XBMC3014N Internet and Web Development

Group Project – Prototype Guideline Documentation

Group 7

Table of Content

1.0 Design and Usability Elements

1.1 Purpose and Goals:

a. Clearly define the purpose of your website and its goals.

b. Understand your target audience and tailor the design to their needs.

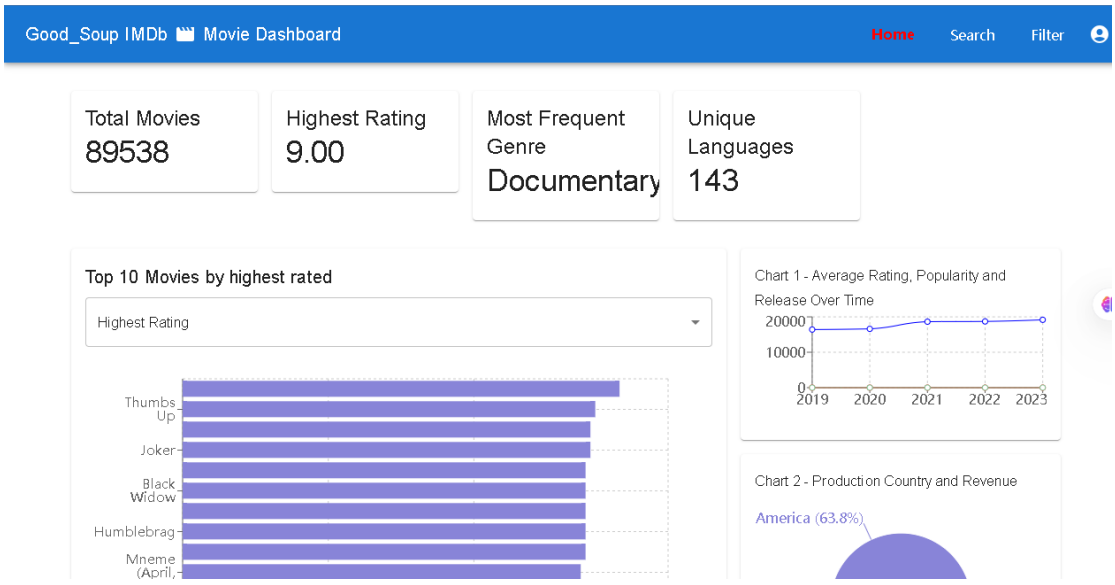

Diagram 1.1.1: This screenshot shows clear information of this dashboard is a movie dashboard and align to the target audience which needs a clear, simple and easy navigate dashboard. Things to be noted:

- The KPI cards placed at top of the dashboard
- Movies (charts) can be generated based on the dropdown options (e.g highest rating, popular, year etc)
- Side charts (additional info/graphs) are also generated based the options selected for overall view.

1.2 Clarity:

a. The dashboard layout clear and intuitive.

b. Allowing users to easily navigate.

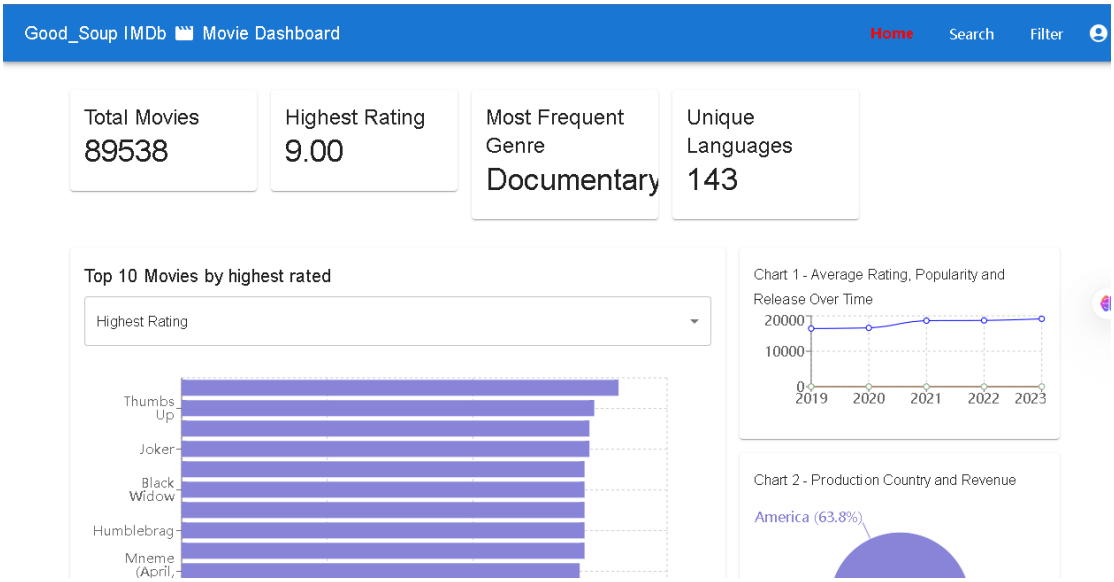

Diagram 1.2.1: This screenshot shows a clean and intuitive layout with a red color highlighted when user located on the page, allowing user easy to navigate.

- Shows the contrast highlight in the dashboard for better navigation

1.3 Responsive Design:

a. Ensure your website is responsive and works well on various devices (desktops, tablets, smartphones).

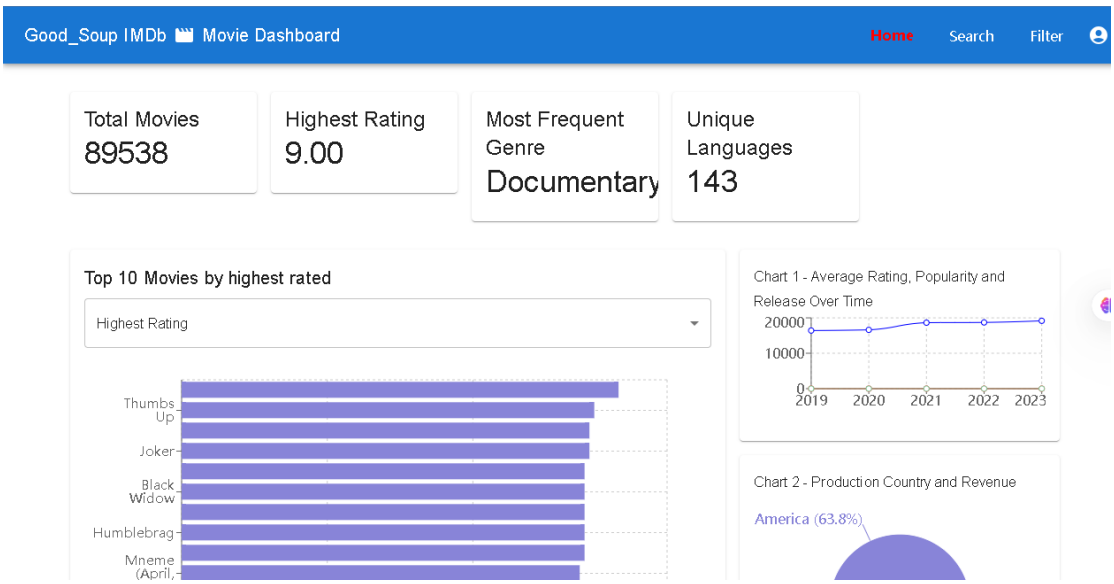

Diagram 1.3.1: This screenshot shows the website works well on laptop screen ratio.

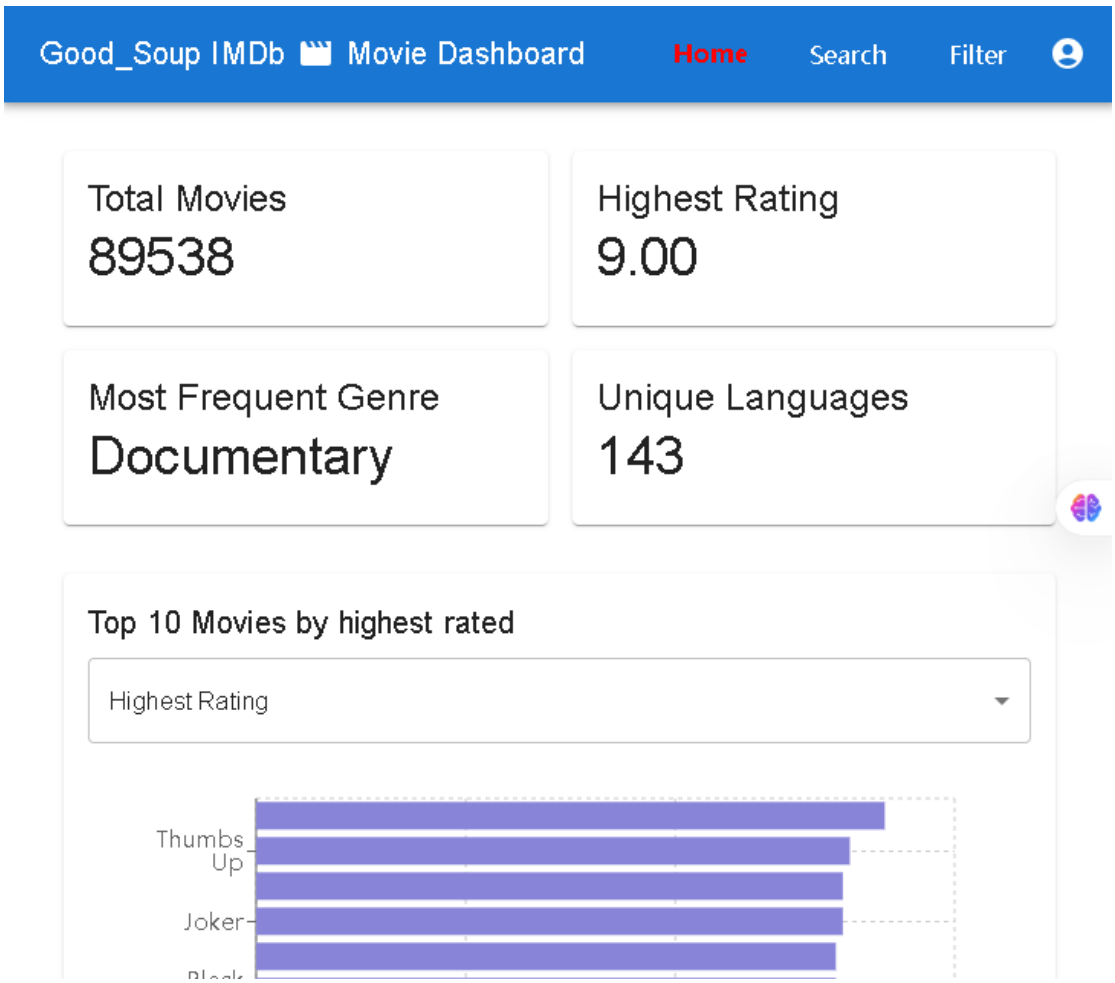

Diagram 1.3.2: This screenshot shows the website works well on phone screen ratio.

- Proves consistency with the shapes of the KPI cards and provides clean interface.

1.4 Aesthetic Appeal:

a. Hierarchy of information & proper use of typography.

b. Use a cohesive colour scheme that aligns with your brand and is visually appealing.

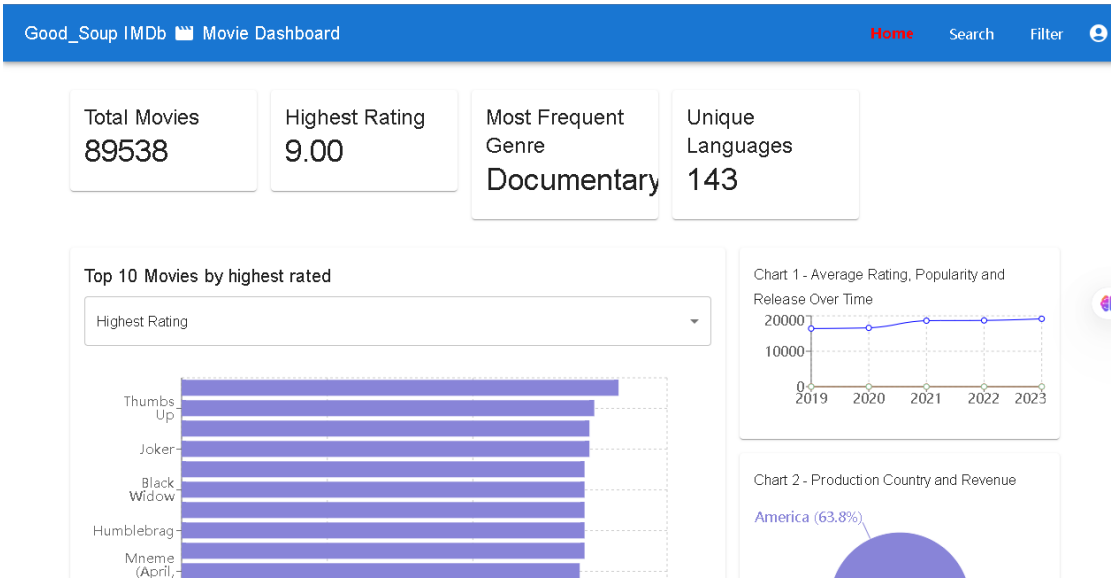

Diagram 1.4.1: This screenshot shows the layout of dashboard provided a hierarchical of information displayed, color and branding.

- Branding – main colors are blue as the main, white as dashboard background, purple as main color for charts and red for highlighting main parts.

1.5 Images and Graphics:

a. Use images and graphics that enhance the visual appeal.

b. Optimize images for fast loading times.

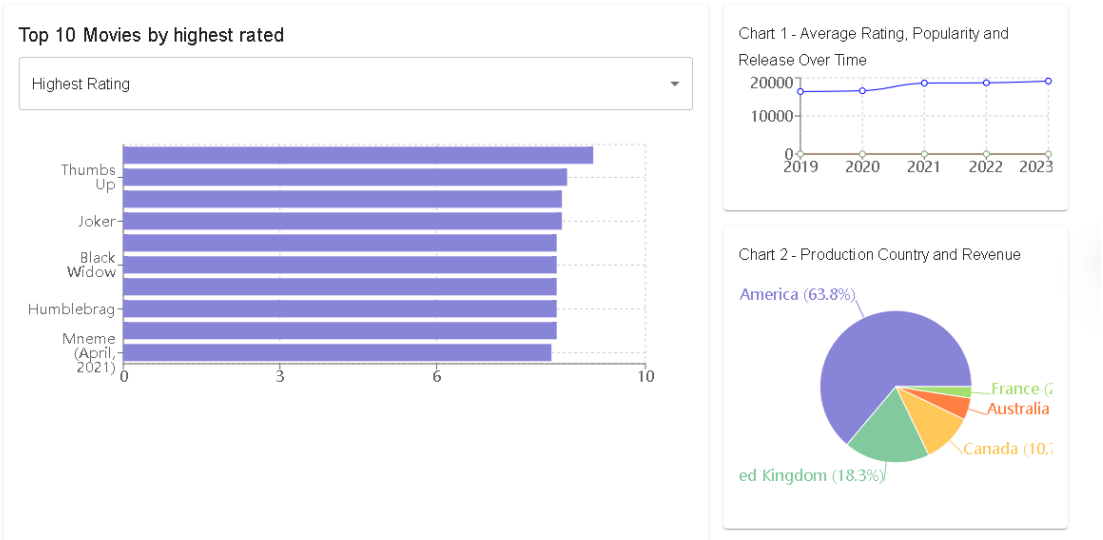

Diagram 1.5.1: This screenshot shows the use of graphic visualization to visualize the data.

2.0 Functionality & Interactivity

2.1 Filtering: Can users filter data effectively (e.g., by genre, year, rating) to explore the dataset?

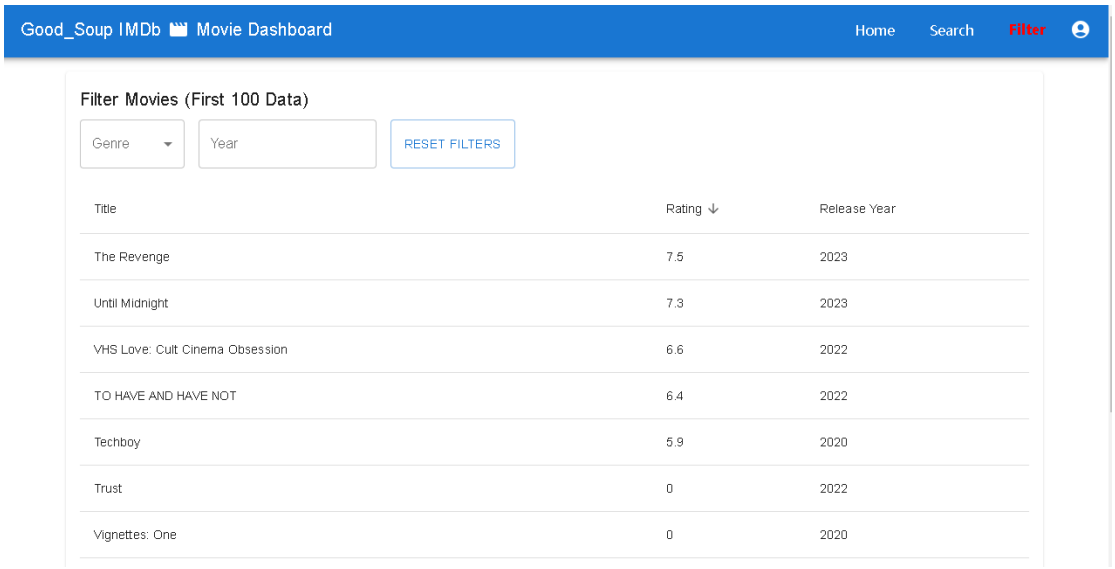

Diagram 2.1.1: This is a filter section provided to users to explore the dataset using type of genre and the year.

2.2 Sorting: Are users able to sort data based on various criteria (e.g., highest-rated movies, alphabetical order)?

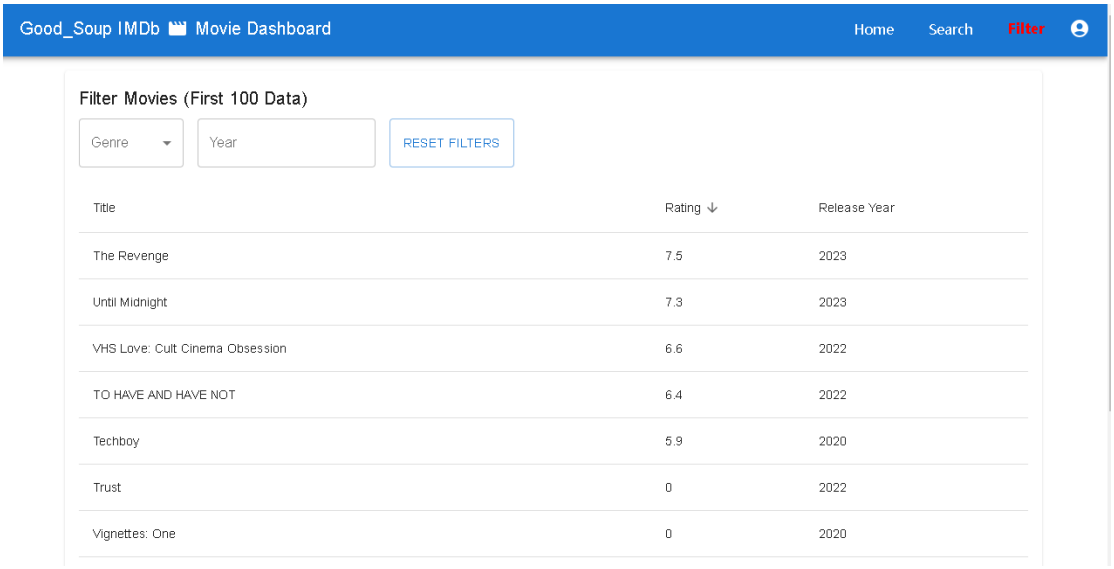

Diagram 2.2.1: This is a filter section provided to users to explore the dataset with the title is clickable to sort based on alphabetical descending or ascending order, rating is clickable to sort based on rating descending or ascending order, release year is clickable to sort based

on year descending or ascending order.

2.3 Search Functionality: Does the dashboard include a search feature for finding specific movies, actors, or directors?

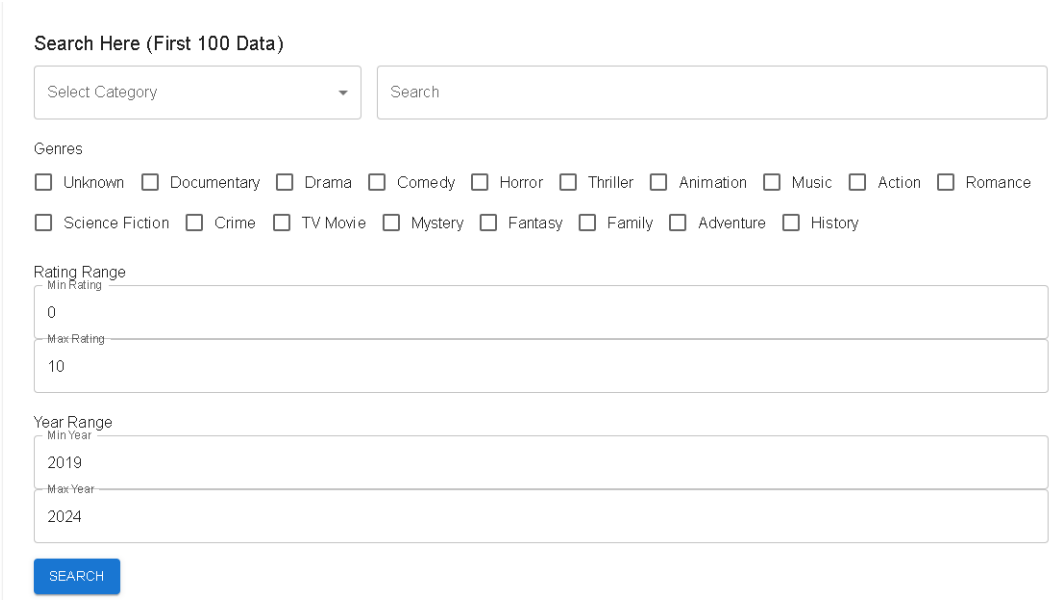

Diagram 2.3.1: This is a search function that provides a detailed search, using selected category, keyword that matched, type of genres, rating range and year range.

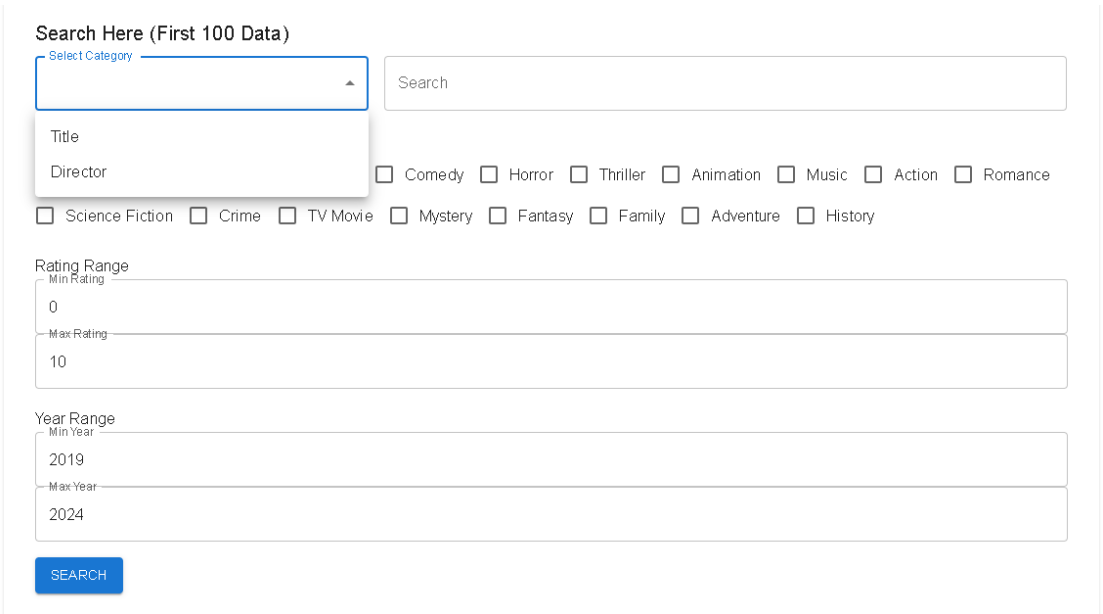

Diagram 2.3.2: This is a screenshot showing that can search based on selected category like Title or Director.

2.4 Dynamic Interactions: Are interactive elements, such as dropdowns or sliders, functioning smoothly without lag?

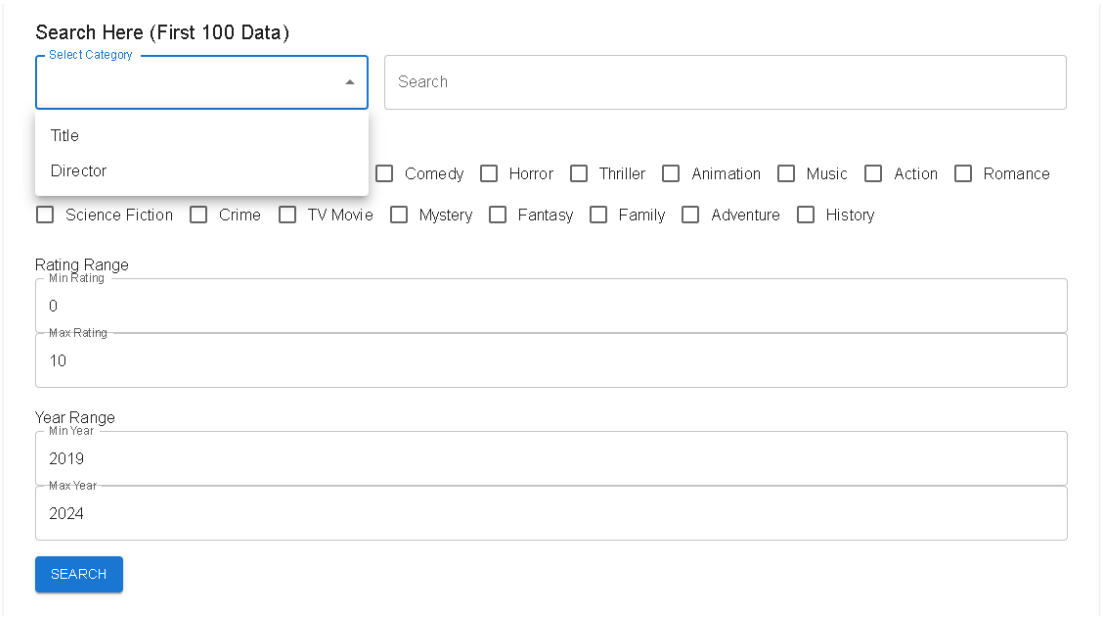

Diagram 2.4.1: This is a screenshot showing that there is a dropdown box in the search function.

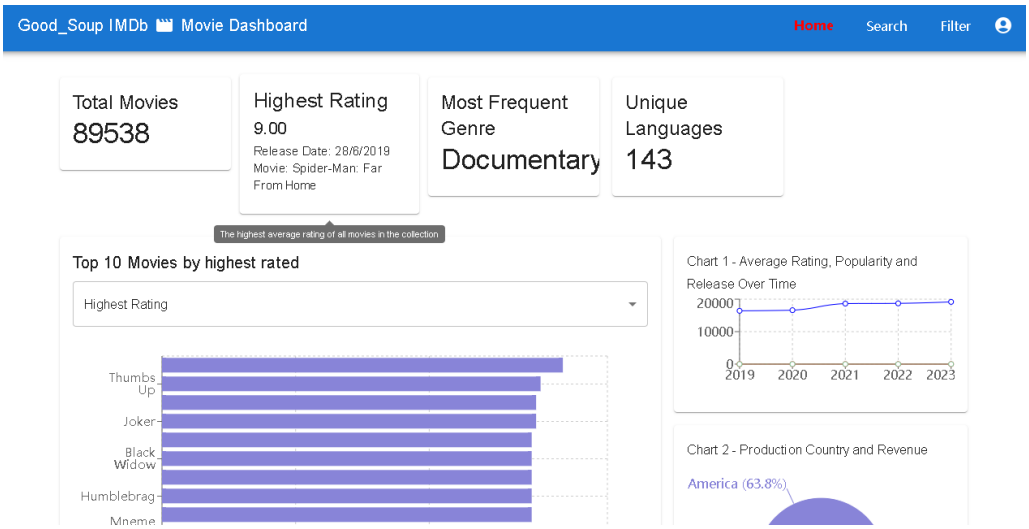

Diagram 2.4.2: This is a screenshot showing that there is a hover effect that guides user to explore the KPI Cards with a more information to facilitate their exploration.

2.5 User Feedback: Is there feedback (e.g., loading indicators, messages) provided to users when actions are performed?

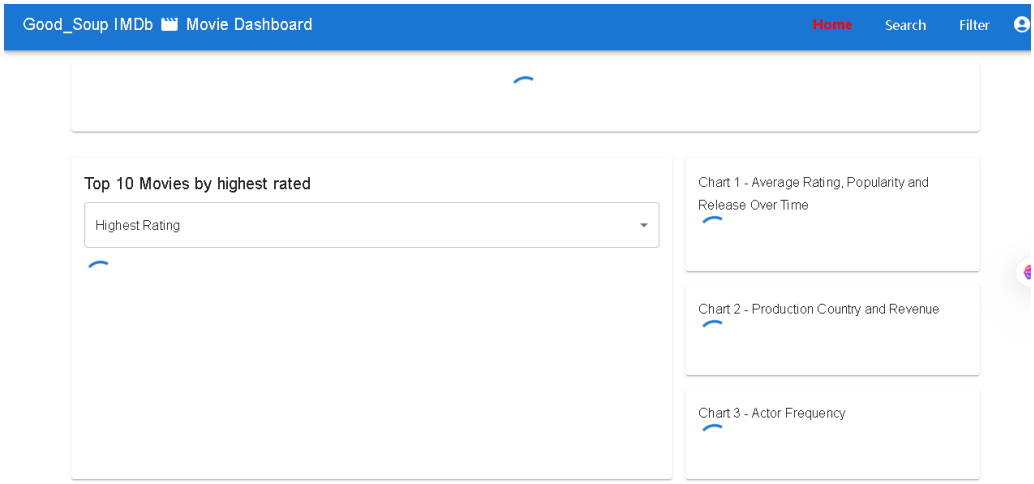

Diagram 2.5.1: This is a screenshot shows that a Circular Process is running when the data is loading.

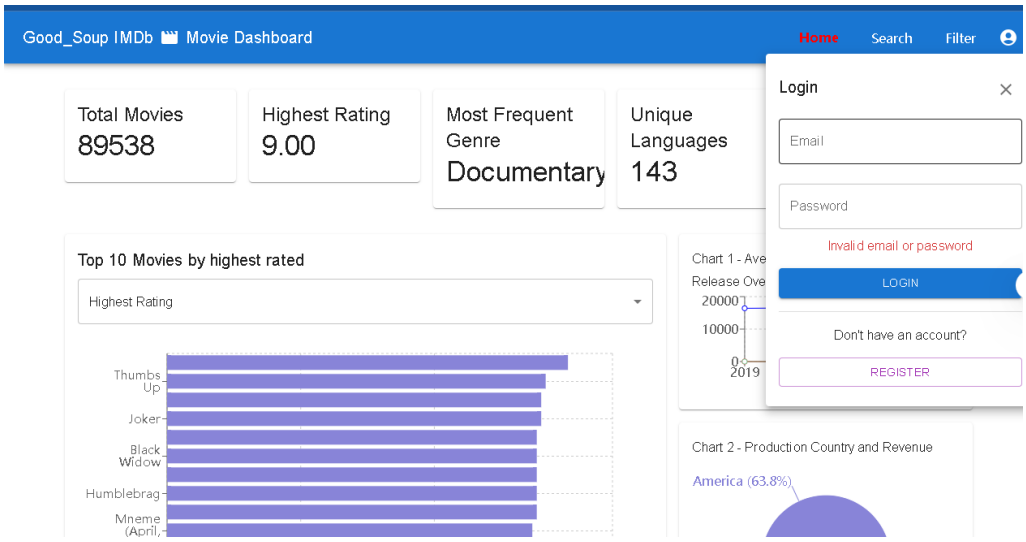

Diagram 2.5.2: This is a screenshot showing that a error message when invalid login.

3.0 Data Handling and Integration

3.1 Data Storage: Is data stored efficiently in a database (e.g., MySQL, MongoDB), with proper data management?

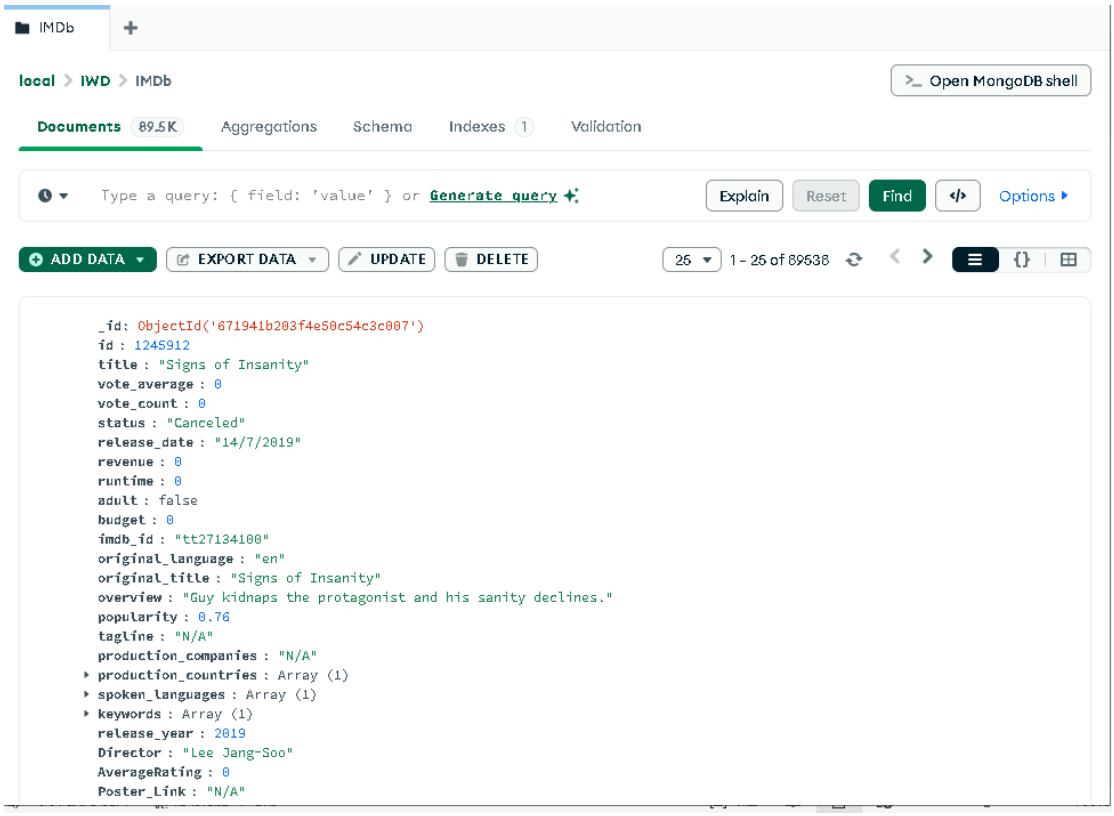

Diagram 3.1.1: This is a screenshot showing that the data is stored in MongoDB database

3.2 Data Querying: Are queries optimized for fast retrieval and accurate results, even with large datasets?

```
    query = {}
    if genre:
        query["genres_list"] = {"$regex": genre, "$options": "i"} # Case-insensitive regex search
    if title:
        query["title"] = {"$regex": title, "$options": "i"}  # Case-insensitive regex search
    if year:
        query["release_year"] = year
    if director:
        query["Director"] = {"$regex": director, "$options": "i"}  # Case-insensitive regex search

    skip = (page - 1) * limit  # Pagination logic
    cursor = movies_collection.find(query).sort(sort_by, -1).skip(skip).limit(limit)

    movies = []
    async for document in cursor:
        movies.append(Movie(**document))

    return movies
```

Diagram 3.2.1: This is a screenshot showing a query search based on different data field, with a number of limitations of results using the ". limit(limit)"

3.3 Backend Integration: Is server-side integration (e.g., PHP, Node.js) effectively implemented to manage data flow?

```
# backend/main.py

from fastapi import FastAPI, HTTPException, Query, Depends
from motor.motor_asyncio import AsyncIOMotorClient
from pydantic import BaseModel, Field
from typing import List, Optional
from fastapi.middleware.cors import CORSMiddleware
from passlib.context import CryptContext  # For hashing passwords
from datetime import datetime, timedelta
from jose import JWTError, jwt
from fastapi.security import OAuth2PasswordBearer

app = FastAPI()

# CORS configuration to allow frontend to access backend
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:3000"],  # React app domain
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# MongoDB connection
client = AsyncIOMotorClient("mongodb://localhost:27017")
db = client["IWD"]
movies_collection = db["IMDb"]
user = db["user"]

# Secret key and algorithm for JWT
```

Diagram 3.3.1: This screenshot shows the backend script using python FastAPI framework.

3.4 Data Preprocessing: Has the dataset been properly cleaned, organized, and imported into the system?

```
[5]: # Fill missing values based on recommended defaults
df['imdb_id'].fillna('N/A', inplace=True)
df['overview'].fillna('N/A', inplace=True)
df['tagline'].fillna('N/A', inplace=True)
df['production_companies'].fillna('N/A', inplace=True)
df['production_countries'].fillna('N/A', inplace=True)
df['spoken_languages'].fillna('N/A', inplace=True)
df['AverageRating'].fillna(0.0, inplace=True)
df['Poster_Link'].fillna('N/A', inplace=True)
df['Certificate'].fillna('N/A', inplace=True)
df['IMDB_Rating'].fillna(0.0, inplace=True)
df['Meta_score'].fillna(0.0, inplace=True)
df['Star1'].fillna('N/A', inplace=True)
df['Star2'].fillna('N/A', inplace=True)
df['Star3'].fillna('N/A', inplace=True)
df['Star4'].fillna('N/A', inplace=True)
df['Writer'].fillna('N/A', inplace=True)
df['Director_of_Photography'].fillna('N/A', inplace=True)
df['Producers'].fillna('N/A', inplace=True)
df['Music_Composer'].fillna('N/A', inplace=True)
```

Diagram 3.4.1: This screenshot shows the dataset is cleaned by replacing the empty value.

```
[18]: # Check for missing values in each column
missing_values = df.isnull().sum()

# Display the result
print(missing_values[missing_values > 0])  # This will show only columns with missing values
```

```
Series([], dtype: int64)
```

```
[43]: import pymongo

# Convert DataFrame to a list of dictionaries
data_dict = df.to_dict(orient='records')

# Set up MongoDB connection
client = pymongo.MongoClient("mongodb://localhost:27017/")

# Specify the database and collection
db = client['IWD']  # Database name
collection = db['IMDb']  # Collection name

# Insert data into MongoDB
collection.insert_many(data_dict)

print("Data inserted successfully!")
```

```
Data inserted successfully!
```

Diagram 3.4.2: This screenshot shows the dataset is checked no missing value, before creating an ETL pipeline to import the data into MongoDB database.

3.5 Error Handling: Are there mechanisms to handle data errors or missing data, ensuring the system remains functional?

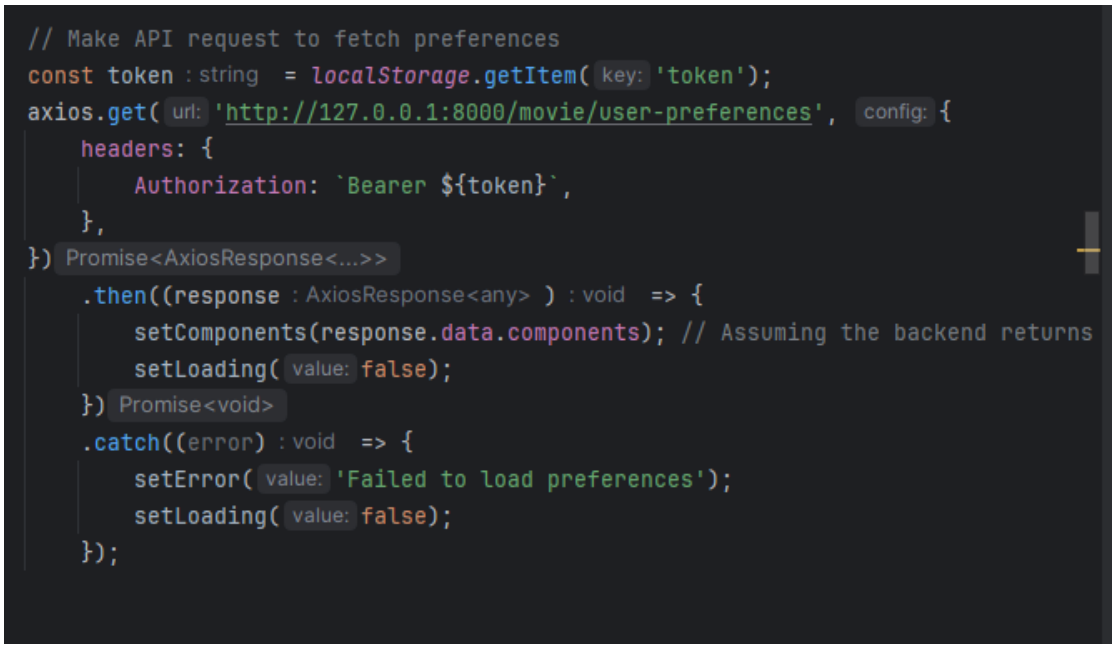

Diagram 3.5.1: This screenshot shows there is a error handling mechanism, then set the setLoading state to false to continue load other components.

4.0 Storytelling with data

4.1 Data Narrative: Does the dashboard provide a clear narrative or story that helps users make sense of the data?

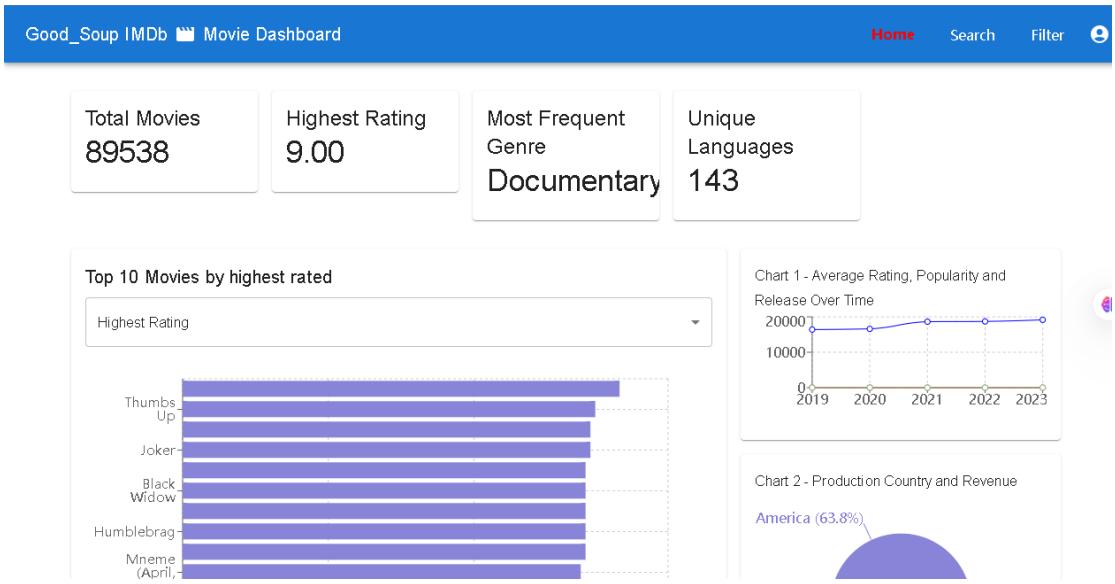

Diagram 4.1.1: This screenshot shows the dashboard provided a clear narrative such as the main KPI Card showing the summary of key insight, and the multiple chart provide user filter by genre help user make sense of the data.

4.2 Key Insights: Are the most important insights highlighted, guiding the user toward meaningful conclusions?

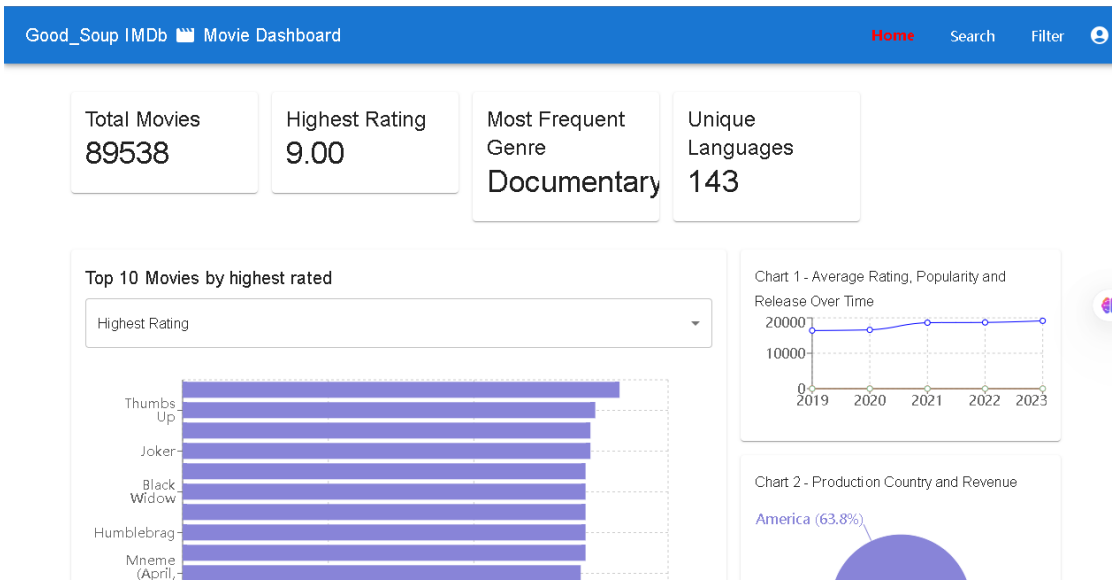

Diagram 4.2.1: This screenshot shows the dashboard providing a KPI Card which is most

important insights.

4.3 Data Visualization: Are charts, graphs, and tables used effectively to enhance the story being told?

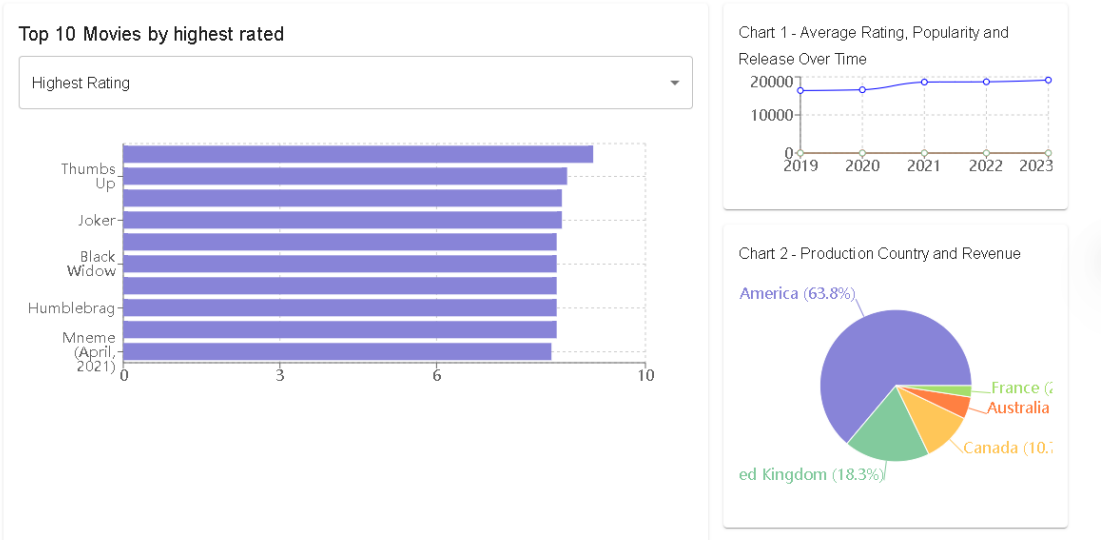

Diagram 4.3.1: This screenshot shows the multiple charts provide user a storytelling by the main chart which is the bigger one “Top 10 Movies” it can filter by genre, and other smaller charts beside will display different result in real-time base on the criteria selected.

4.4 Structure: Is the data presented in a logical sequence, with an introduction, key message, and conclusion?

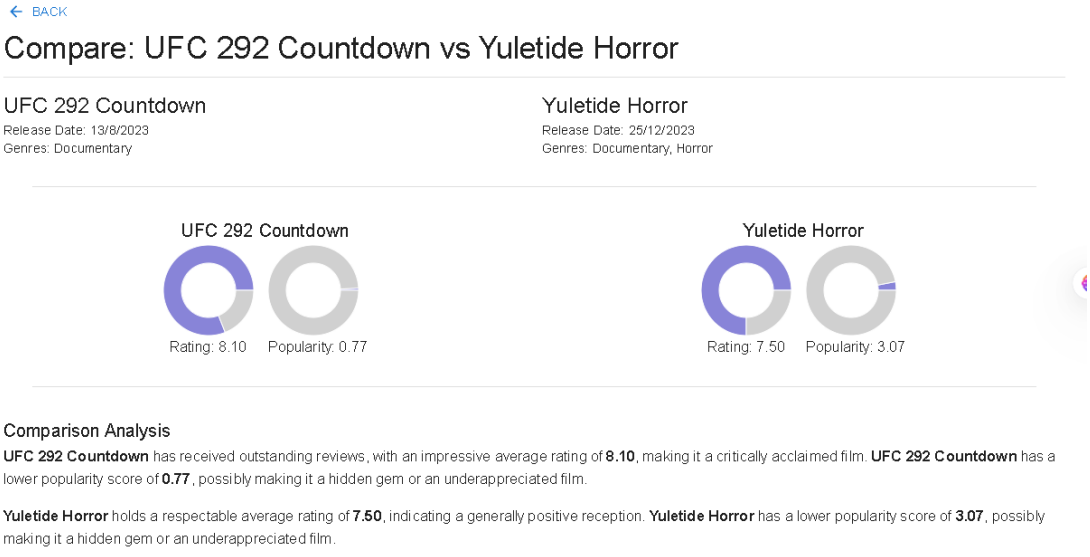

Diagram 4.1.1: This screenshot shows the comparison page provided a data presentation with an introduction key such as release date and genres, key message such as popularity and average rating and comparison analysis which as a conclusion of the 2 compared movies.

4.5 Engagement: Does the dashboard engage users, making data exploration easy and insightful without overwhelming them?

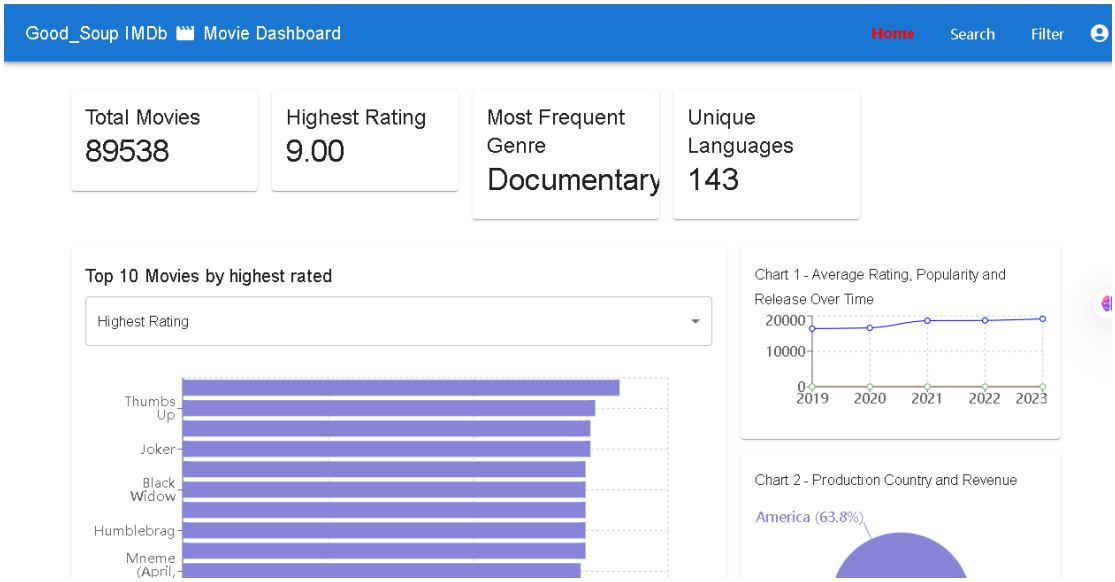

Diagram 4.5.1: This screenshot shows the dashboard provided a clean, and clear dashboard which engage user to explore the data and insightful without overwhelming user.

5.0 Basic Functionality

5.1 Sorting: Is data sorting functional for basic columns like movie ratings, release dates, or titles?

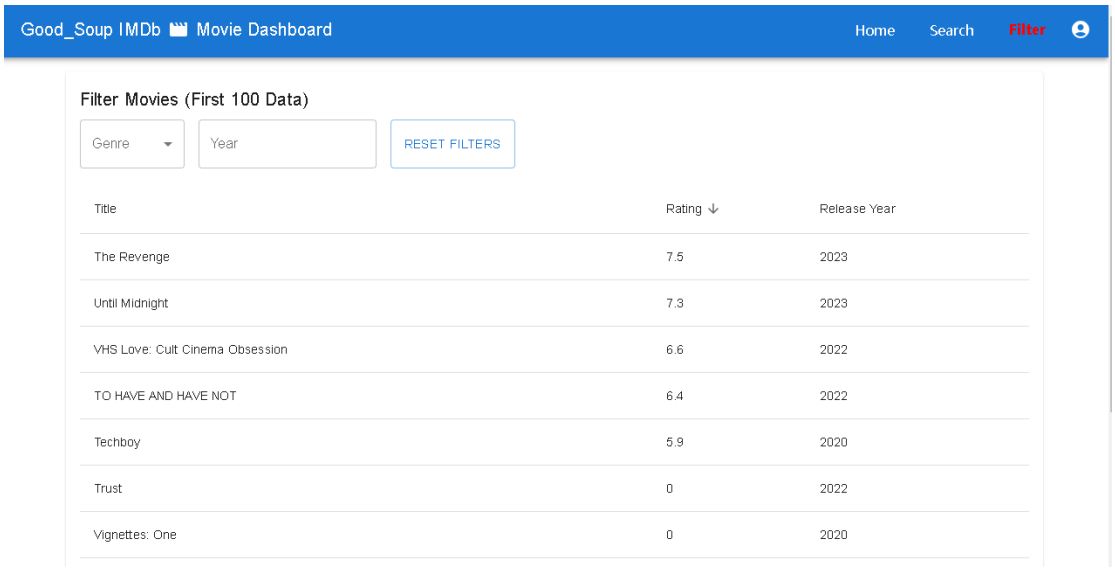

Diagram 5.1.1: This is a filter section provided to users to explore the dataset with the title is clickable to sort based on alphabetical descending or ascending order, rating is clickable to sort based on rating descending or ascending order, release year is clickable to sort based on year descending or ascending order.

5.2 Filtering by Year/Genre: Can users filter the dataset by simple criteria, such as year or genre?

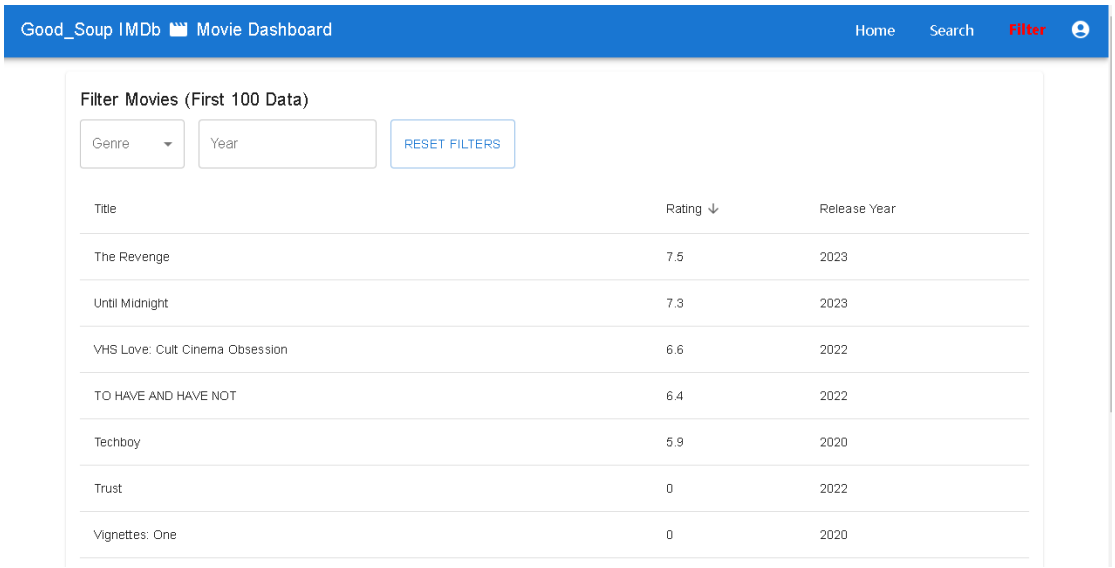

Diagram 5.2.1: This is a filter section provided to users to explore the dataset using type of genre and the year.

5.3 Basic Data Visualization: Are basic charts (e.g., bar charts, pie charts) included to visualize key metrics?

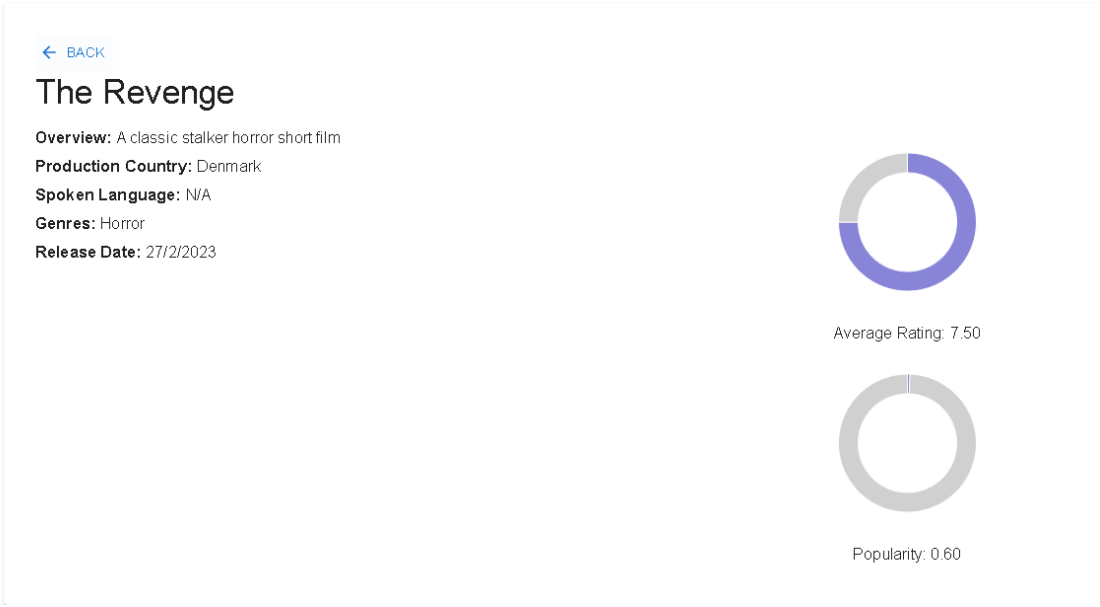

Diagram 5.3.1: This screenshot shows a basic data visualization on Average Rating and Popularity in Movie Detail Page.

5.4 Navigation: Does the dashboard have a functional navigation menu that helps users explore different sections?

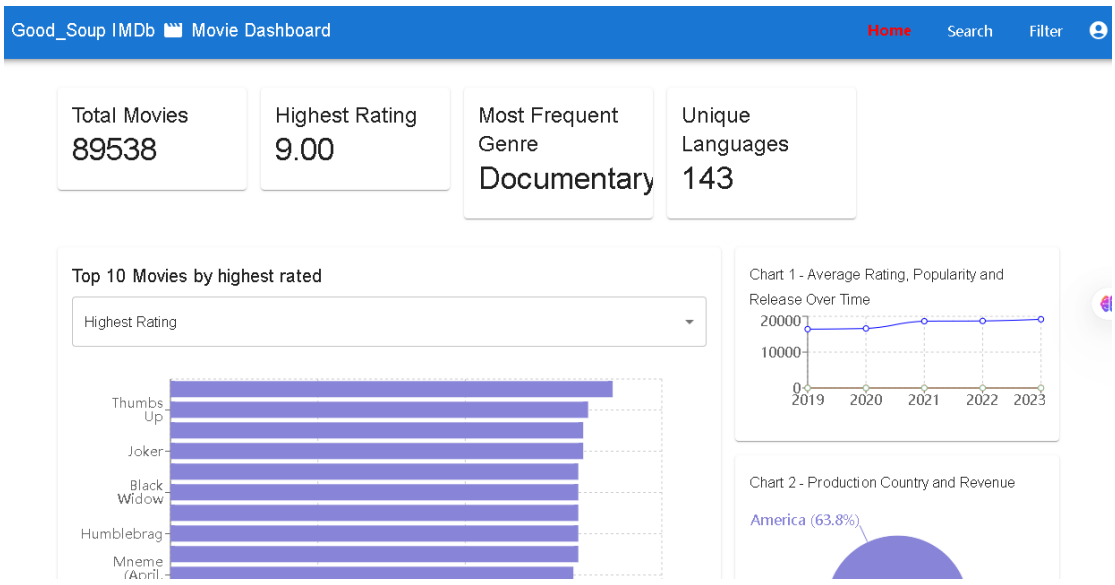

Diagram 5.4.1: This screenshot shows a navigation bar provided, with a red color highlighting when user on the page.

5.5 Static Data Display: Can users see a static display of important data points, such as top-rated movies or total movies released?

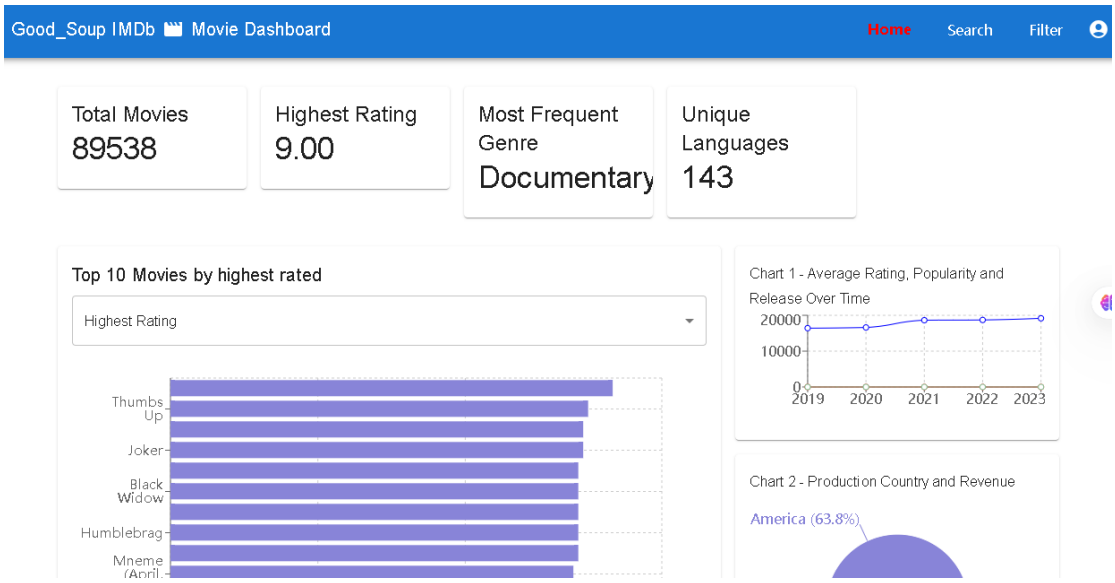

Diagram 5.5.1: This screenshot shows the KPI Cards displaying the key insight or summary of the entire dataset.

6.0 Intermedia Functionality

6.1 Multi-Criteria Filtering: Can users filter data by multiple criteria simultaneously (e.g., filtering by genre and year)?

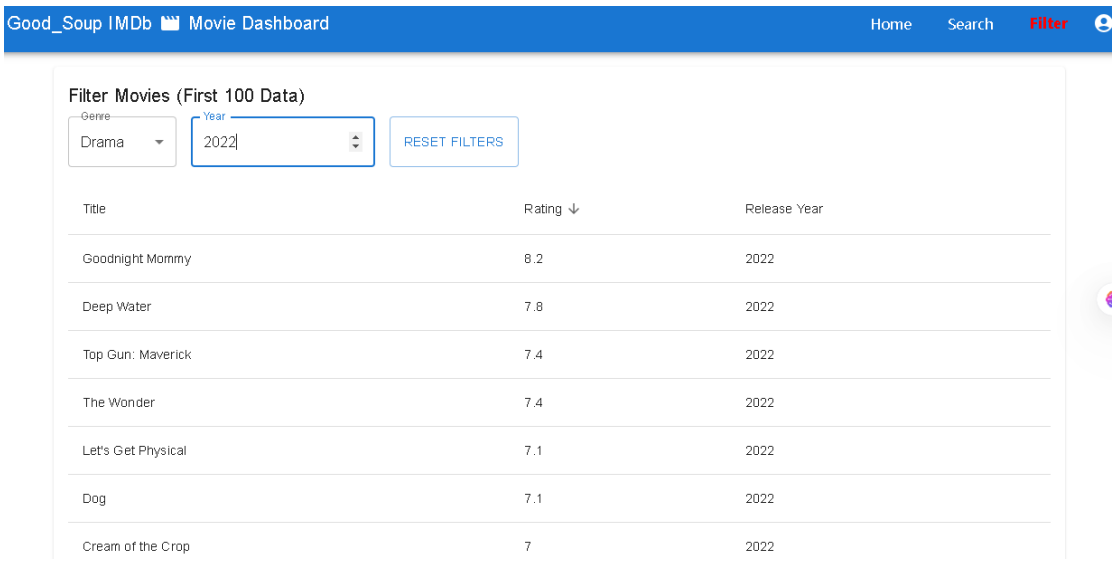

Diagram 6.1.1: This screenshot shows the multi-criteria filtering function in filter section to explore the dataset.

6.2 Advanced Data Visualization: Are more complex charts or graphs (e.g., line charts, histograms) included?

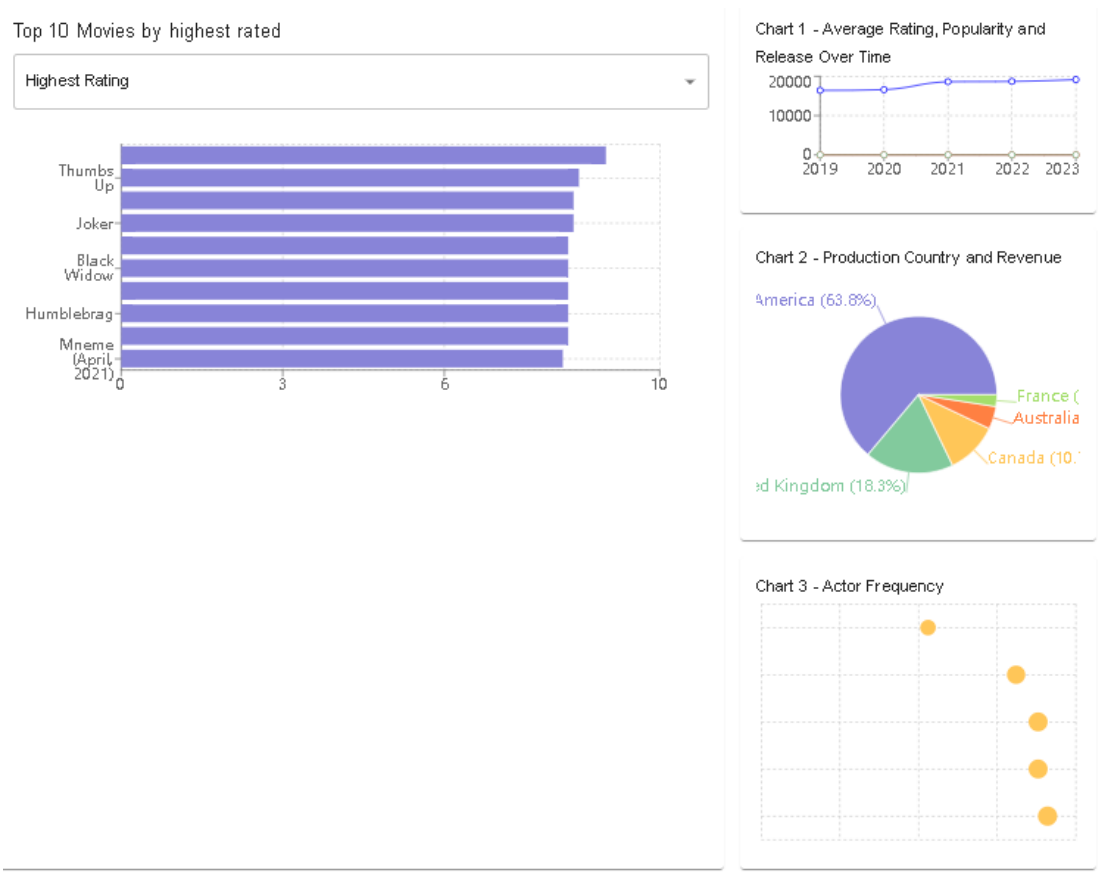

Diagram 6.2.1: This screenshot shows the different charts displayed to give information to users.

6.3 Search: Is there a functional search bar that allows users to search for specific movies, actors, or directors?

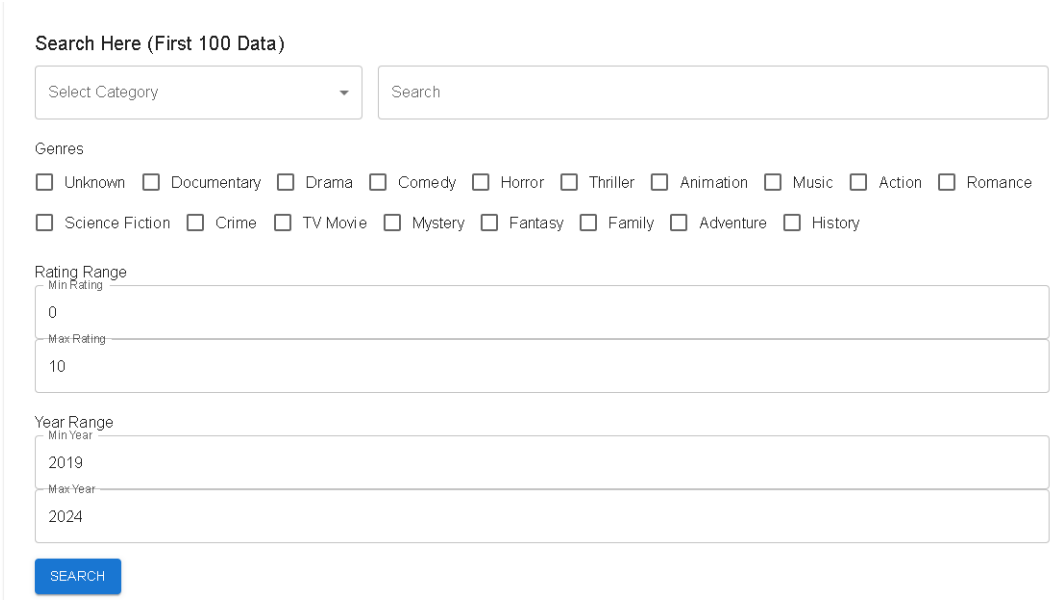

Diagram 6.3.1: This is a search function that provides a detailed search, using selected category, keyword that matched, type of genres, rating range and year range.

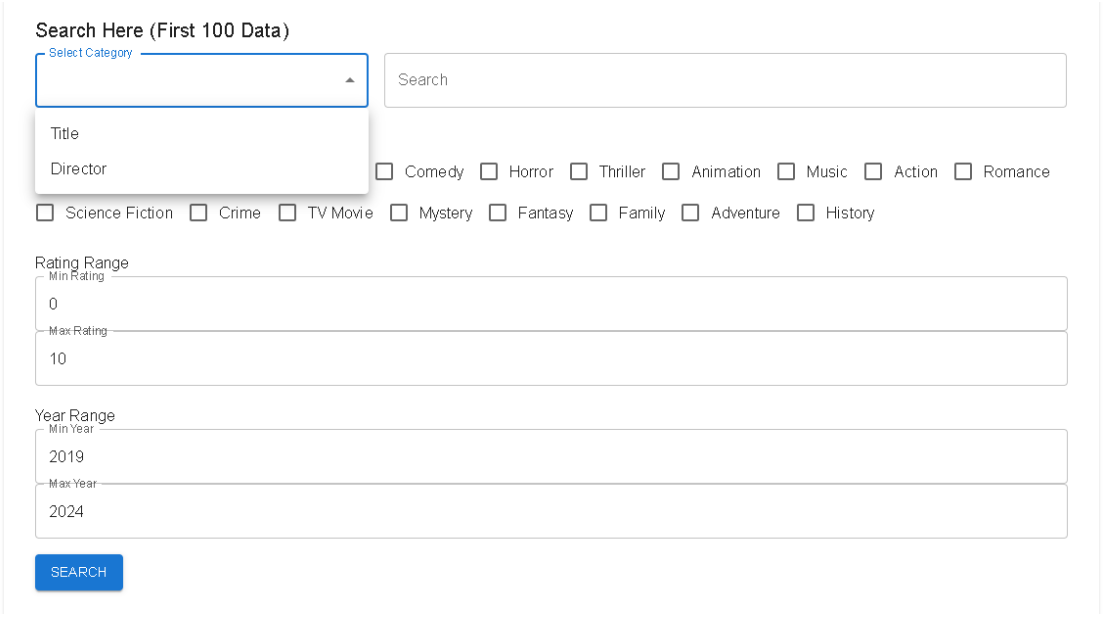

Diagram 6.3.2: This is a screenshot showing that can search based on selected category like Title or Director.

6.4 User Preferences: Are user preferences (e.g., saved filters, favorite lists) implemented to personalize the experience?

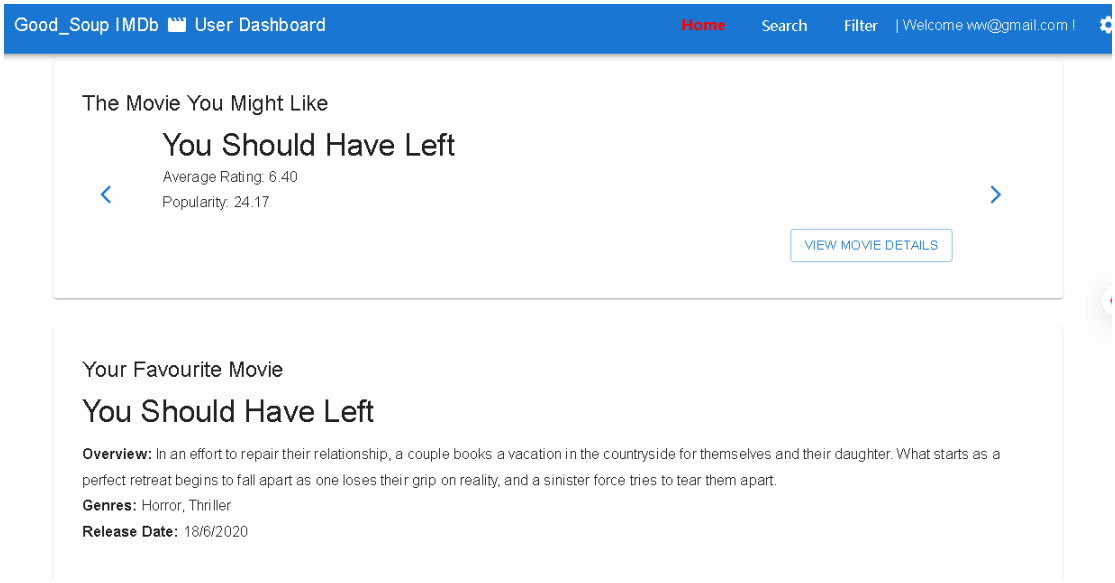

Diagram 6.4.1: This screenshot shows when login as user, a user dashboard is provided some user preferences function for example Favourite Movie component and Movie Suggestion component.

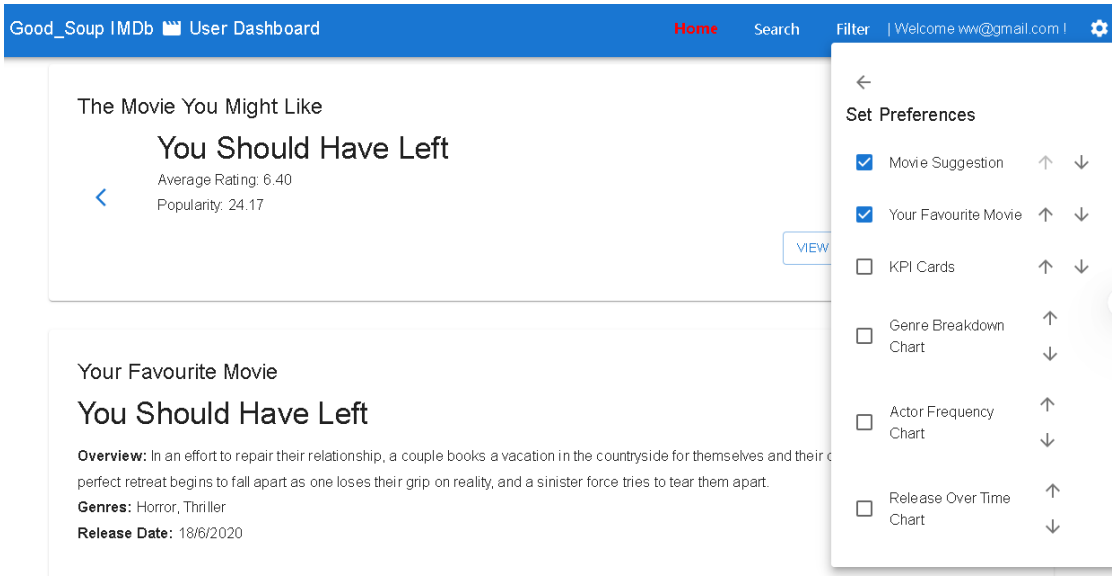

Diagram 6.4.2: This screenshot shows there is a set preference function to customize the displaying of dashboard according to visible and arrangement.

6.5 Interactive Visualizations: Do visualizations respond to user interactions, such as hovering or clicking to reveal more information?

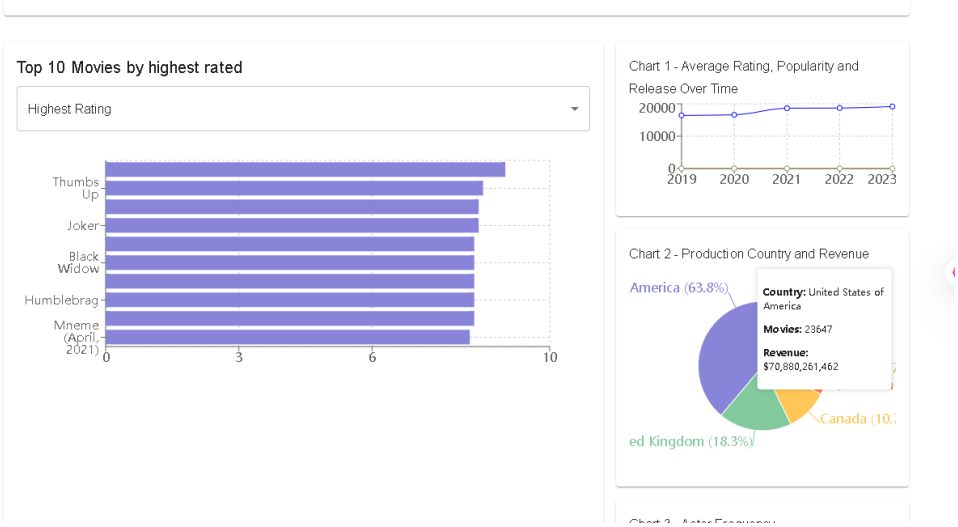

Diagram 6.5.1: This screenshot shows a hover effect to display more information on the pie chart.

7.0 Advance Functionality

7.1 Data Updates: Is there an API or mechanism to update the dataset in real time or periodically?

```
1+ usages
const fetchTopMovies = async (filter) : Promise<any>  => {
    const response : Response  = await fetch( input: `http://127.0.0.1:8000/movies/top-
    if (!response.ok) {
        throw new Error('Network response was not ok');
    }
    return response.json();
};

1+ usages
const fetchPopRating = async (filter) : Promise<any>  => {
    const response : Response  = await fetch( input: `http://127.0.0.1:8000/movies/pop-
    if (!response.ok) {
        throw new Error('Network response was not ok');
    }
    return response.json();
};

1+ usages
const fetchProduction = async (filter) : Promise<any>  => {
    const response : Response  = await fetch( input: `http://127.0.0.1:8000/movies/prod
    if (!response.ok) {
        throw new Error('Network response was not ok');
```

Diagram 7.1.1: This screenshot shows using API to call backend for a real-time fetching the data from database.

7.2 User Authentication: Can users create accounts and save their dashboard preferences or data views?

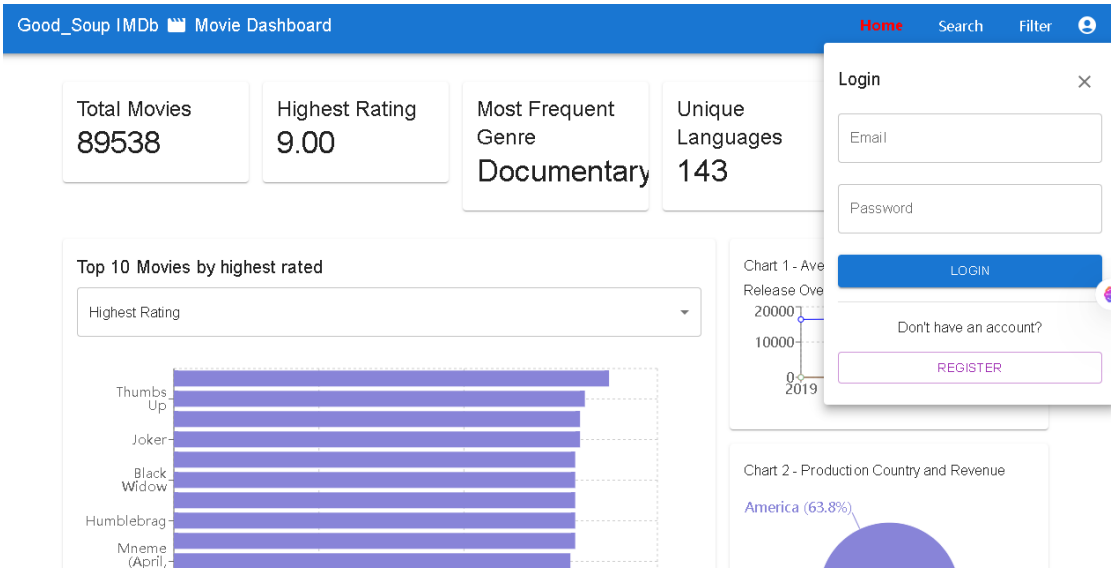

Diagram 7.2.1: This screenshot shows there is a login form and register form for user to login or create their account.

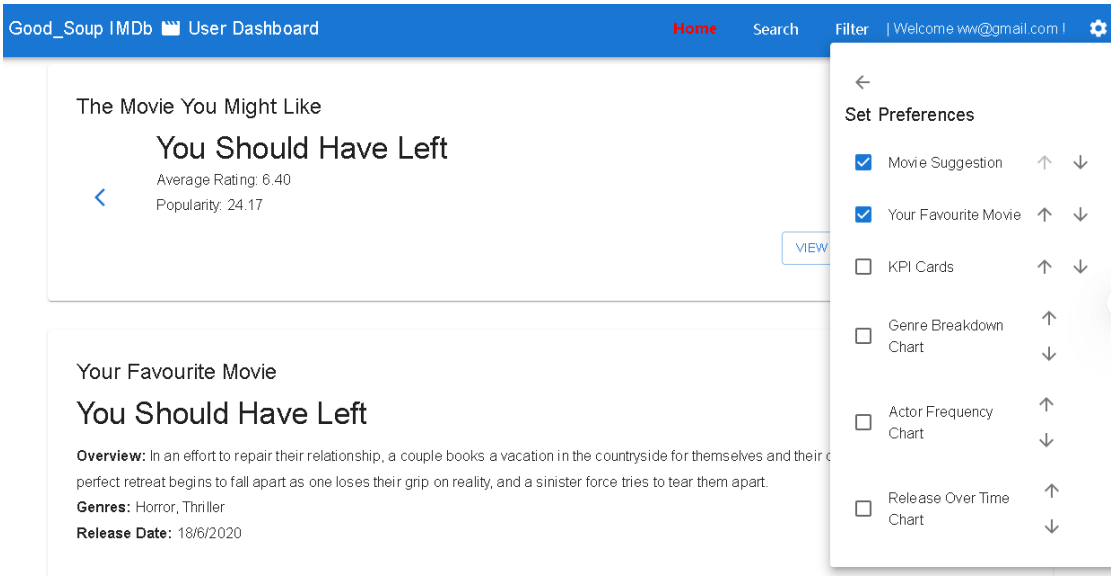

Diagram 7.2.2: This screenshot shows there is a set preference function to customize and save the displaying of dashboard according to visible and arrangement for the user account

7.3 Comparison Tools: Are there tools that allow users to compare movies, genres, or income side by side?

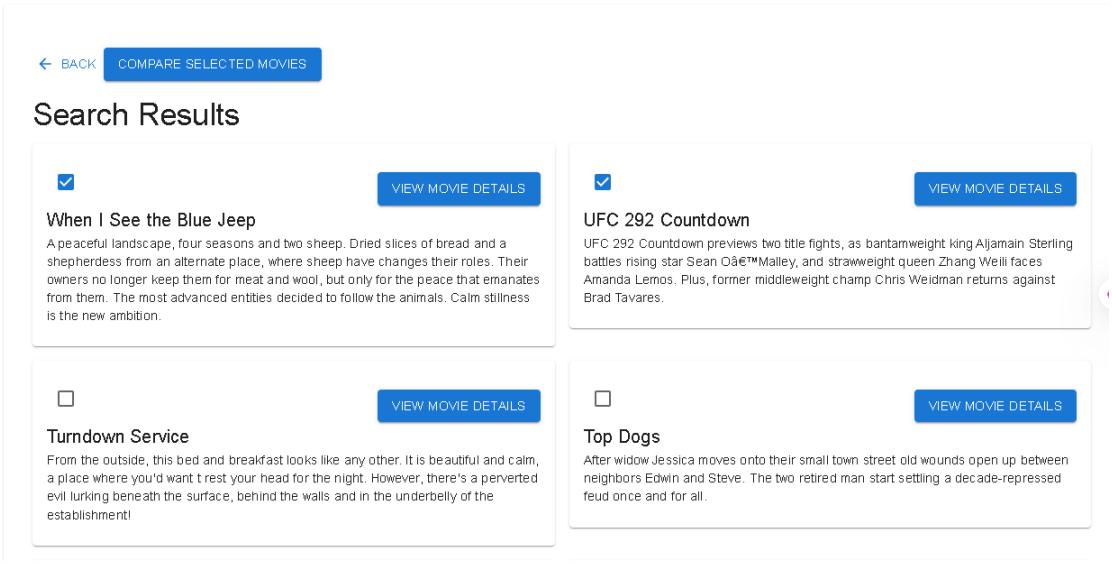

Diagram 7.3.1: This screenshot shows that the search result allow user to compare the movie when selected 2 movies.

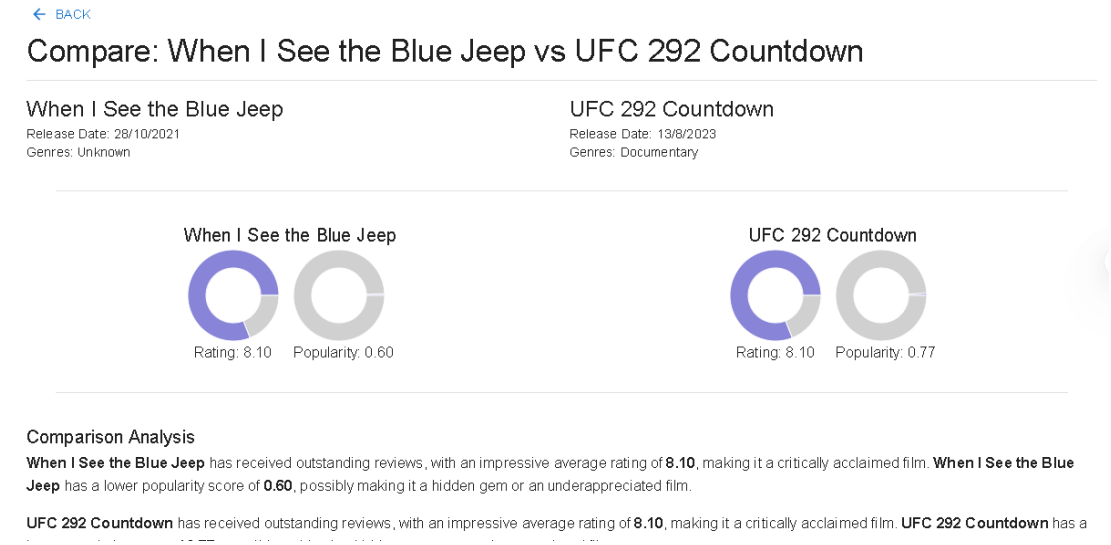

Diagram 7.3.2: This screenshot shows a movie comparison side by side, with a summarize analysis.

7.4 Performance Optimization: Have performance optimizations been implemented to improve loading times and data processing speed?

Diagram 7.4.1: This screenshot shows the use of caching the search results to improve the loading times, when navigator back to search result page from movie detail page or compare movie page.

7.5 Real-Time Visualization: Are there advanced, real-time visualizations that display live or constantly updating data?

Diagram 7.5.1: This screenshot shows using API to call backend for a real-time fetching the data from database.