**

2.1.12 State and compare the cost of watching a movie on the various platforms: **(b)**

2.1.13 Display links to the IMDb and Rotten Tomatoes webpages for a movie if the movie has already been released: **(b)**

2.1.14 Show the IMDb and Rotten Tomatoes ratings for an existing movie: **(b)**

**3.User Account management**

Features for a user to manage their account

3.1 Sign up: : **(b)**

3.2 Login: **(b)**

3.3 Logout: **(b)**

3.4 Maintain login status: **(b)**

3.5 Third-party login**: (c)**

3.6 Change username, password or email address: **(b)**

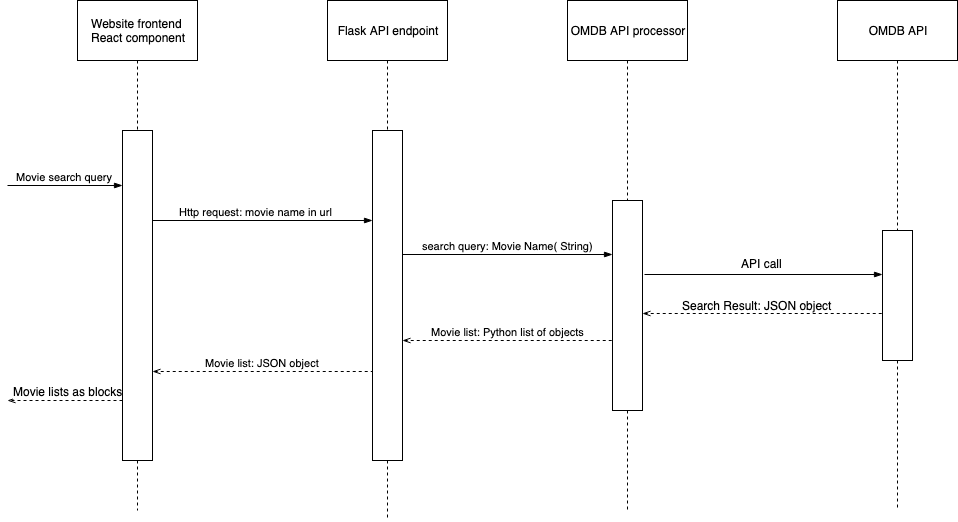
### 

4. **Integration with serverless framework - AWS: (c)**

**3. Sequence diagram and Updated Use Cases**

**3.1 Movie search Use case**

This use case allows a user to search for a movie on the homepage by its name. Upon clicking and entering text into the search bar, then I get a list of movies matching the search term. (Including the result page,name and releasing date). All use cases under the movie search use case follows the basic overview in the sequence diagram below.

Figure 3.1 Sequence diagram for movie search use case

**3.1 Movie Information Use case**

This use case allows the user to obtain general information of a movie. Upon clicking on a given movie on the search result page, the user is redirected to a page for the given movie. On this page, they are given all relevant information for the movie such as as its poster, title, cast, director, viewing time and summary. This allows the user to know more about the movie and make an informed decision on whether they want to view it or not.

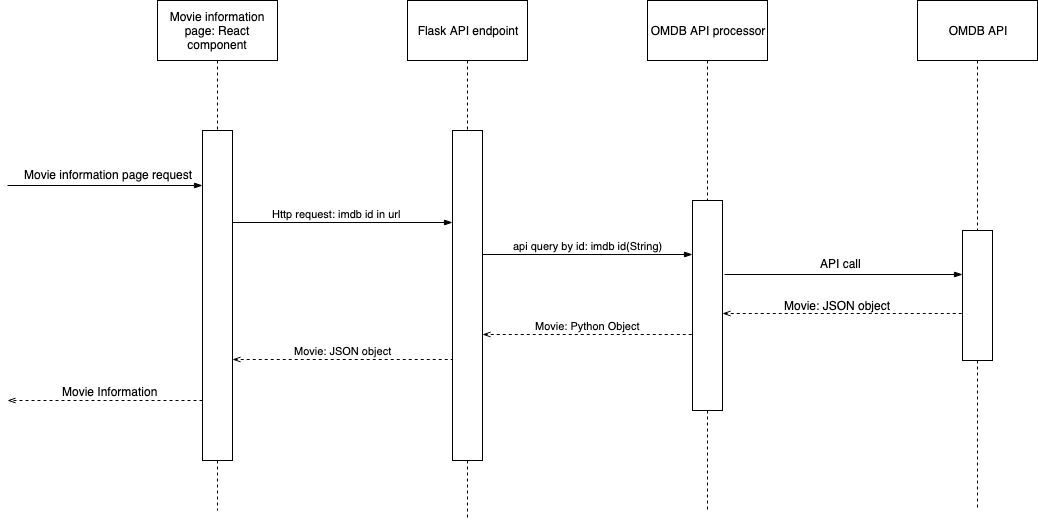
****

Figure 3.2 Sequence diagram for movie information use case

**3.2, 3.3: Movie Streaming Use Case** and **Movie Price Use Case**

This use case allows the user to view the streaming platforms that are available for the user to watch the movie legally. Given that the user is on the information page for a specific movie, they can view the streaming platforms that are available to watch the movie on legally. These platforms currently include ITunes, YouTube and Google Play. More platforms will be added in the following weeks, such as Vudu and Hulu.

When the user clicks on a link for any of the streaming platforms available, it redirects them to a page where they can view the movie on the website. Under the platforms, the user can also view the price for watching a movie on different platforms, if any.This will allow the user to determine the most financially affordable platform to watch the movie.

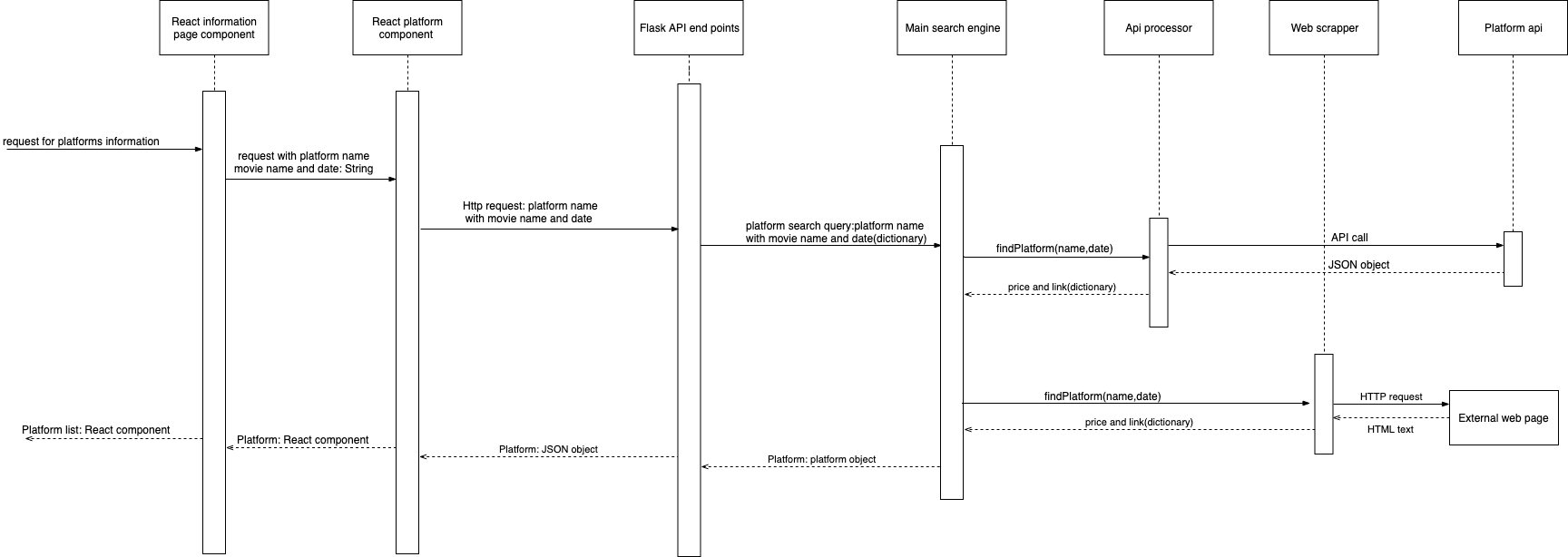
****

Figure 3.4 Sequence diagram for movie pricing and streaming use cases