Project Report

on

BingeWatch

Submitted in partial fulfillment of the requirement for the award of degree of

Master of Computer Application (MCA)

at

University of Mumbai

Submitted by

Ajay Suresh Bhaskaran

Under the guidance of

Dr.Suhasini Vijaykumar

Bharati Vidyapeeth's

Institute of Management and Information Technology

Sector 8, CBD Belapur

Navi Mumbai

2023 - 2024

Bharati Vidyapeeth's

Institute of Management and Information Technology

Navi Mumbai

Certificate of Approval

This is to certify that the Project titled **'BingeWatch'** is successfully done by **Jitin Gopakumar** during internship of his course in partial fulfillment of **Masters of Computer Application** under the **University of Mumbai, Mumbai**, through the Bharati Vidypeeths Institute of Management and Information Technology, Navi Mumbai carried out by him under our guidance and supervision.

Sign & Date
Dr.Suhasini Vijaykumar

External Examiner
Signature and Date

Principal
Dr.Suhasini Vijaykumar

College Seal

Declaration

(Signature)

Ajay Suresh Bhaskaran

Date

Acknowledgement

I am delighted to extend my heartfelt appreciation to all those who played a pivotal role in the development of BingeWatch, my project on an innovative Movie Recommendation system and sentiment analysis website. This project aimed to provide users with a comprehensive and user-friendly platform to track, manage, and analyze their expenses. I am profoundly grateful for the guidance and support offered by my project guide, Dr.Suhasini Vijaykumar, whose expertise and constructive feedback were instrumental in shaping the project.

I would also like to express my gratitude to my classmates and friends who actively contributed by providing valuable feedback, insightful suggestions, and unwavering encouragement throughout the development process. Their support and enthusiasm played a crucial role in keeping me motivated and focused on delivering a high-quality product.

Additionally, I extend my appreciation to the end users of BingeWatch, whose feedback and usage patterns proved invaluable in identifying areas for improvement and refining the application's features. Their engagement and input significantly contributed to the success of the project.

...

Date:

Jitin Gopakumar

Abstract

BingeWatch is an advanced movie recommendation system aimed at revolutionizing how users discover and enjoy films tailored to their preferences. The project tackles the challenges inherent in traditional movie selection methods by offering a sophisticated web application that provides real-time suggestions, personalized movie lists, and interactive user support.

The primary goal of BingeWatch is to empower users by providing a seamless and centralized platform for discovering movies that match their tastes. The development involves leveraging HTML, CSS, JavaScript, Python, and machine learning algorithms to create a robust web-based application.

The approach includes an extensive analysis of existing movie recommendation systems, highlighting gaps in services like CinemaRater and FlixPulse. BingeWatch addresses these gaps by introducing superior recommendation capabilities and a more intuitive interface.

Key findings from the project include the successful implementation of a reliable movie recommendation engine that effectively addresses user pain points. The conclusion underscores the significant value BingeWatch brings to enhancing movie discovery experiences and outlines future enhancements, such as enhanced recommendation algorithms, automated genre detection, and deeper integration with streaming platforms.

Contents

List of Tables

List of Figures

Nomenclature

ROI	Return on Investment.
recommenderalg	Recommender algorithm
mlmodel	Machine Learning Model
HTML	Hypertext Markup Language
Python	A high-level, interpreted programming language
GHz	GigaHertz
CinemaRater	An existing movie recommendation website

Chapter 1

Introduction

1.1 Introduction to Project

Welcome to BingeWatch, an innovative movie recommendation system designed to transform your streaming experience. BingeWatch goes beyond mere movie suggestions; it's your personalized guide to discovering the perfect content tailored to your tastes. With a sophisticated blend of machine learning algorithms and comprehensive data analysis, BingeWatch ensures that every recommendation aligns seamlessly with your preferences, making your entertainment choices effortless and enjoyable.

BingeWatch is your virtual movie companion, offering a dynamic platform where users can explore a vast library of films with ease. Featuring intuitive navigation and personalized recommendations, BingeWatch empowers users to browse, select, and enjoy movies that resonate with their interests. The system leverages advanced content-based filtering techniques enhanced by multi-faceted sentiment analysis, providing insights into user preferences through expressive emojissmiles for recommended hits and frowns for misses.

In the realm of entertainment recommendation systems, BingeWatch stands out by combining cutting-edge technology with a user-centric approach. Unlike conventional streaming platforms or generic movie databases, BingeWatch harnesses the power of Python, natural language processing (NLP), and collaborative filtering algorithms to deliver a seamless and interactive interface. This technological fusion ensures that users not only discover new favorites effortlessly but also enjoy a personalized and engaging movie discovery journey.

This introduction lays the foundation for the subsequent chapters, where we delve into the technical intricacies, functionalities, and implementation strategies of BingeWatch. Chapter 2 will provide an in-depth exploration of the system architecture, followed by installation instructions and user guides in Chapters 3 and 4. Administrative considerations and the technological backbone of BingeWatch will be detailed in Chapters 5 and 6, respectively. Chapters 7 and 8 will address troubleshooting, future enhancements, and strategic insights, ensuring that BingeWatch remains at the forefront of personalized entertainment discovery.

1.2 Domain Knowledge

BingeWatch operates within the domain of personalized movie recommendation systems, focusing on enhancing the streaming experience through tailored content discovery and user engagement. This domain encompasses processes related to recommending, exploring, and enjoying movies based on individual preferences and behaviors. Unlike conventional streaming services, BingeWatch aims to simplify movie selection by leveraging advanced technologies and data-driven insights.

BingeWatch introduces a sophisticated recommendation engine that analyzes user preferences and behavior to suggest movies that align with their tastes. The platform features a user-friendly interface where users can seamlessly browse, select, and enjoy movies recommended specifically for them. By integrating machine learning algorithms and comprehensive data analysis, BingeWatch provides personalized movie suggestions that cater to diverse viewing preferences.

Users utilize the platform to explore movie recommendations, view detailed information about movies, and interact with sentiment analysis features that provide insights through emotive responses like smiles for recommended hits and frowns for less suitable options. BingeWatch enhances the movie-watching experience with features such as personalized dashboards, sentiment-based recommendations, and interactive user interfaces.

Technologies employed for BingeWatch's implementation include Python for backend processing, machine learning libraries such as SKlearn for recommendation algorithms, Flask for web framework development, and Streamlit for creating intuitive data science applications. The development process emphasizes user-centric design principles, real-time data processing capabilities, and deep integration of machine learning techniques in the domain of entertainment recommendation systems.

1.3 Problem Description

BingeWatch addresses the growing demand for a sophisticated and personalized movie recommendation system in the era of streaming services. Current platforms often overwhelm users with vast content libraries, making it challenging to discover movies aligned with their preferences and interests. BingeWatch seeks to redefine the streaming experience by offering a centralized platform that leverages advanced algorithms for seamless content discovery and user engagement.

The landscape of movie streaming lacks a unified solution for users seeking personalized content recommendations. Existing platforms often rely on generic suggestions or rudimentary filtering methods, which may not accurately reflect individual viewing preferences. BingeWatch aims to fill this gap by providing a dynamic platform where users can explore, select, and enjoy movies tailored to their unique tastes.

Traditional methods of movie discovery are often disjointed, leaving users to navigate through extensive catalogs without personalized guidance. BingeWatch simplifies this process by employing state-of-the-art machine learning techniques and comprehensive data analysis to deliver accurate and relevant movie recommendations. By analyzing user behavior and preferences, BingeWatch ensures that each recommendation enhances the user's viewing experience.

Users interact with BingeWatch through a user-friendly interface where they can browse recommended movies, view detailed information, and engage with sentiment analysis features that convey emotive responses like smiles for recommended hits and frowns for less suitable options. The platform enhances user satisfaction by providing intuitive navigation, real-time updates, and interactive features that optimize the movie-watching journey.

Chapter 2

System Study

Before BingeWatch, existing movie recommendation systems offered basic browsing and generic recommendations, often failing to personalize the movie discovery experience effectively. Users relied on manual browsing or simplistic algorithms that didn't align closely with individual preferences.BingeWatch revolutionizes personalized entertainment discovery by integrating advanced machine learning and data analysis. It analyzes user behavior and movie preferences to deliver highly tailored recommendations. The platform's user-friendly interface allows easy browsing, selection, and enjoyment of movies personalized to each user's tastes.

2.1 Existing system

The current landscape of movie recommendation systems offers diverse methods for users to discover movies, often relying on basic browsing and generic recommendations. Users typically resort to manual browsing or external reviews to find movies aligned with their preferences.BingeWatch distinguishes itself by employing advanced machine learning and data analytics to analyze user behavior, movie preferences, and sentiment from reviews, delivering highly personalized recommendations.However, Cinematrix faces challenges in scalability and advanced personalization with its technological stack using Java and MySQL, potentially limiting real-time feedback integration.

2.2 Limitations

Limitations of Cinematrix

Lack of Real-time Recommendation: Cinematrix may lack real-time alerts, hindering immediate notifications about newly recommended movies or changes in user preferences.

Technological Constraints: Using Java and MySQL, Cinematrix may face scalability challenges and limitations in handling large datasets efficiently.

User Interface Usability: The user interface of Cinematrix may be basic, lacking interactivity and customization options that could enhance user experience and navigation.

Limited User Support: Cinematrix may offer limited interactive user support features, impacting user engagement and satisfaction during movie selection.

2.3 Proposed system

BingeWatch Solutions Real-time Recommendation: BingeWatch overcomes this limitation by introducing timely movie suggestions based on user preferences.Users can timely content aligned with their preferences.

Centralized Recommendation: BingeWatch simplify content discovery, BingeWatch features a centralized dashboard. Users gain a comprehensive view of recommended movies, genres, and trending content in one cohesive interface.

Sentiment Analysis of Reviews: BingeWatch distinguishes itself by introducing personalized content suggestions aligned with user preferences and emotional responses.

2.4 Objectives

Objectives of BingeWatch

- To enhance user engagement with real-time alerts and personalized recommendations.
- To implement sentiment analysis to tailor recommendations based on user emotions.
- To implement sentiment analysis to tailor recommendations based on user emotions.
- To enhance the quality of movie recommendations by refining content similarity and user preference analysis.
- To scale the recommendation system to handle increasing user data and content volumes efficiently.
- To enhance user experience by providing intuitive interfaces and responsive features.

2.5 Feasibility Studies

2.5.1 Economical

The economic feasibility of BingeWatch is assessed by considering both tangible and intangible benefits derived from the project. A cost-benefit analysis is employed to evaluate the economic viability. While the project incurs initial investment costs, it is justifiable as it promises to enhance the quality of streaming service management. The economic feasibility is determined by estimating overall costs, including development, marketing, and maintenance expenses. The evaluation considers the potential return on investment and aims to establish whether the expected revenue justifies the investment. BingeWatch proves to be economically feasible, as it offers improved user engagement and retention over existing methods while maintaining cost-effectiveness.

2.5.2 Technical

The technical feasibility of BingeWatch involves an analysis of current resources, including hardware and the technology stack required for development. This study ensures the availability of the necessary resources and technologies for the project's successful implementation. BingeWatch is developed using a technology stack that includes HTML, CSS, and JavaScript for the frontend. There is no backend as the system relies on datasets for its operation. The use of these technologies ensures that the application can accommodate future modifications and updates, enhancing its technical viability.

2.5.3 Operational

Operational feasibility is evaluated by examining how BingeWatch will function within its intended environment. The compatibility of the application with different devices, operating systems, and screen resolutions is carefully considered. The goal is to ensure that the application meets user expectations in terms of performance and usability. BingeWatch is designed to be versatile, catering to various user preferences and technological environments. The operational feasibility assessment confirms that the application aligns with user expectations and seamlessly operates across different platforms, making it a robust and user-friendly solution.

Chapter 3

System Analysis

3.1 Gantt chart

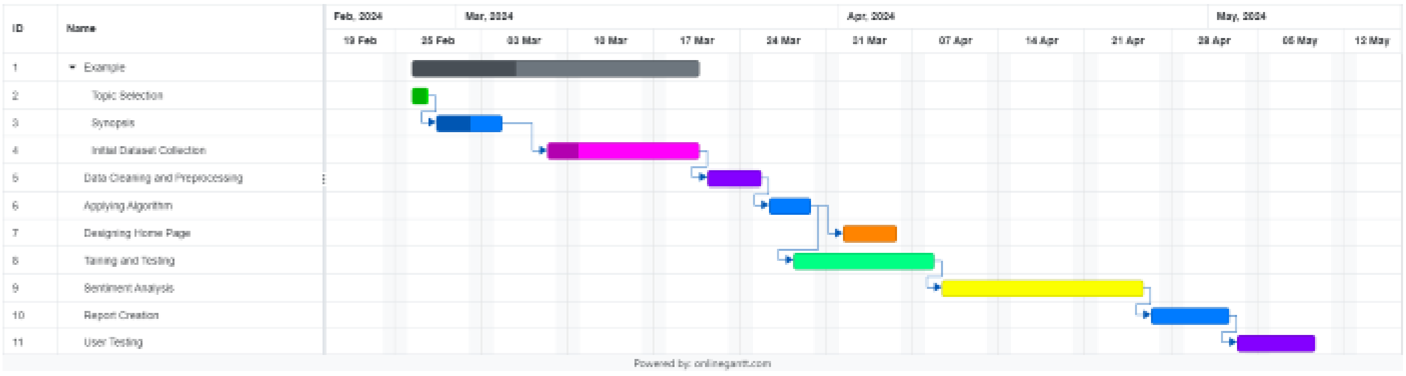


Figure 3.1: Gantt Chart For BingeWatch

The Gantt chart outlines the project timeline for developing the BingeWatch Movie Recommendation System, covering the period from February to May 2024.

3.2 Use Case Diagram

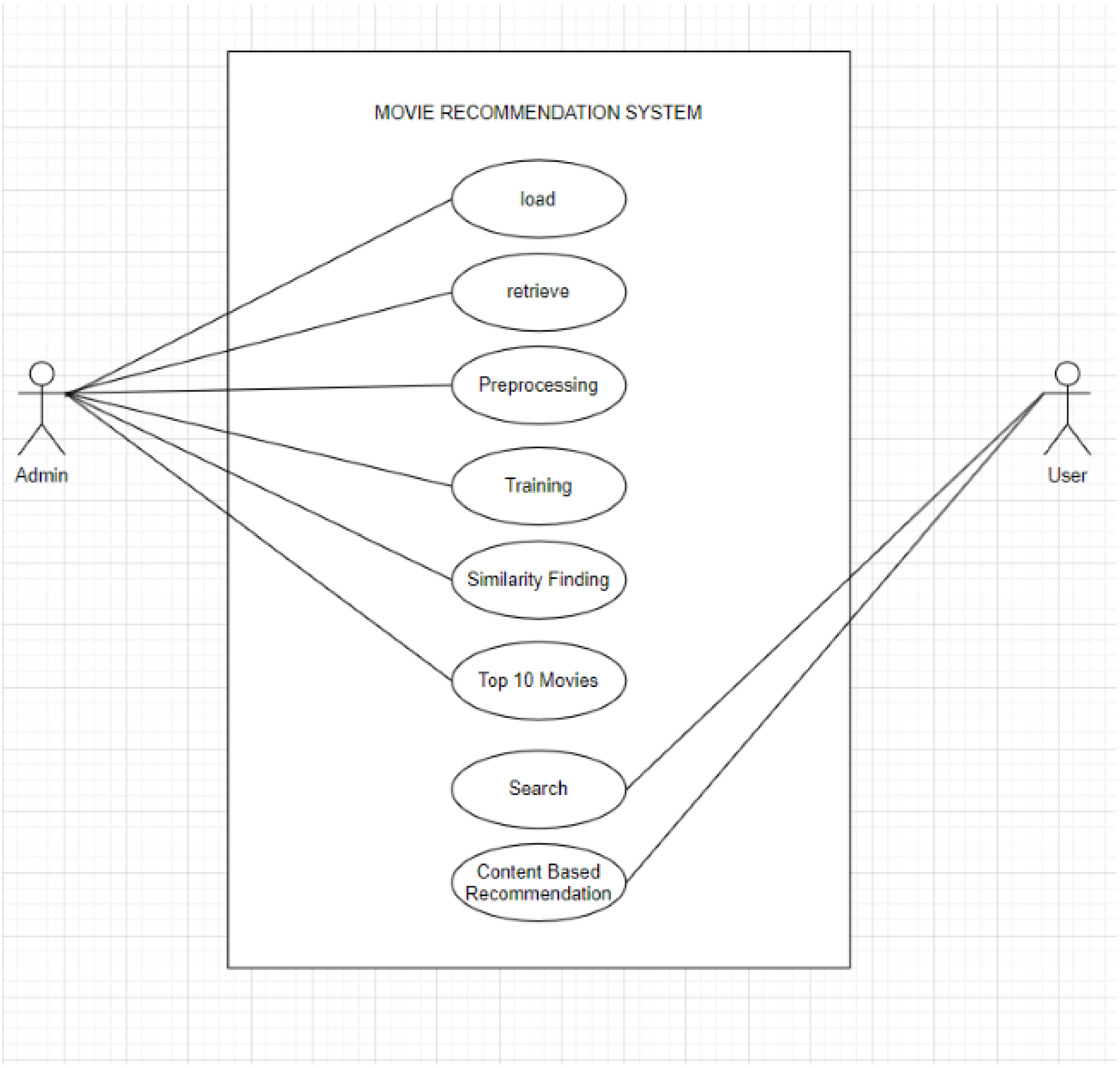


Figure 3.2: Use Case Diagram For BingeWatch

This use case diagram depicts the interactions between an Admin, a Dataset, and a User within the Movie Recommendation system.

3.3 Operating tools and technology

Front-End Development: HTML, CSS, JavaScript
HTML (HyperText Markup Language): HTML is the standard markup language used to create the structure of web pages. It provides the basic building blocks for web content, defining elements such as headings, paragraphs, and links.CSS (Cascading Style Sheets): CSS is used for styling and formatting HTML elements, enhancing the visual presentation of web pages. It controls aspects like layout, color, and fonts. JavaScript: JavaScript is a scripting language that enables interactive and dynamic features on web pages. It is essential for creating responsive and user-friendly interfaces.
Software Requiremnets:
Text Editor (VS Code/Jupyter Notebook): Essential for writing and editing code. Anaconda Distribution Package: Simplifies the management of Python packages and dependencies. Python Libraries: Required for data processing, analysis, and visualization.
Hardware Requirements:
PC with Windows/Linux OS: Ensures compatibility with the required software and provides a stable environment for development. Processor with 2.40GHz or 2.50 GHz speed: Sufficient processing power to handle data processing and analysis tasks efficiently. Minimum of 8GB RAM: Adequate memory to support the execution of data analysis and machine learning algorithms.
New learning while developing the system (software and hardware) Developing BingeWatch provides an opportunity for continuous learning in both software and hardware domains. On the software side, the implementation of HTML, CSS, and JavaScript for the front-end introduces insights into creating visually appealing and interactive user interfaces. HTML (HyperText Markup Language) is fundamental for structuring web pages, defining elements such as headings, paragraphs, images, and links. CSS (Cascading Style Sheets) is essential for styling these elements, enhancing the visual presentation of web pages by controlling layout, color, and fonts. JavaScript enables interactive and dynamic features, creating responsive and user-friendly interfaces that allow for dynamic content updates without reloading the page.
In addition to front-end development, working with Python and data analysis libraries is crucial for the back-end and data processing aspects of BingeWatch. The Anaconda distribution simplifies package management and deployment, providing an environment that includes essential data-science packages. SKlearn (Scikit-learn) is instrumental for implementing machine learning algorithms, such as classification, regression, and clustering, which are vital for generating personalized movie recommendations. NumPy offers tools for high-performance array processing, crucial for numerical computations. Pandas provides data structures and analysis tools, enabling efficient manipulation of numerical data and time series, which is essential for analyzing user preferences and behavior.
In the hardware domain, understanding the intricacies of CPU architectures, processor speeds, memory requirements, and disk space considerations is essential. Developing BingeWatch involves using a processor with a speed of 2.40 GHz or faster and a minimum of 8 GB RAM to ensure smooth performance. This knowledge contributes to a holistic understanding of how software interacts with underlying hardware components, enabling developers to optimize performance and ensure a seamless user experience. Additionally, understanding system compatibility and optimizing software performance based on hardware specifications are crucial for delivering an efficient and user-friendly application.

Chapter 4

System Design

4.1 ER Diagram

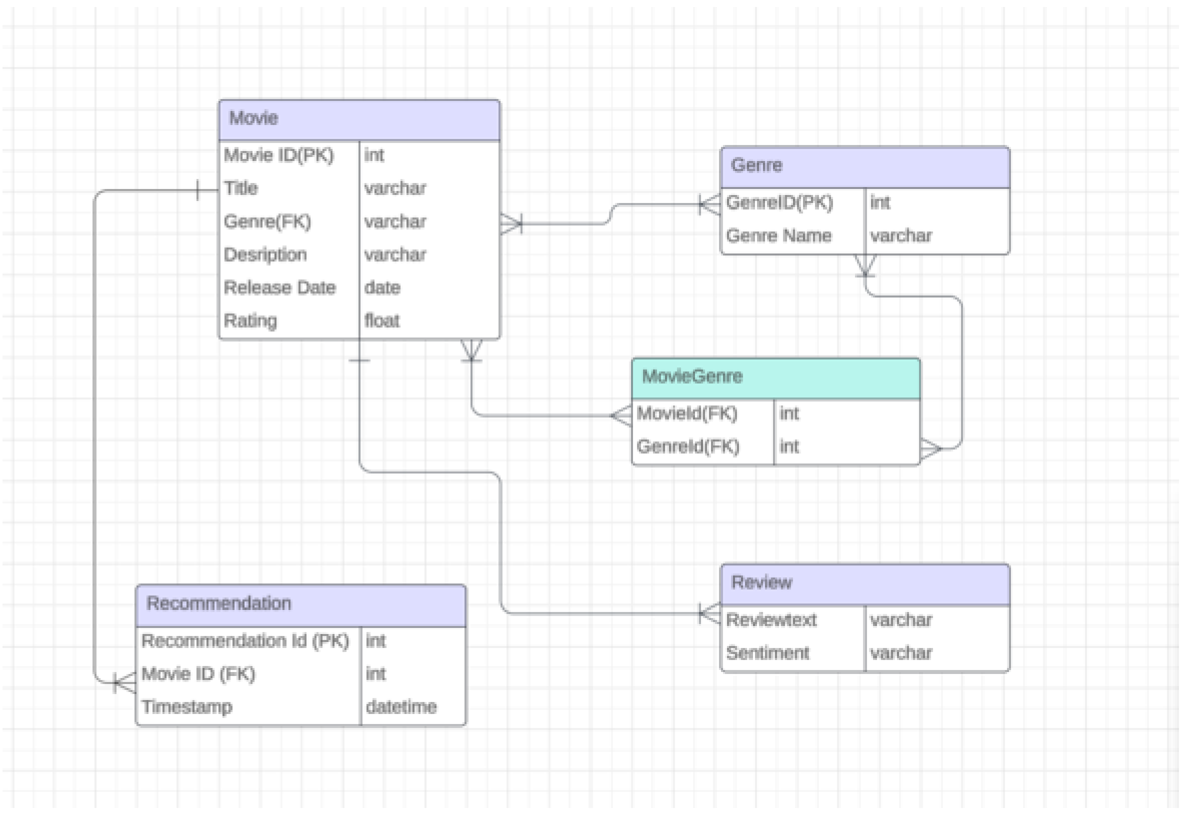


Figure 4.1: ER Diagram For BingeWatch

This diagram shows the database schema for a movie recommendation system. The key components are movie, genre, moviegenre, recommendation and review.

4.2 Activity Diagram

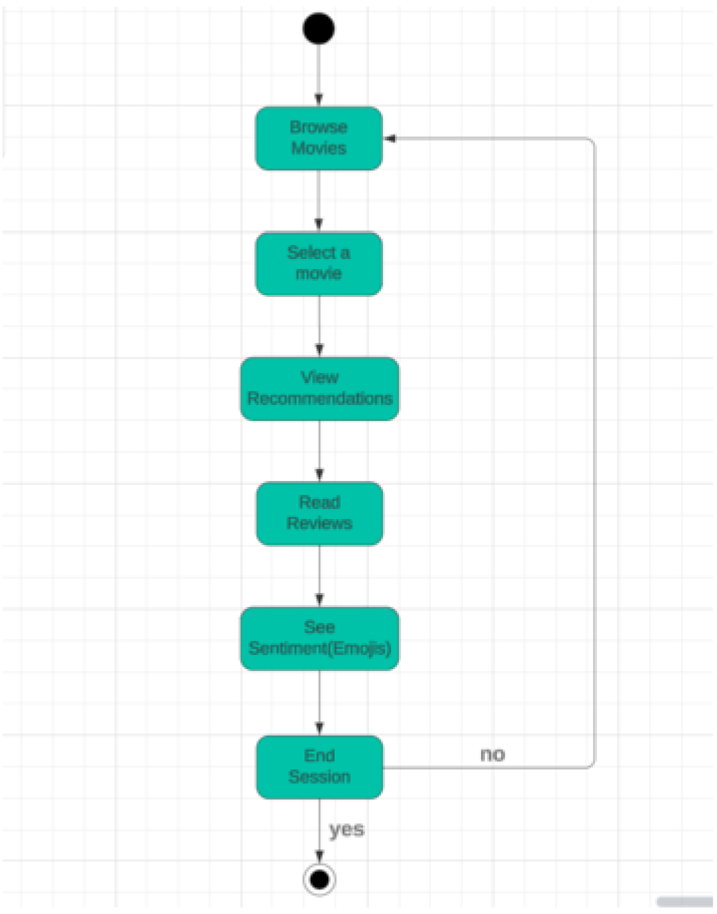


Figure 4.2: Activity Diagram For BingeWatch

The Activity diagram outlines the key user interactions with the BingeWatch application.

4.3 Input/Output Design

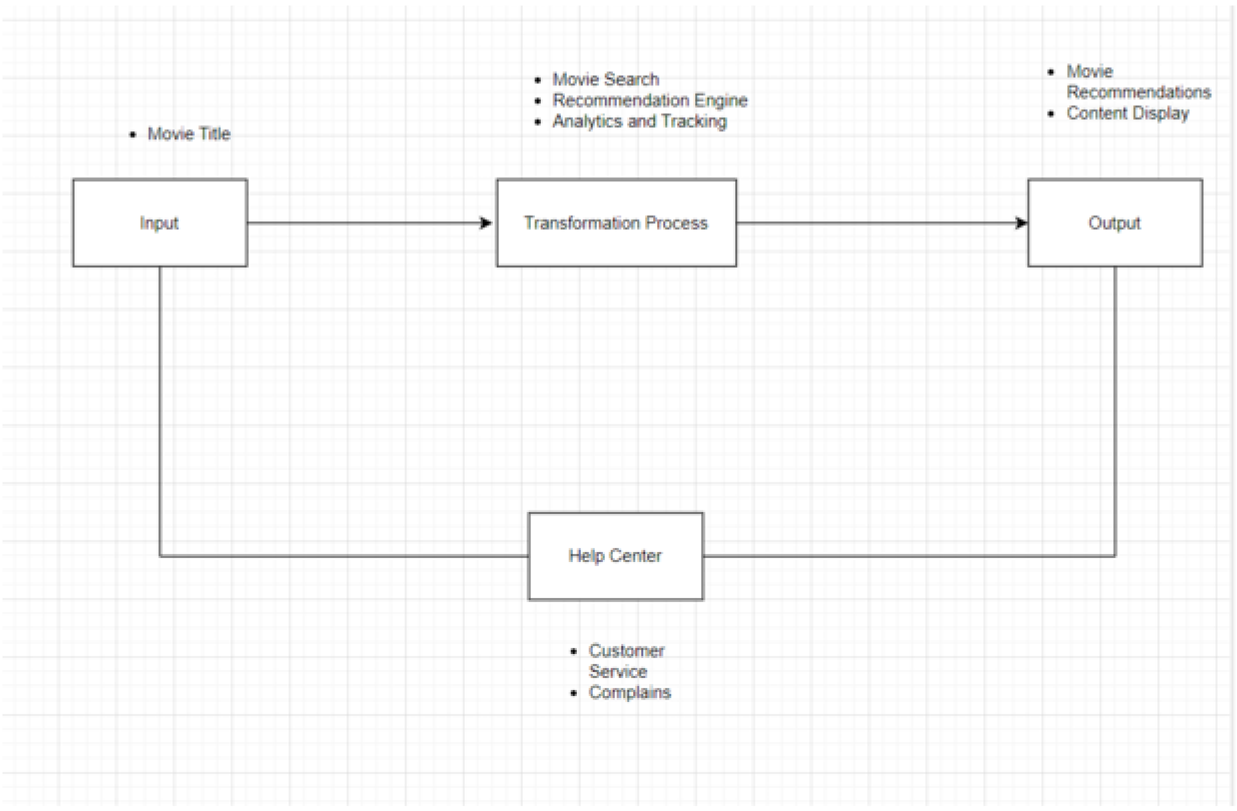
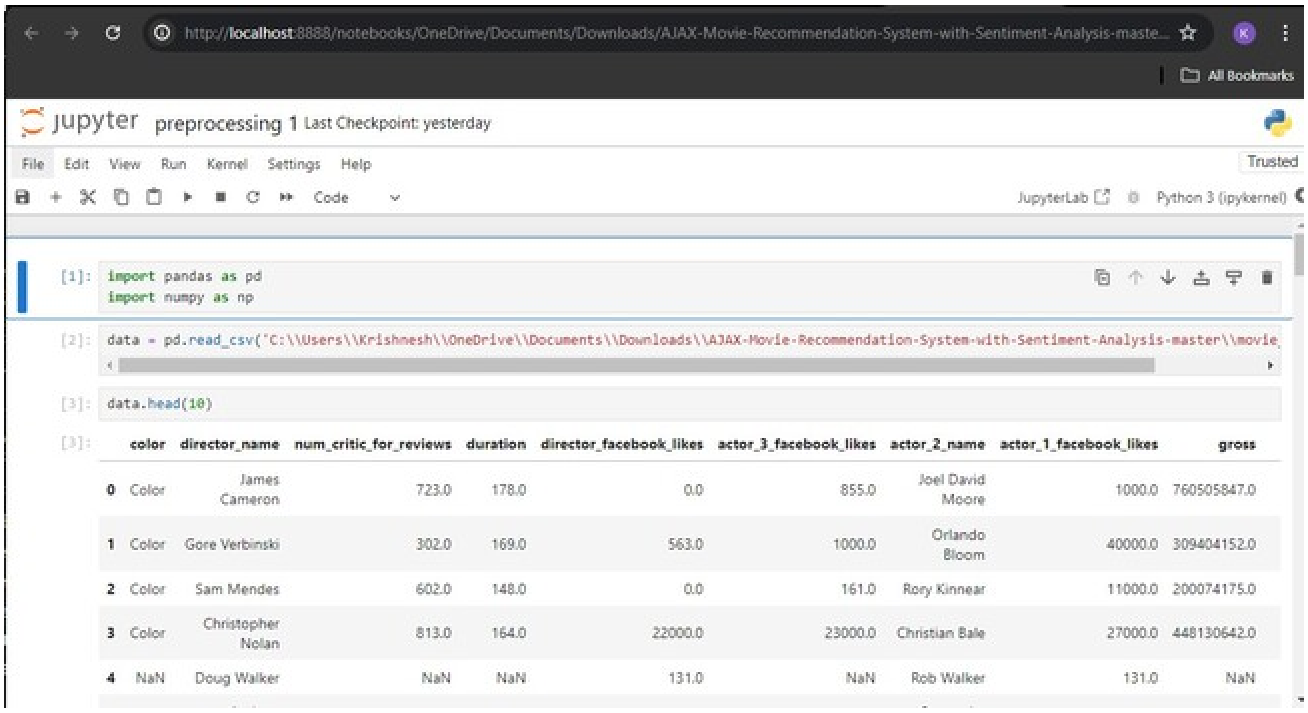


Figure 4.3: Input/Output Design For BingeWatch

This diagram shows the process flow within the movie recommendation system. The components used are input, transformation process, help and output.

Chapter 5

Coding Jupyter and Home Page


http://localhost:8888/notebooks/OneDrive/Documents/Downloads/AJAX-Movie-Recommendation-System-with-Sentiment-Analysis-maste...
All Bookmarks
jupyter preprocessing 1 Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

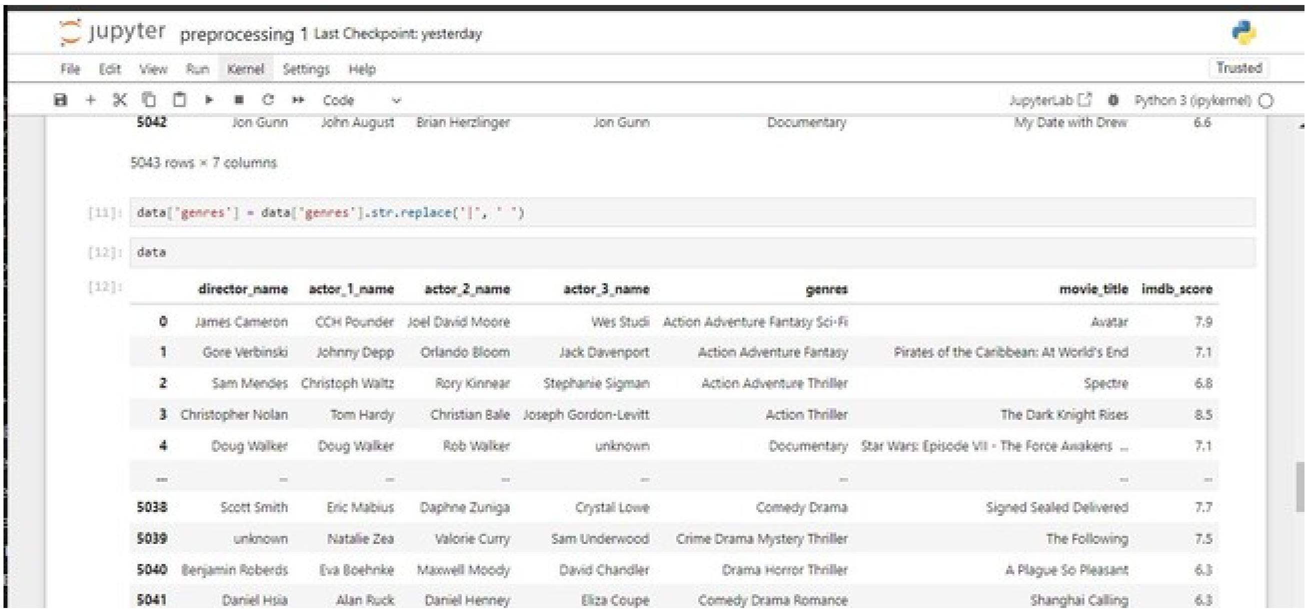
```
[1]: import pandas as pd
     import numpy as np

[2]: data = pd.read_csv('C:\\Users\\Krishnesh\\OneDrive\\Documents\\Downloads\\AJAX-Movie-Recommendation-System-with-Sentiment-Analysis-master\\movie_

[3]: data.head(10)
```

[3]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN

Figure 5.1: Reading the csv file

Loads a CSV file containing movie data and displays Displays the first ten rows of the Dataframe to provide an overview of the data, which includes columns like actorname, director name, duration, and gross.


jupyter preprocessing 1 Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
5042    Jon Gunn    John August    Brian Herzlinger    Jon Gunn    Documentary    My Date with Drew    6.6

5043 rows × 7 columns
```

```
[11]: data['genres'] = data['genres'].str.replace('|', ' ')
```

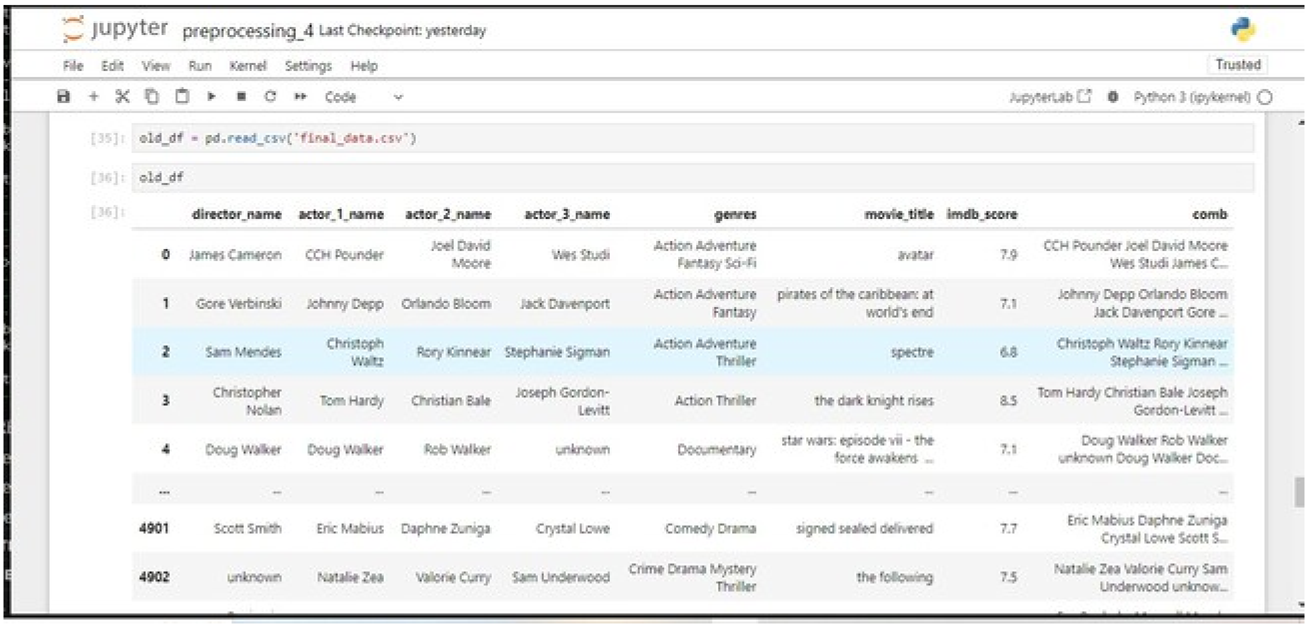
```
[12]: data
```

[12]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	imdb_score
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy Sci-Fi	Avatar	7.9
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	Pirates of the Caribbean: At World's End	7.1
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	Spectre	6.8
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action Thriller	The Dark Knight Rises	8.5
4	Doug Walker	Doug Walker	Rob Walker	unknown	Documentary	Star Wars: Episode VII - The Force Awakens ...	7.1
...	...	...	...	...	...	...	...
5038	Scott Smith	Eric Mabius	Daphne Zuniga	Crystal Lowe	Comedy Drama	Signed Sealed Delivered	7.7
5039	unknown	Natalie Zea	Valorie Curry	Sam Underwood	Crime Drama Mystery Thriller	The Following	7.5
5040	Benjamin Roberds	Eva Boehnke	Maxwell Moody	David Chandler	Drama Horror Thriller	A Plague So Pleasant	6.3
5041	Daniel Hsia	Alan Ruck	Daniel Henney	Eliza Coupe	Comedy Drama Romance	Shanghai Calling	6.3

Figure 5.2: Cleaning the data

Selects specific columns from the DataFrame (director name, actor 1 name, actor 2 name, actor 3 name, genres, movie title, and imdb score) and displays the resulting DataFrame.Cleans the genres column by removing square brackets ([]) using the str.replace function and updates the DataFrame.


jupyter preprocessing_4 Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[35]: old_df = pd.read_csv('final_data.csv')
```

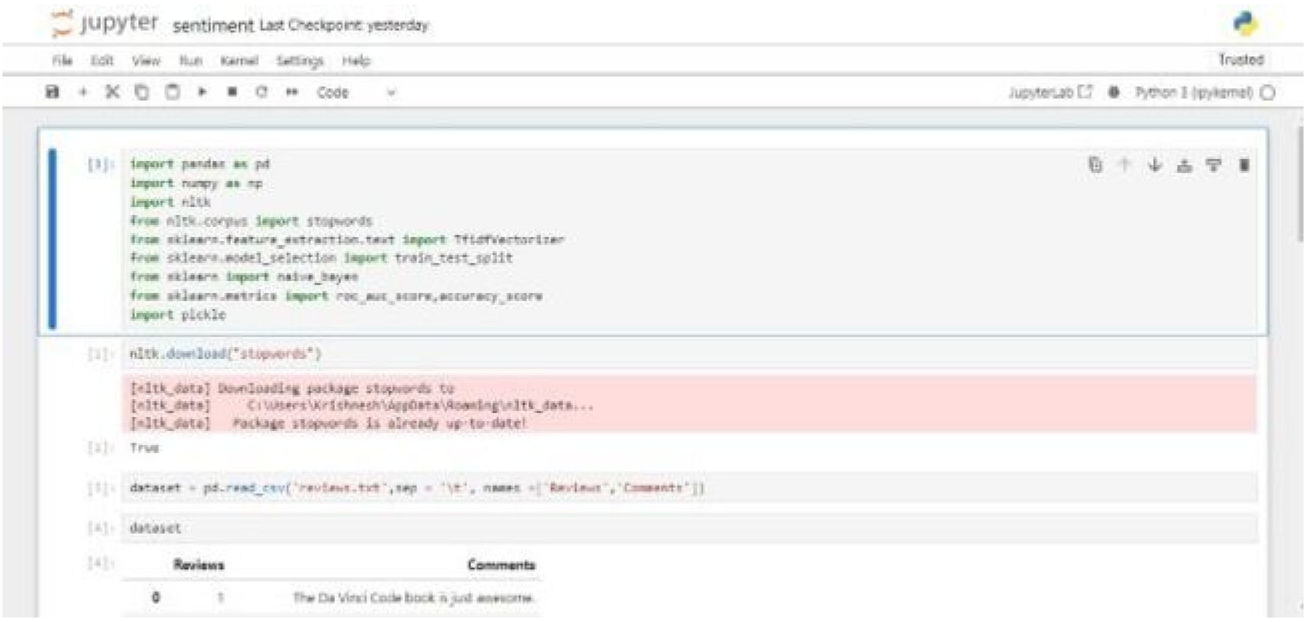
```
[36]: old_df
```

[36]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	imdb_score	comb
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy Sci-Fi	avatar	7.9	CCH Pounder Joel David Moore Wes Studi James C...
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	pirates of the caribbean: at world's end	7.1	Johnny Depp Orlando Bloom Jack Davenport Gore ...
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	spectre	6.8	Christoph Waltz Rory Kinnear Stephanie Sigman ...
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action Thriller	the dark knight rises	8.5	Tom Hardy Christian Bale Joseph Gordon-Levitt ...
4	Doug Walker	Doug Walker	Rob Walker	unknown	Documentary	star wars: episode vii - the force awakens ...	7.1	Doug Walker Rob Walker unknown Doug Walker Doc...
...	...	...	...	...	...	...	...	...
4901	Scott Smith	Eric Mabius	Daphne Zuniga	Crystal Lowe	Comedy Drama	signed sealed delivered	7.7	Eric Mabius Daphne Zuniga Crystal Lowe Scott S...
4902	unknown	Natalie Zea	Valorie Curry	Sam Underwood	Crime Drama Mystery Thriller	the following	7.5	Natalie Zea Valorie Curry Sam Underwood unknow...

Figure 5.3: Data after getting cleaned

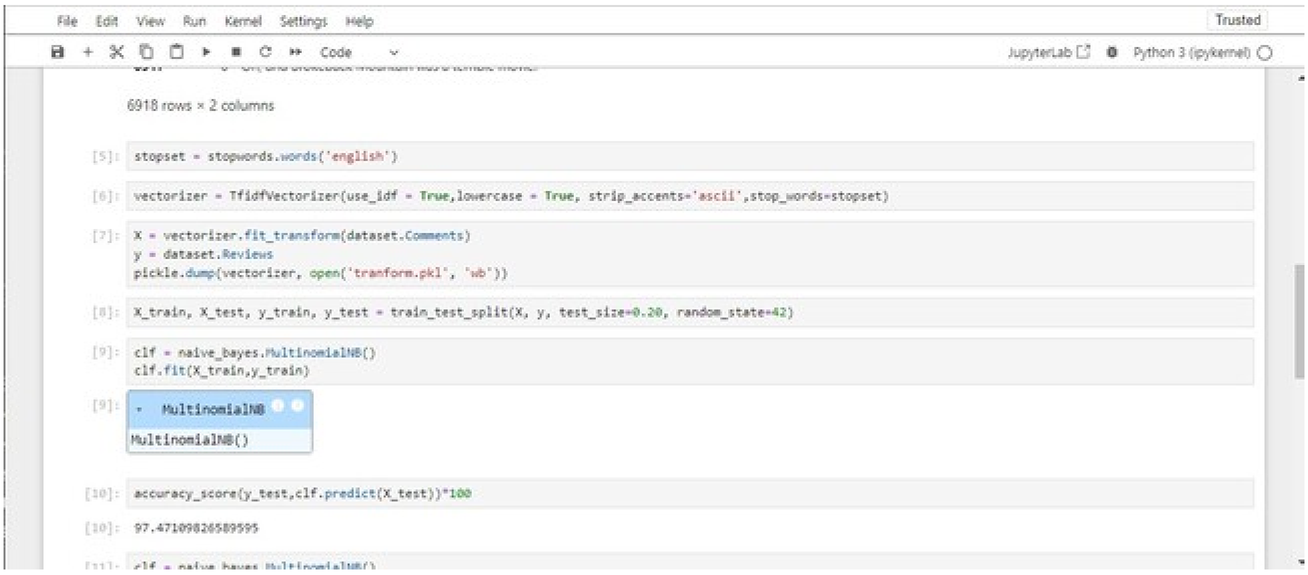
Imports necessary libraries, including pandas, numpy, requests, BeautifulSoup (bs4), and url-lib.Defines a link to a Wikipedia page containing a list of American films of 2020.Uses urllib to open the link and read the source HTML content.


jupyter sentiment Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[3]: import pandas as pd
     import numpy as np
     import nltk
     from nltk.corpus import stopwords
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.model_selection import train_test_split
     from sklearn import naive_bayes
     from sklearn.metrics import roc_auc_score,accuracy_score
     import pickle

[1]: nltk.download("stopwords")

     [nltk_data] Downloading package stopwords to
     [nltk_data]     C:\Users\Krishnesh\AppData\Roaming\nltk_data...
     [nltk_data]   Package stopwords is already up-to-date!
[1]: True

[3]: dataset = pd.read_csv('reviews.txt',sep = '\t', names =['Reviews','Comments'])

[4]: dataset

[4]:     Reviews                              Comments
       0       1    The Da Vinci Code book is just awesome.
```

Figure 5.4: Csv File for Sentiment Analysis

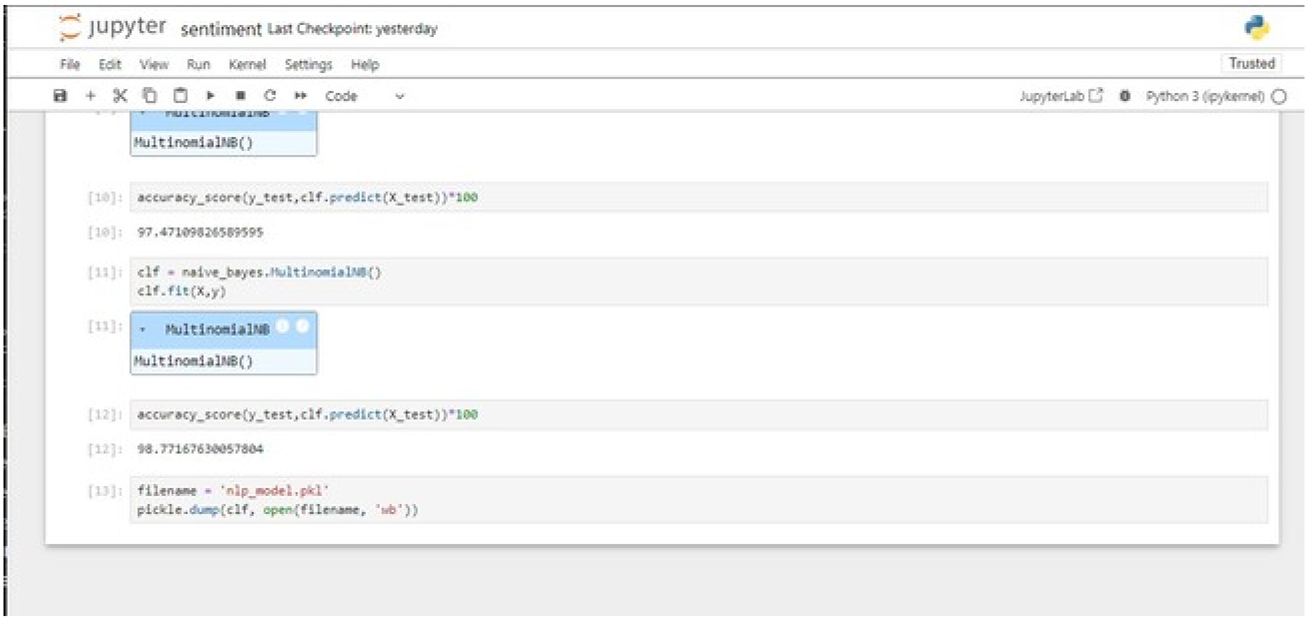
This code snippet demonstrates the process of importing libraries and reading a dataset for sentiment analysis.It includes the setup for text vectorization and classification using a Multinomial Naive Bayes model.



```
6918 rows × 2 columns

[5]: stopset = stopwords.words('english')

[6]: vectorizer = TfidfVectorizer(use_idf = True,lowercase = True, strip_accents='ascii',stop_words=stopset)

[7]: X = vectorizer.fit_transform(dataset.Comments)
     y = dataset.Reviews
     pickle.dump(vectorizer, open('tranform.pkl', 'wb'))

[8]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

[9]: clf = naive_bayes.MultinomialNB()
     clf.fit(X_train,y_train)

[9]: MultinomialNB
     MultinomialNB()

[10]: accuracy_score(y_test,clf.predict(X_test))*100

[10]: 97.47109826589595
```

Figure 5.5: Training the data

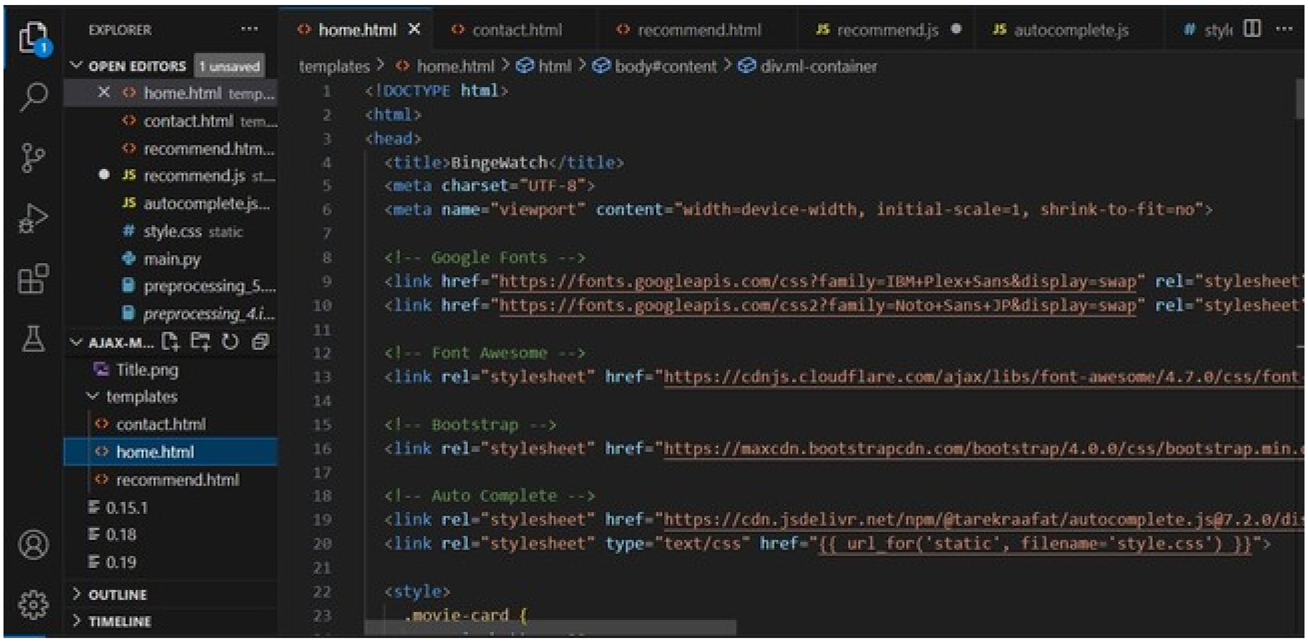
This continuation shows the fitting of the Naive Bayes model and evaluating its accuracy.The trained model is then saved using pickle for later use.



```
MultinomialNB()

[10]: accuracy_score(y_test,clf.predict(X_test))*100

[10]: 97.47109826589595

[11]: clf = naive_bayes.MultinomialNB()
      clf.fit(X,y)

[11]: MultinomialNB
      MultinomialNB()

[12]: accuracy_score(y_test,clf.predict(X_test))*100

[12]: 98.77167630057804

[13]: filename = 'nlp_model.pkl'
      pickle.dump(clf, open(filename, 'wb'))
```

Figure 5.6: Testing the Accuracy

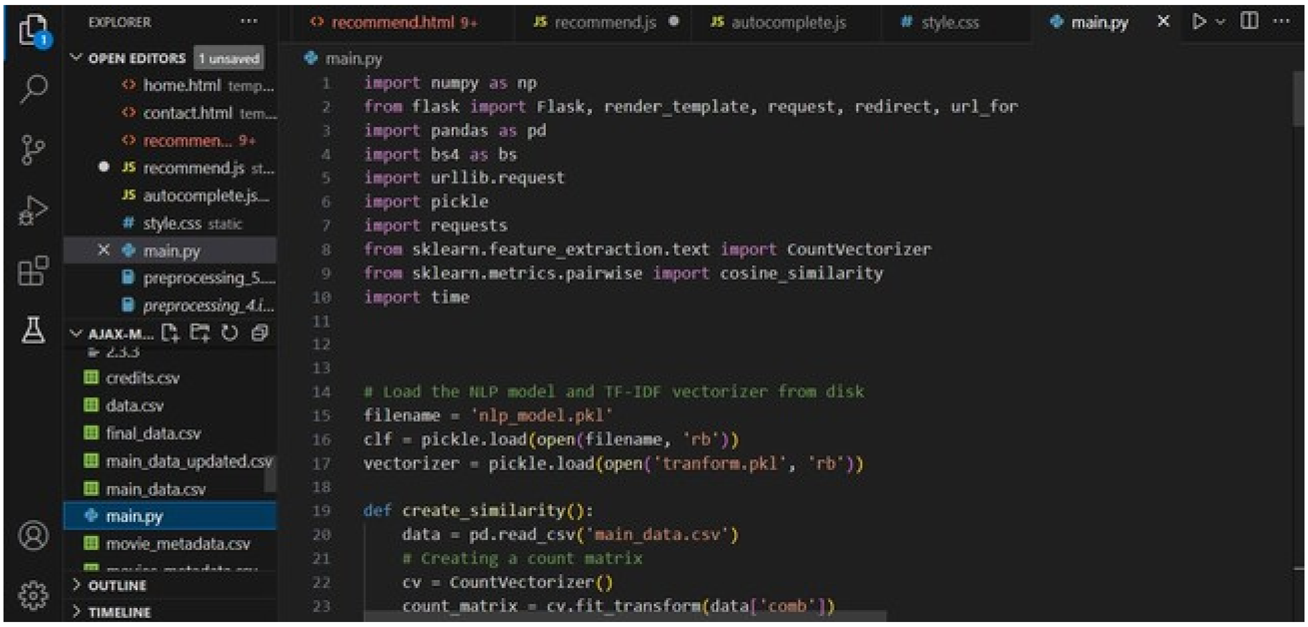
This continuation shows the fitting of the Naive Bayes model and evaluating its accuracy.Testing the Accuracy.



```html
<!DOCTYPE html>
<html>
<head>
  <title>BingeWatch</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Google Fonts -->
  <link href="https://fonts.googleapis.com/css?family=IBM+Plex+Sans&display=swap" rel="stylesheet
  <link href="https://fonts.googleapis.com/css2?family=Noto+Sans+JP&display=swap" rel="stylesheet

  <!-- Font Awesome -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font

  <!-- Bootstrap -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.

  <!-- Auto Complete -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@tarekraafat/autocomplete.js@7.2.0/di
  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">

  <style>
    .movie-card {
```

Figure 5.7: Importing fonts and Bootstrap

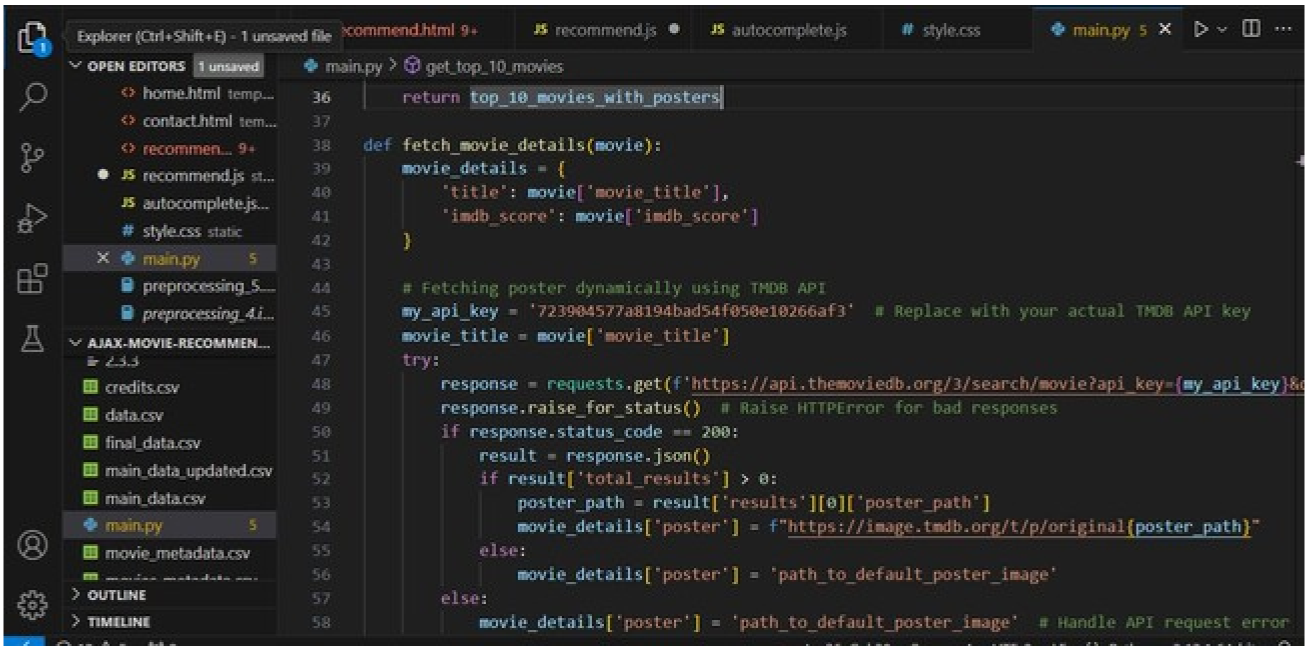
This image shows the code for importing fonts and the Bootstrap library.It sets up the basic styling framework for the web application.



```python
import numpy as np
from flask import Flask, render_template, request, redirect, url_for
import pandas as pd
import bs4 as bs
import urllib.request
import pickle
import requests
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import time



# Load the NLP model and TF-IDF vectorizer from disk
filename = 'nlp_model.pkl'
clf = pickle.load(open(filename, 'rb'))
vectorizer = pickle.load(open('tranform.pkl', 'rb'))

def create_similarity():
    data = pd.read_csv('main_data.csv')
    # Creating a count matrix
    cv = CountVectorizer()
    count_matrix = cv.fit_transform(data['comb'])
```

Figure 5.8: Importing libraries and Loading NPL model

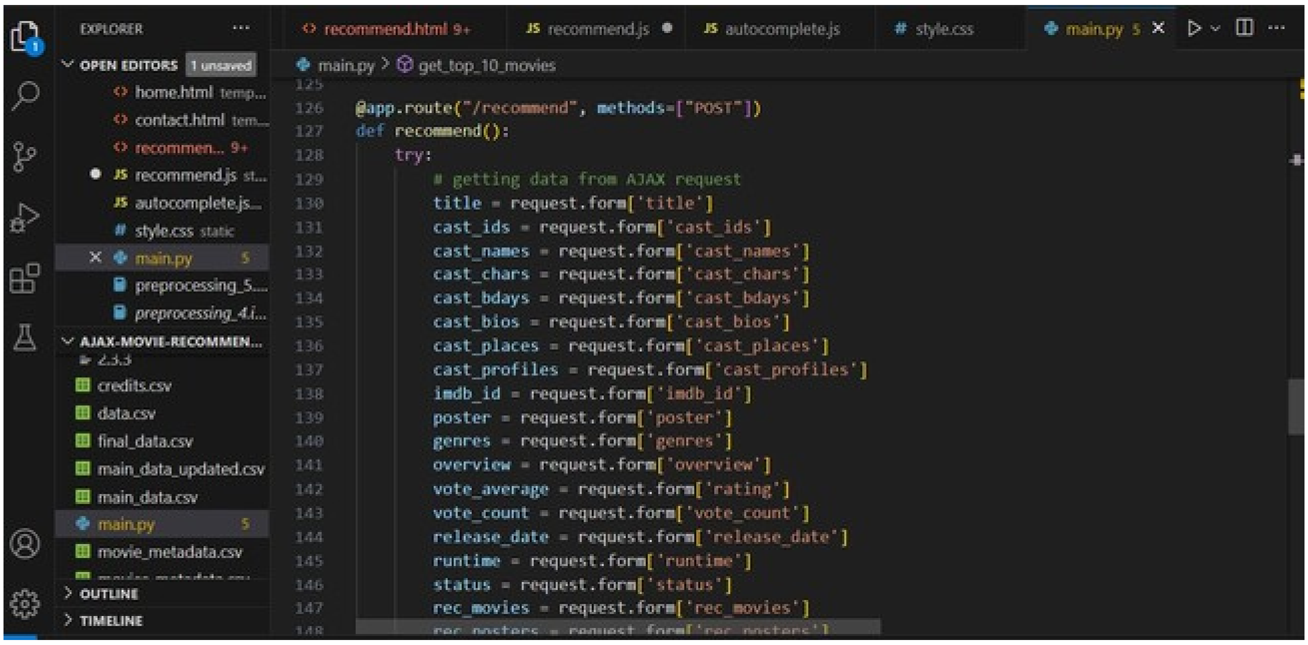
This figure displays the code for importing necessary libraries and loading the NLP model.The model is then used for further processing in the application.



```python
    return top_10_movies_with_posters

def fetch_movie_details(movie):
    movie_details = {
        'title': movie['movie_title'],
        'imdb_score': movie['imdb_score']
    }

    # Fetching poster dynamically using TMDB API
    my_api_key = '723904577a8194bad54f050e10266af3'  # Replace with your actual TMDB API key
    movie_title = movie['movie_title']
    try:
        response = requests.get(f'https://api.themoviedb.org/3/search/movie?api_key={my_api_key}&
        response.raise_for_status()  # Raise HTTPError for bad responses
        if response.status_code == 200:
            result = response.json()
            if result['total_results'] > 0:
                poster_path = result['results'][0]['poster_path']
                movie_details['poster'] = f"https://image.tmdb.org/t/p/original{poster_path}"
            else:
                movie_details['poster'] = 'path_to_default_poster_image'
        else:
            movie_details['poster'] = 'path_to_default_poster_image'  # Handle API request error
```

Figure 5.9: Fetching the posters from TMDB using API

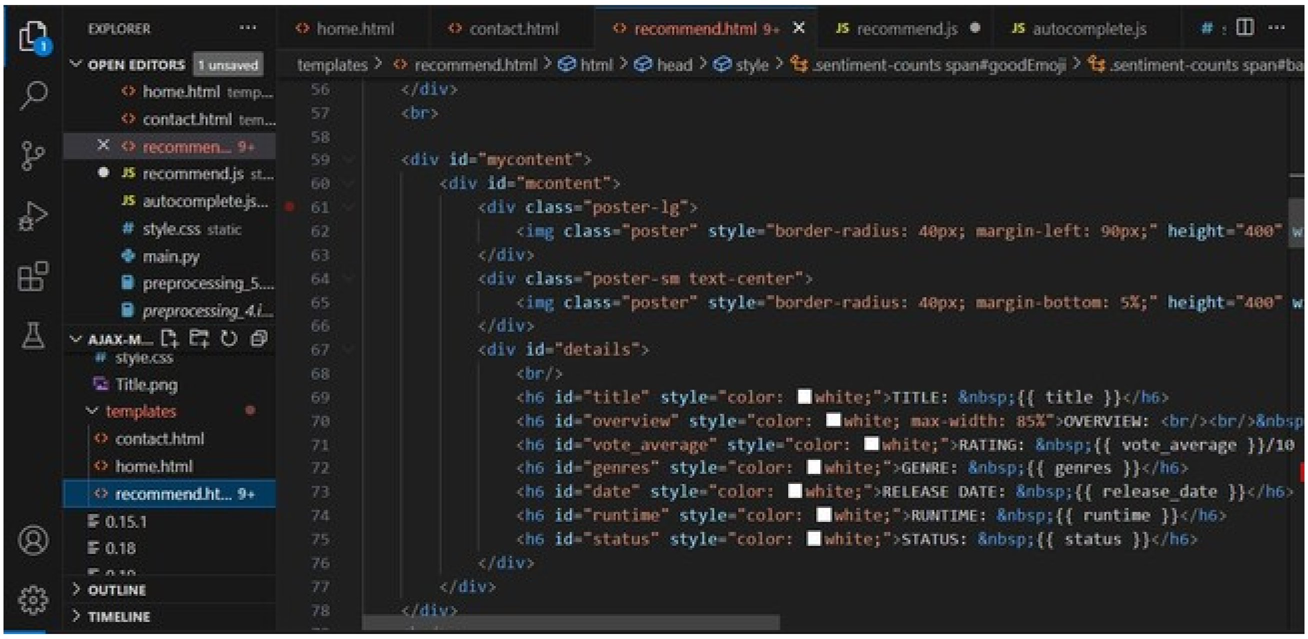
The code snippet here demonstrates fetching movie posters from The Movie Database (TMDB) using their API. It includes making API requests and handling the received data.



```
@app.route("/recommend", methods=["POST"])
def recommend():
    try:
        # getting data from AJAX request
        title = request.form['title']
        cast_ids = request.form['cast_ids']
        cast_names = request.form['cast_names']
        cast_chars = request.form['cast_chars']
        cast_bdays = request.form['cast_bdays']
        cast_bios = request.form['cast_bios']
        cast_places = request.form['cast_places']
        cast_profiles = request.form['cast_profiles']
        imdb_id = request.form['imdb_id']
        poster = request.form['poster']
        genres = request.form['genres']
        overview = request.form['overview']
        vote_average = request.form['rating']
        vote_count = request.form['vote_count']
        release_date = request.form['release_date']
        runtime = request.form['runtime']
        status = request.form['status']
        rec_movies = request.form['rec_movies']
        rec_posters = request.form['rec_posters']
```

Figure 5.10: Getting data from AJAX request

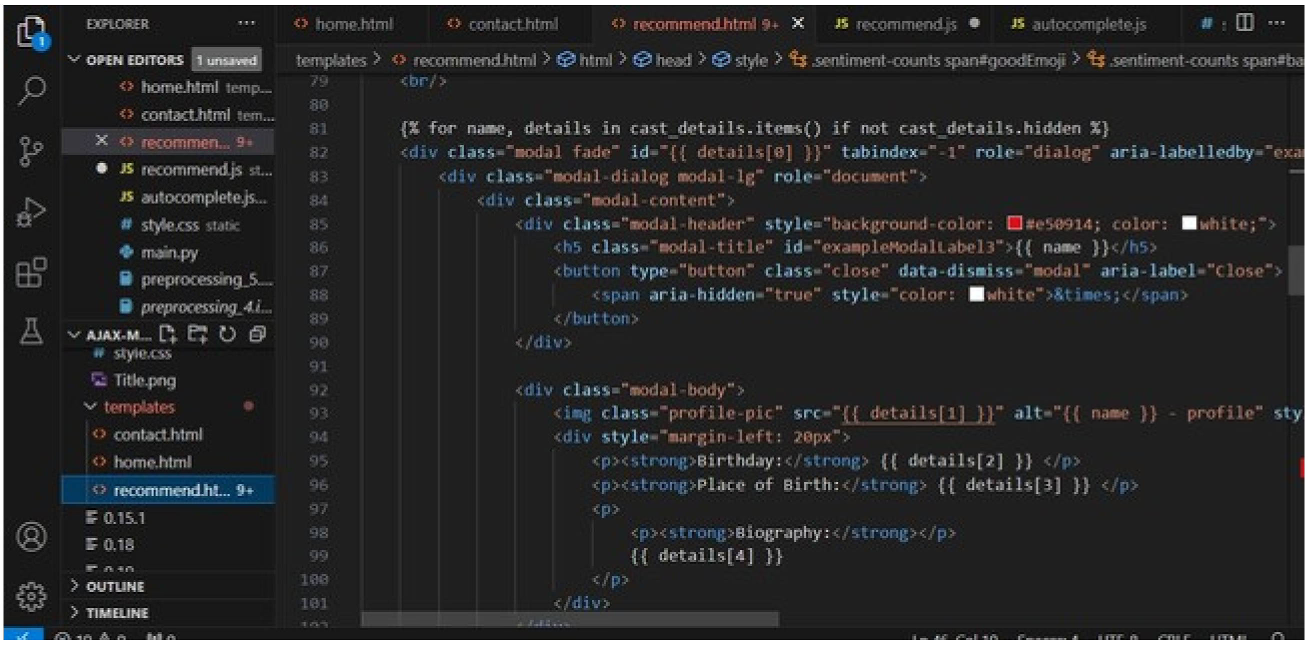
This image shows the code responsible for handling data received from an AJAX request.It processes the data and prepares it for further use in the application.



```
    </div>
    <br>

    <div id="mycontent">
        <div id="mcontent">
            <div class="poster-lg">
                <img class="poster" style="border-radius: 40px; margin-left: 90px;" height="400" w
            </div>
            <div class="poster-sm text-center">
                <img class="poster" style="border-radius: 40px; margin-bottom: 5%;" height="400" w
            </div>
            <div id="details">
                <br/>
                <h6 id="title" style="color: white;">TITLE: &nbsp;{{ title }}</h6>
                <h6 id="overview" style="color: white; max-width: 85%">OVERVIEW: <br/><br/>&nbsp
                <h6 id="vote_average" style="color: white;">RATING: &nbsp;{{ vote_average }}/10
                <h6 id="genres" style="color: white;">GENRE: &nbsp;{{ genres }}</h6>
                <h6 id="date" style="color: white;">RELEASE DATE: &nbsp;{{ release_date }}</h6>
                <h6 id="runtime" style="color: white;">RUNTIME: &nbsp;{{ runtime }}</h6>
                <h6 id="status" style="color: white;">STATUS: &nbsp;{{ status }}</h6>
            </div>
        </div>
    </div>
```

Figure 5.11: Recommendation Code

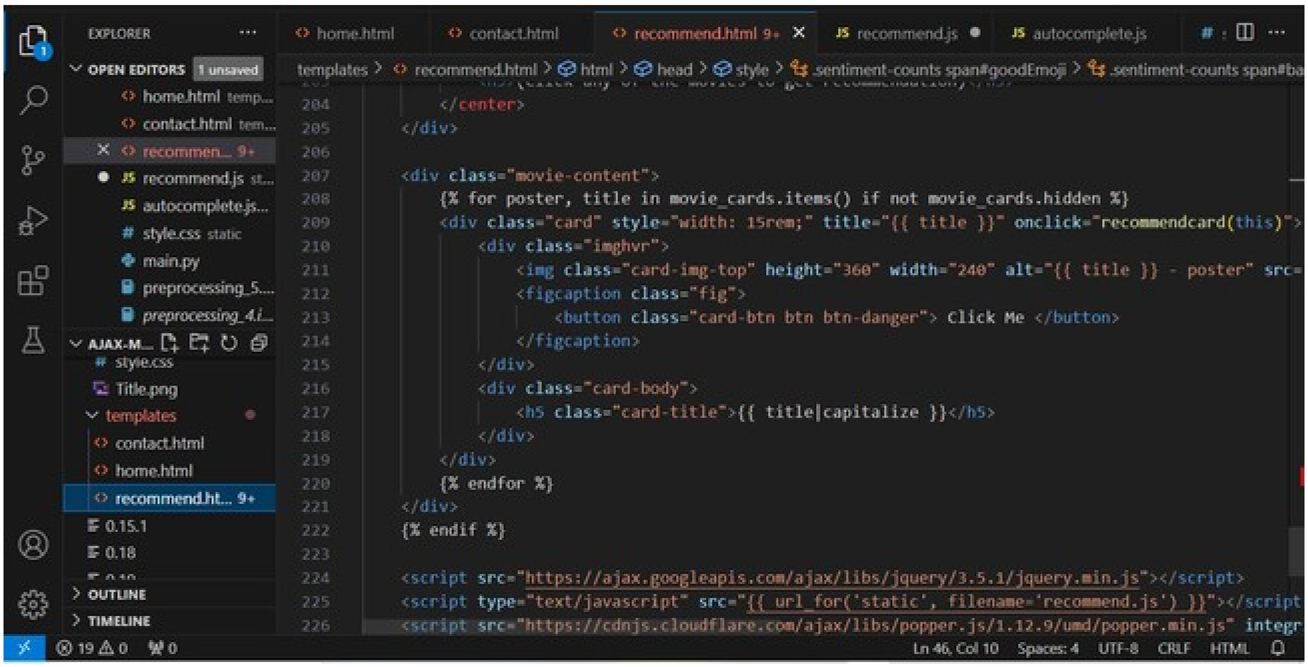
The figure displays the recommendation algorithm code. It calculates and generates movie recommendations based on the input data.



```
    <br/>

    {% for name, details in cast_details.items() if not cast_details.hidden %}
    <div class="modal fade" id="{{ details[0] }}" tabindex="-1" role="dialog" aria-labelledby="exa
        <div class="modal-dialog modal-lg" role="document">
            <div class="modal-content">
                <div class="modal-header" style="background-color: #e50914; color: white;">
                    <h5 class="modal-title" id="exampleModalLabel3">{{ name }}</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true" style="color: white">&times;</span>
                    </button>
                </div>

                <div class="modal-body">
                    <img class="profile-pic" src="{{ details[1] }}" alt="{{ name }} - profile" sty
                    <div style="margin-left: 20px">
                        <p><strong>Birthday:</strong> {{ details[2] }} </p>
                        <p><strong>Place of Birth:</strong> {{ details[3] }} </p>
                        <p>
                            <p><strong>Biography:</strong></p>
                            {{ details[4] }}
                        </p>
                    </div>
```

Figure 5.12: Recommendation Code

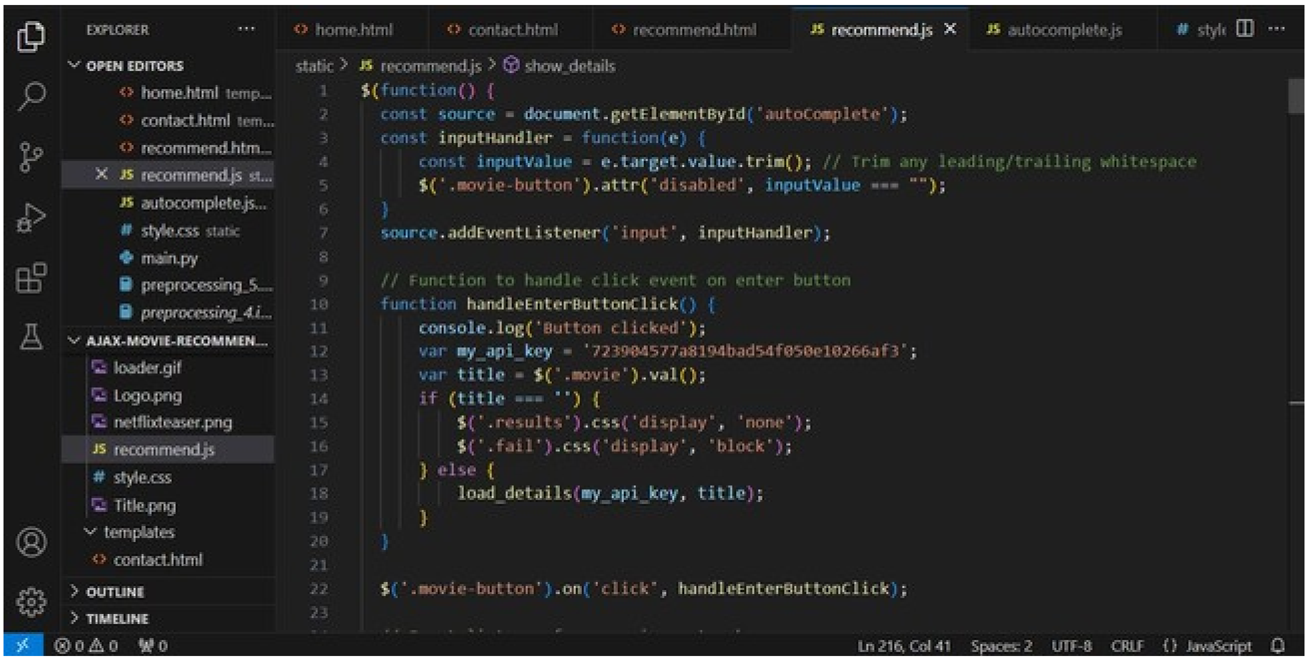
This continuation of the recommendation code further refines the recommendation logic. It processes additional data to enhance the recommendation accuracy.



```
        </center>
    </div>

    <div class="movie-content">
        {% for poster, title in movie_cards.items() if not movie_cards.hidden %}
        <div class="card" style="width: 15rem;" title="{{ title }}" onclick="recommendcard(this)">
            <div class="imghvr">
                <img class="card-img-top" height="360" width="240" alt="{{ title }} - poster" src=
                <figcaption class="fig">
                    <button class="card-btn btn btn-danger"> Click Me </button>
                </figcaption>
            </div>
            <div class="card-body">
                <h5 class="card-title">{{ title|capitalize }}</h5>
            </div>
        </div>
        {% endfor %}
    </div>
    {% endif %}

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script type="text/javascript" src="{{ url_for('static', filename='recommend.js') }}"></script
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integr
```

Figure 5.13: Recommendation Code

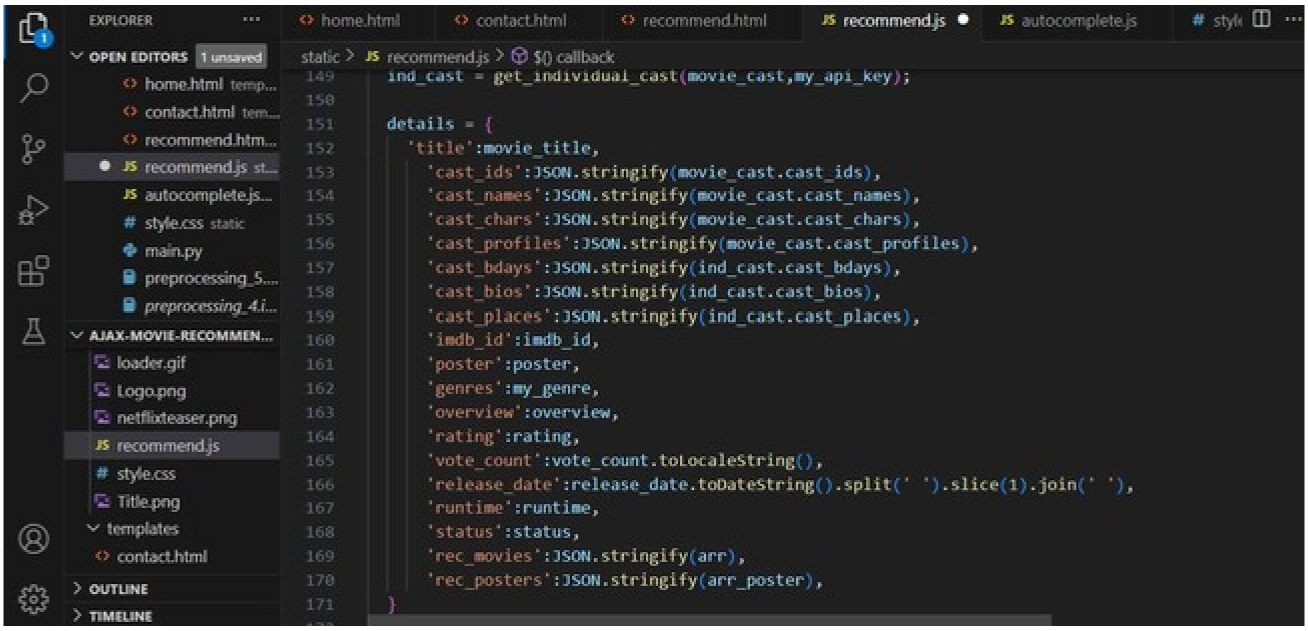
Another part of the recommendation code, focusing on integrating the recommendations into the application. It ensures the recommendations are displayed correctly to the user.



```
$(function() {
  const source = document.getElementById('autoComplete');
  const inputHandler = function(e) {
      const inputValue = e.target.value.trim(); // Trim any leading/trailing whitespace
      $('.movie-button').attr('disabled', inputValue === "");
  }
  source.addEventListener('input', inputHandler);

  // Function to handle click event on enter button
  function handleEnterButtonClick() {
      console.log('Button clicked');
      var my_api_key = '723904577a8194bad54f050e10266af3';
      var title = $('.movie').val();
      if (title === '') {
          $('.results').css('display', 'none');
          $('.fail').css('display', 'block');
      } else {
          load_details(my_api_key, title);
      }
  }

  $('.movie-button').on('click', handleEnterButtonClick);
```

Figure 5.14: JavaScript for Enter Button

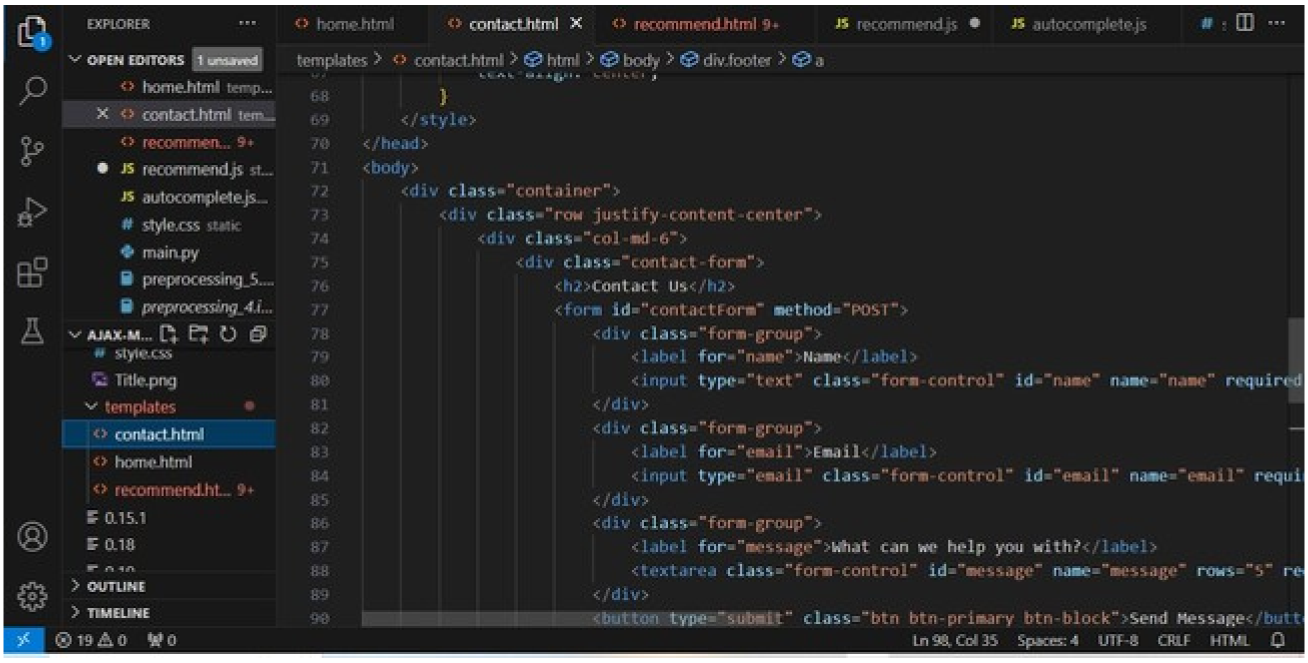
This image shows the JavaScript code for handling the ”Enter” button functionality.It triggers the recommendation process when the user presses the ”Enter” key.



```
  ind_cast = get_individual_cast(movie_cast,my_api_key);

  details = {
    'title':movie_title,
      'cast_ids':JSON.stringify(movie_cast.cast_ids),
      'cast_names':JSON.stringify(movie_cast.cast_names),
      'cast_chars':JSON.stringify(movie_cast.cast_chars),
      'cast_profiles':JSON.stringify(movie_cast.cast_profiles),
      'cast_bdays':JSON.stringify(ind_cast.cast_bdays),
      'cast_bios':JSON.stringify(ind_cast.cast_bios),
      'cast_places':JSON.stringify(ind_cast.cast_places),
      'imdb_id':imdb_id,
      'poster':poster,
      'genres':my_genre,
      'overview':overview,
      'rating':rating,
      'vote_count':vote_count.toLocaleString(),
      'release_date':release_date.toDateString().split(' ').slice(1).join(' '),
      'runtime':runtime,
      'status':status,
      'rec_movies':JSON.stringify(arr),
      'rec_posters':JSON.stringify(arr_poster),
  }
```

Figure 5.15: Data after clicking the enter button

The figure shows the data handling process after the ”Enter” button is clicked.It manages the data flow and updates the application accordingly.



```html
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-6">
                <div class="contact-form">
                    <h2>Contact Us</h2>
                    <form id="contactForm" method="POST">
                        <div class="form-group">
                            <label for="name">Name</label>
                            <input type="text" class="form-control" id="name" name="name" required
                        </div>
                        <div class="form-group">
                            <label for="email">Email</label>
                            <input type="email" class="form-control" id="email" name="email" requi
                        </div>
                        <div class="form-group">
                            <label for="message">What can we help you with?</label>
                            <textarea class="form-control" id="message" name="message" rows="5" re
                        </div>
                        <button type="submit" class="btn btn-primary btn-block">Send Message</butt
```

Figure 5.16: Contact Page

This figure displays the contact page of the application. It includes the layout and styling for the contact form.

Chapter 6

System Implementation

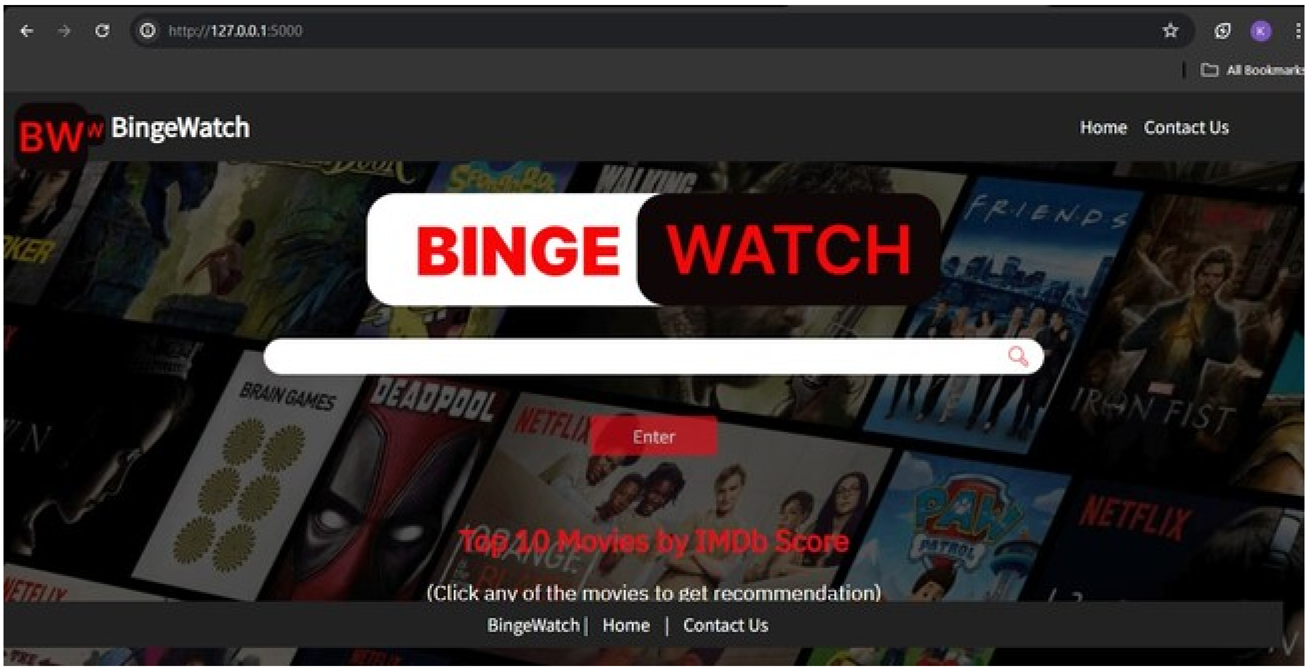

http://127.0.0.1:5000
All Bookmarks
BingeWatch
Home
Contact Us
BINGE WATCH
Enter
Top 10 Movies by IMDb Score
(Click any of the movies to get recommendation)
BingeWatch | Home | Contact Us

Figure 6.1: Home Page
This figure displays the home page of the movie recommendation system. It provides an overview and access to various features of the application.

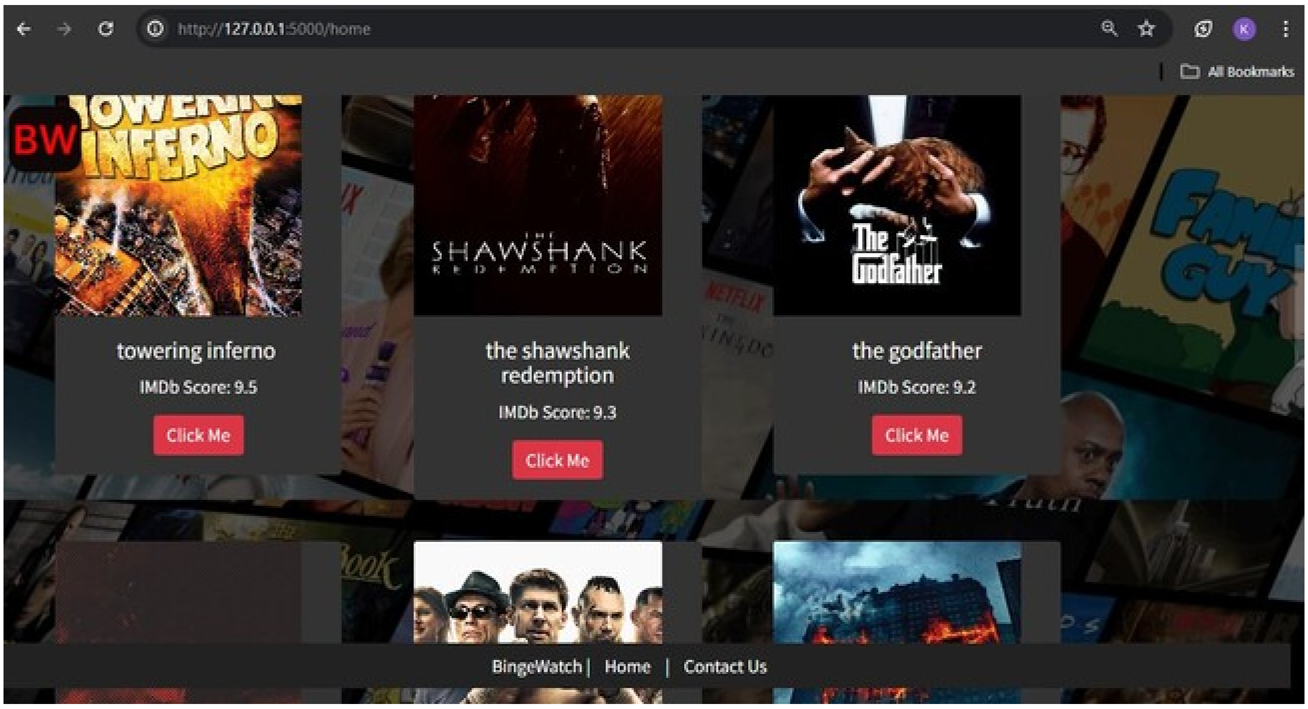


Figure 6.2: Top 10 Movies according to Top Ratings
This image shows a list of the top 10 movies based on ratings. It helps users quickly find highly-rated movies.

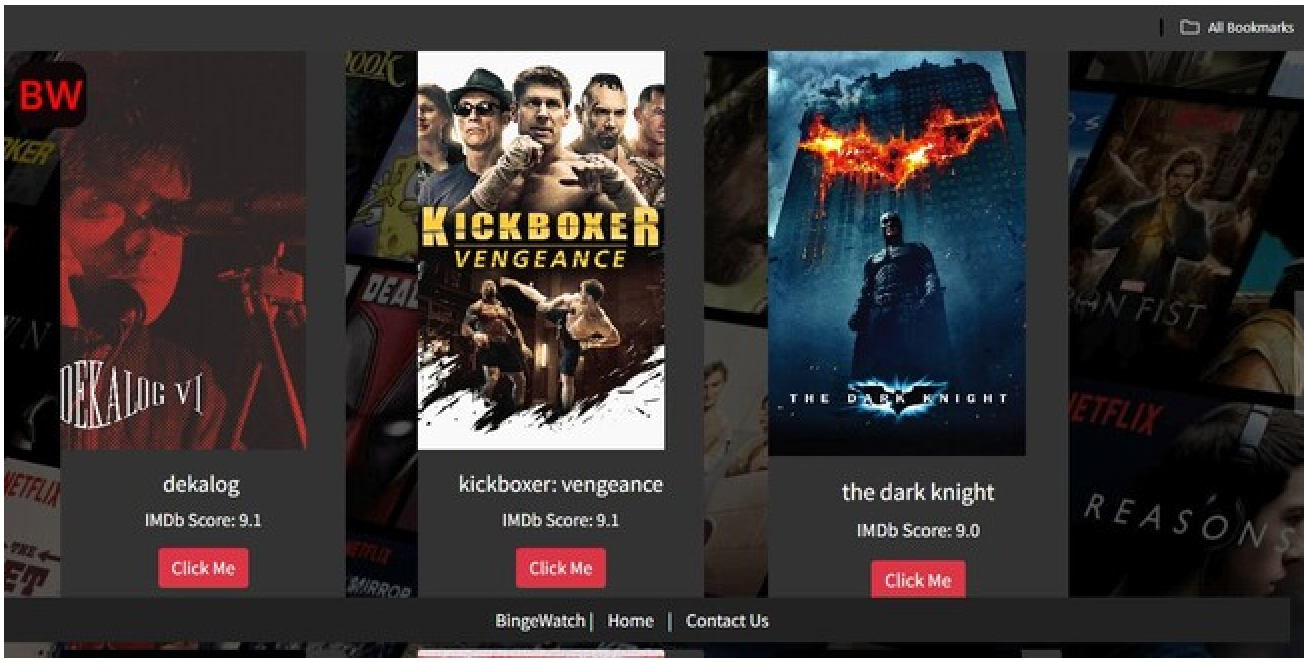


Figure 6.3: Top 10 Movies according to the Top Ratings
Another view of the top 10 movies according to user ratings. It reinforces the visibility of highly-rated content for users.

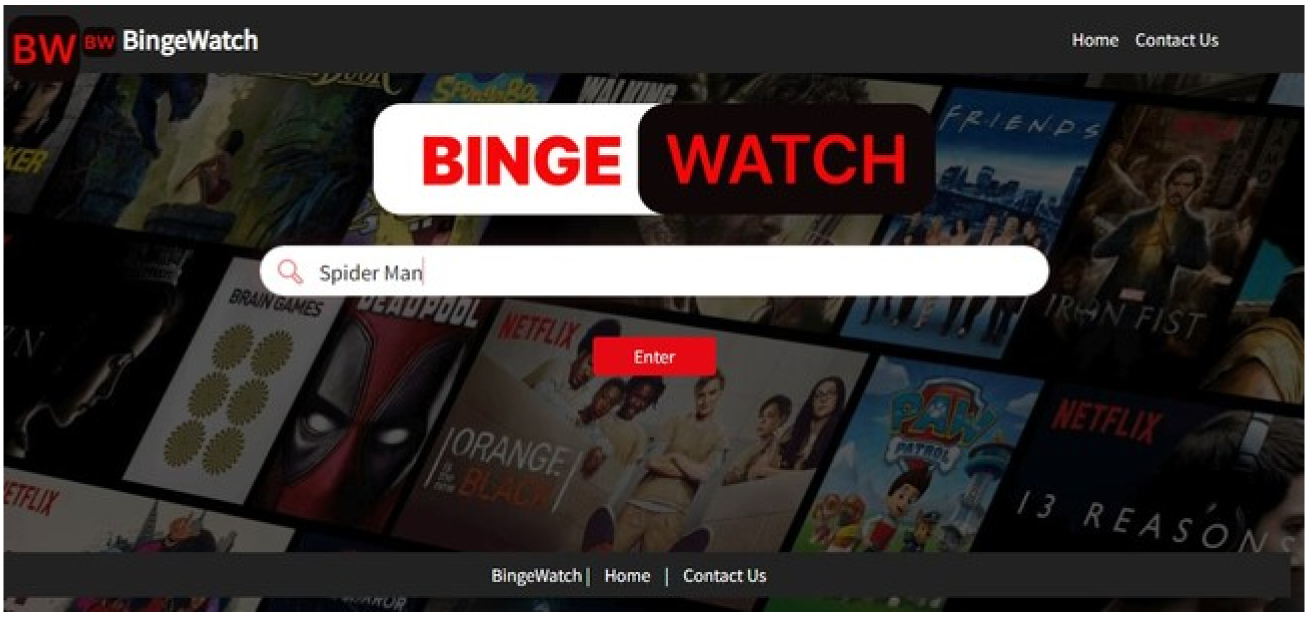
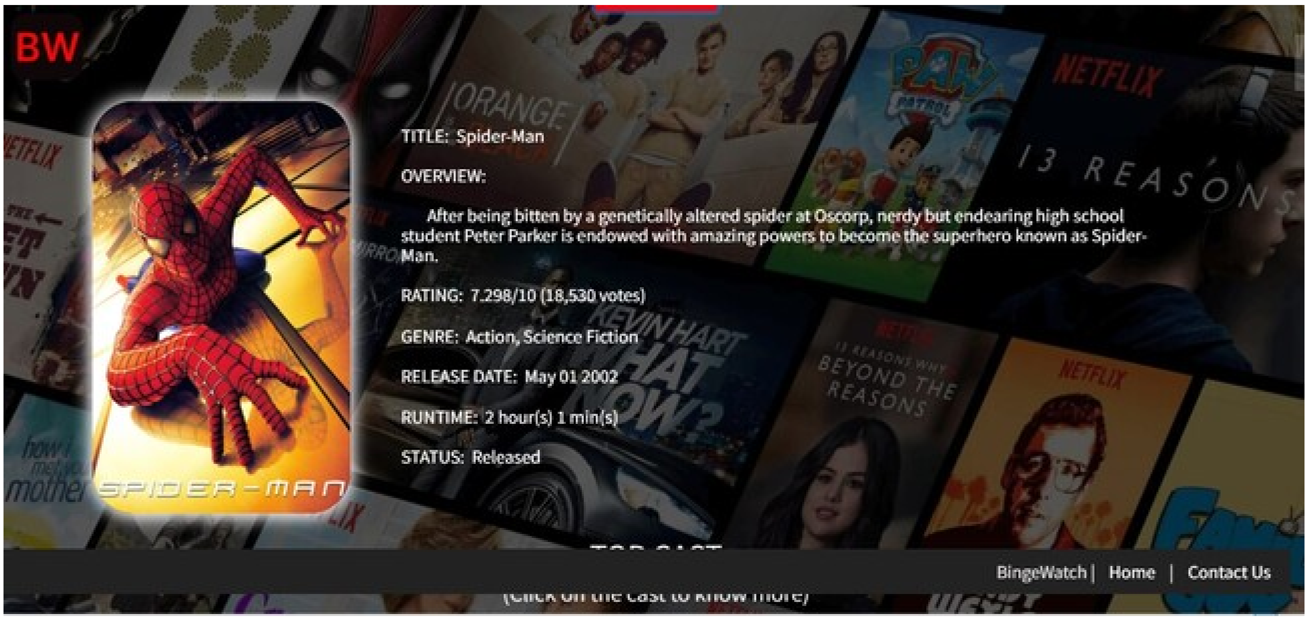


Figure 6.4: Search Bar where user can search movies
The search bar allows users to find movies by entering keywords. It provides a convenient way to navigate the movie dataset.



TITLE: Spider-Man

OVERVIEW:

After being bitten by a genetically altered spider at Oscorp, nerdy but endearing high school student Peter Parker is endowed with amazing powers to become the superhero known as Spider-Man.

RATING: 7.298/10 (18,530 votes)

GENRE: Action, Science Fiction

RELEASE DATE: May 01 2002

RUNTIME: 2 hour(s) 1 min(s)

STATUS: Released

Figure 6.5: The movie which user searched
This figure shows the result of a user's search for a specific movie. It displays detailed information about the searched movie.

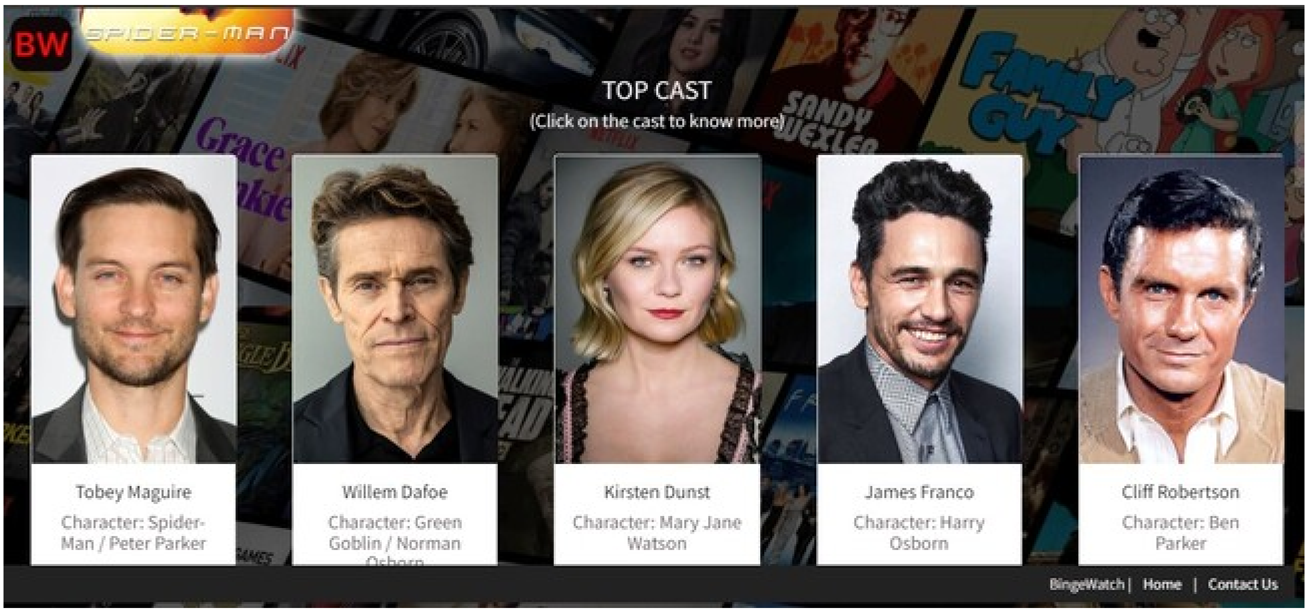
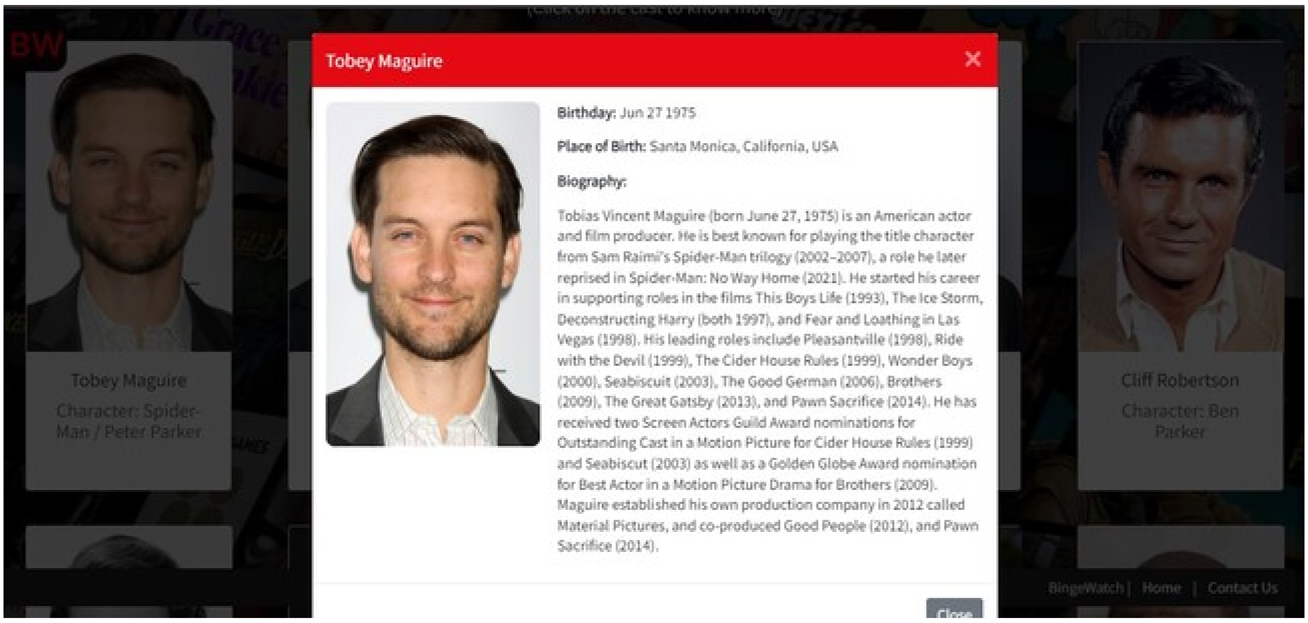

BW
SPIDER-MAN
TOP CAST
(Click on the cast to know more)
Tobey Maguire
Character: Spider-Man / Peter Parker
Willem Dafoe
Character: Green Goblin / Norman Osborn
Kirsten Dunst
Character: Mary Jane Watson
James Franco
Character: Harry Osborn
Cliff Robertson
Character: Ben Parker
BingeWatch | Home | Contact Us

Figure 6.6: Cast for the movie
This image lists the cast members of the searched movie. It helps users see the actors involved in the movie.


Tobey Maguire

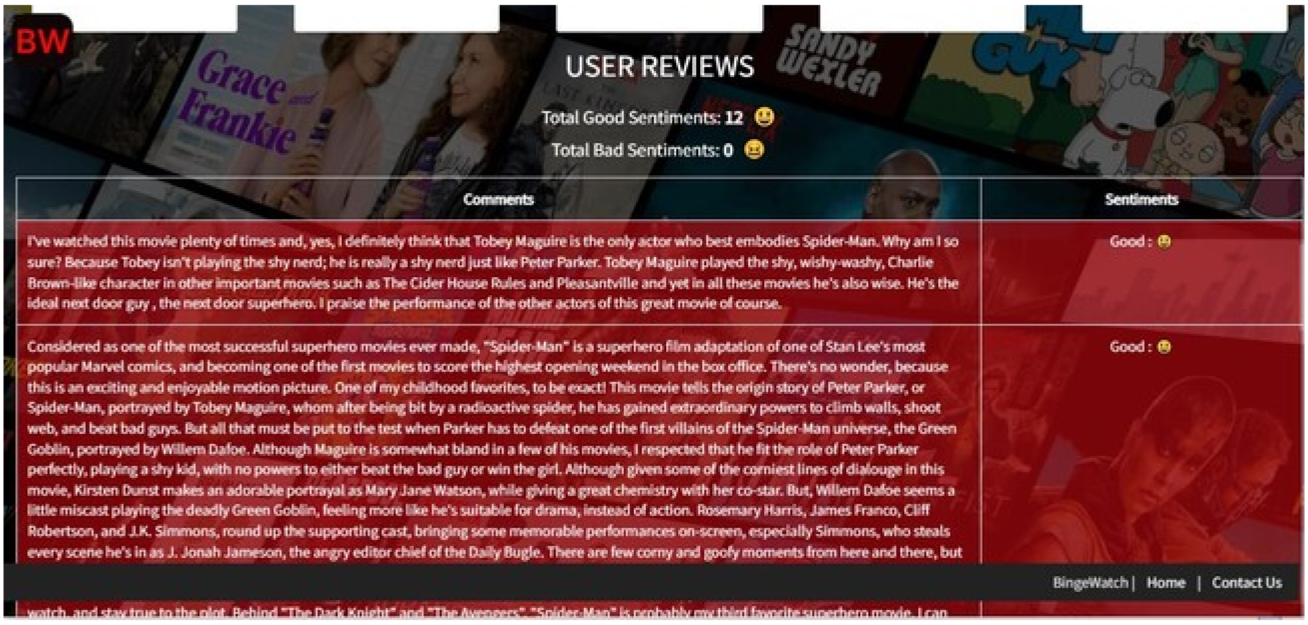
Birthday: Jun 27 1975

Place of Birth: Santa Monica, California, USA

Biography:

Tobias Vincent Maguire (born June 27, 1975) is an American actor and film producer. He is best known for playing the title character from Sam Raimi's Spider-Man trilogy (2002–2007), a role he later reprised in Spider-Man: No Way Home (2021). He started his career in supporting roles in the films This Boys Life (1993), The Ice Storm, Deconstructing Harry (both 1997), and Fear and Loathing in Las Vegas (1998). His leading roles include Pleasantville (1998), Ride with the Devil (1999), The Cider House Rules (1999), Wonder Boys (2000), Seabiscuit (2003), The Good German (2006), Brothers (2009), The Great Gatsby (2013), and Pawn Sacrifice (2014). He has received two Screen Actors Guild Award nominations for Outstanding Cast in a Motion Picture for Cider House Rules (1999) and Seabiscut (2003) as well as a Golden Globe Award nomination for Best Actor in a Motion Picture Drama for Brothers (2009). Maguire established his own production company in 2012 called Material Pictures, and co-produced Good People (2012), and Pawn Sacrifice (2014).

Figure 6.7: Description about the movie
This figure provides a detailed description of the selected movie. It includes information like the plot summary and other relevant details.



Total Good Sentiments: 12

Total Bad Sentiments: 0

Comments	Sentiments
I've watched this movie plenty of times and, yes, I definitely think that Tobey Maguire is the only actor who best embodies Spider-Man. Why am I so sure? Because Tobey isn't playing the shy nerd; he is really a shy nerd just like Peter Parker. Tobey Maguire played the shy, wishy-washy, Charlie Brown-like character in other important movies such as The Cider House Rules and Pleasantville and yet in all these movies he's also wise. He's the ideal next door guy , the next door superhero. I praise the performance of the other actors of this great movie of course.	Good :
Considered as one of the most successful superhero movies ever made, "Spider-Man" is a superhero film adaptation of one of Stan Lee's most popular Marvel comics, and becoming one of the first movies to score the highest opening weekend in the box office. There's no wonder, because this is an exciting and enjoyable motion picture. One of my childhood favorites, to be exact! This movie tells the origin story of Peter Parker, or Spider-Man, portrayed by Tobey Maguire, whom after being bit by a radioactive spider, he has gained extraordinary powers to climb walls, shoot web, and beat bad guys. But all that must be put to the test when Parker has to defeat one of the first villains of the Spider-Man universe, the Green Goblin, portrayed by Willem Dafoe. Although Maguire is somewhat bland in a few of his movies, I respected that he fit the role of Peter Parker perfectly, playing a shy kid, with no powers to either beat the bad guy or win the girl. Although given some of the corniest lines of dialouge in this movie, Kirsten Dunst makes an adorable portrayal as Mary Jane Watson, while giving a great chemistry with her co-star. But, Willem Dafoe seems a little miscast playing the deadly Green Goblin, feeling more like he's suitable for drama, instead of action. Rosemary Harris, James Franco, Cliff Robertson, and J.K. Simmons, round up the supporting cast, bringing some memorable performances on-screen, especially Simmons, who steals every scene he's in as J. Jonah Jameson, the angry editor chief of the Daily Bugle. There are few corny and goofy moments from here and there, but watch, and stay true to the plot. Behind "The Dark Knight" and "The Avengers", "Spider-Man" is probably my third favorite superhero movie. I can	Good :

Figure 6.8: User reviews with sentiment analysis
This image shows user reviews for the movie along with sentiment analysis. It helps users gauge the general reception of the movie.

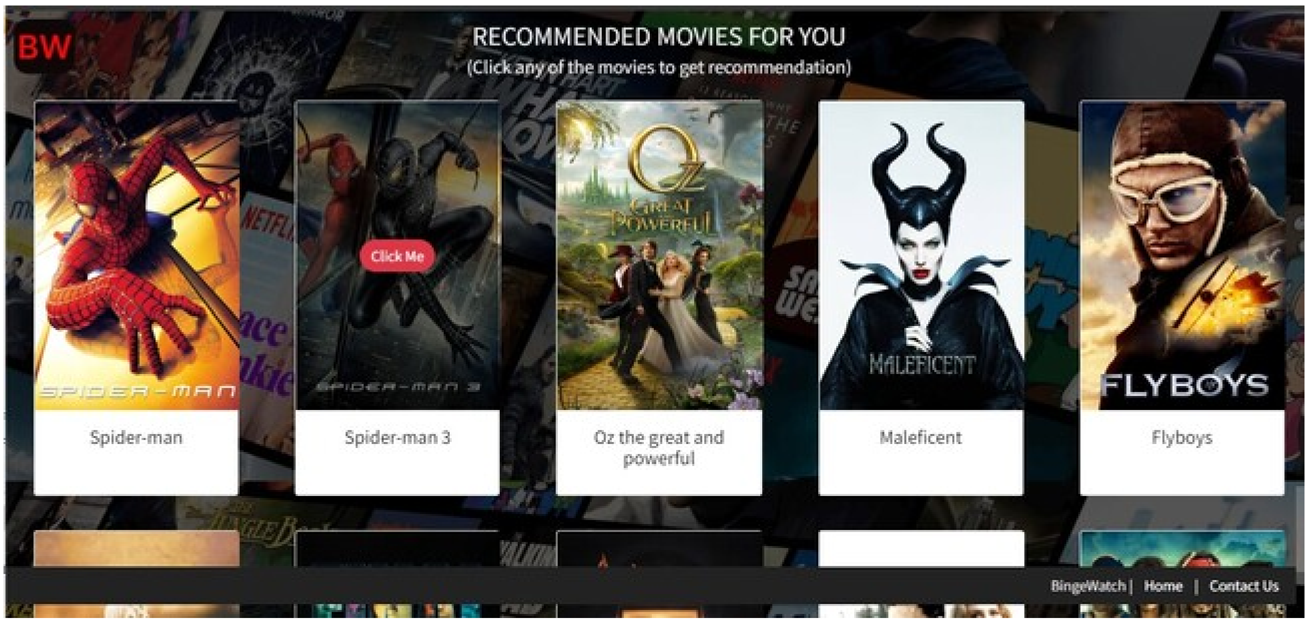
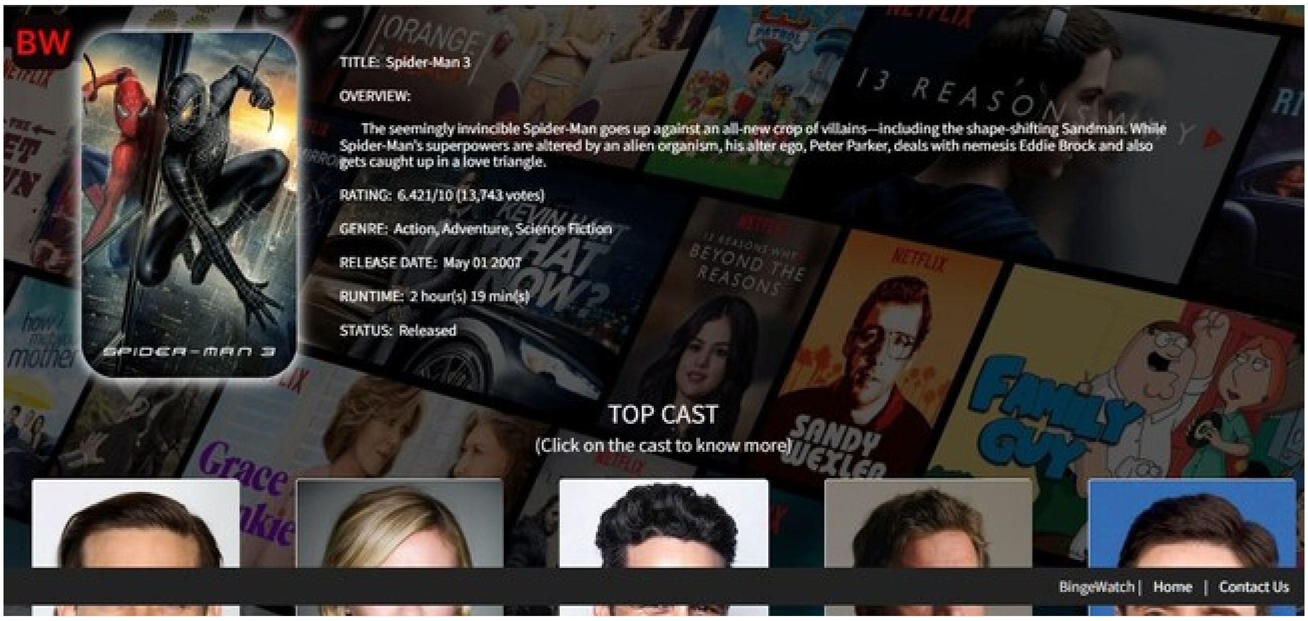

BW
RECOMMENDED MOVIES FOR YOU
(Click any of the movies to get recommendation)
Click Me
Spider-man
Spider-man 3
Oz the great and powerful
Maleficent
Flyboys
BingeWatch | Home | Contact Us

Figure 6.9: Recommendations for other Movies
This figure displays movie recommendations based on user preferences. It offers suggestions for other movies that the user might enjoy.


TITLE: Spider-Man 3

OVERVIEW:

The seemingly invincible Spider-Man goes up against an all-new crop of villains—including the shape-shifting Sandman. While Spider-Man's superpowers are altered by an alien organism, his alter ego, Peter Parker, deals with nemesis Eddie Brock and also gets caught up in a love triangle.

RATING: 6.421/10 (13,743 votes)

GENRE: Action, Adventure, Science Fiction

RELEASE DATE: May 01 2007

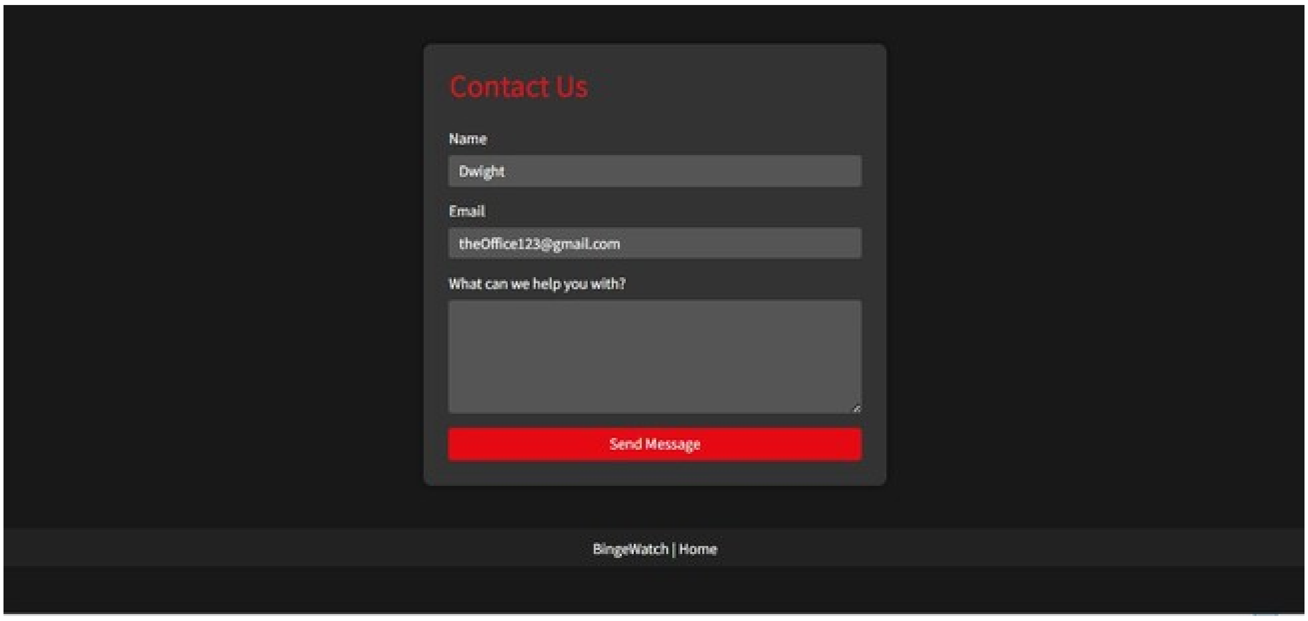
RUNTIME: 2 hour(s) 19 min(s)

STATUS: Released

TOP CAST

(Click on the cast to know more)

Figure 6.10: After Clicking on one of the Recommended Movie
This image shows the detailed view after a user clicks on a recommended movie. It provides additional information and options related to the recommended movie.


Contact Us

Name

Dwight

Email

theOffice123@gmail.com

What can we help you with?

Figure 6.11: Contact Page
This figure shows the contact page of the application. It includes a form for users to reach out for support or feedback.

Chapter 7

System Testing

7.1 Testing Methodologies

The testing phase of BingeWatch underwent a comprehensive approach encompassing various methodologies to ensure the system's reliability and functionality.

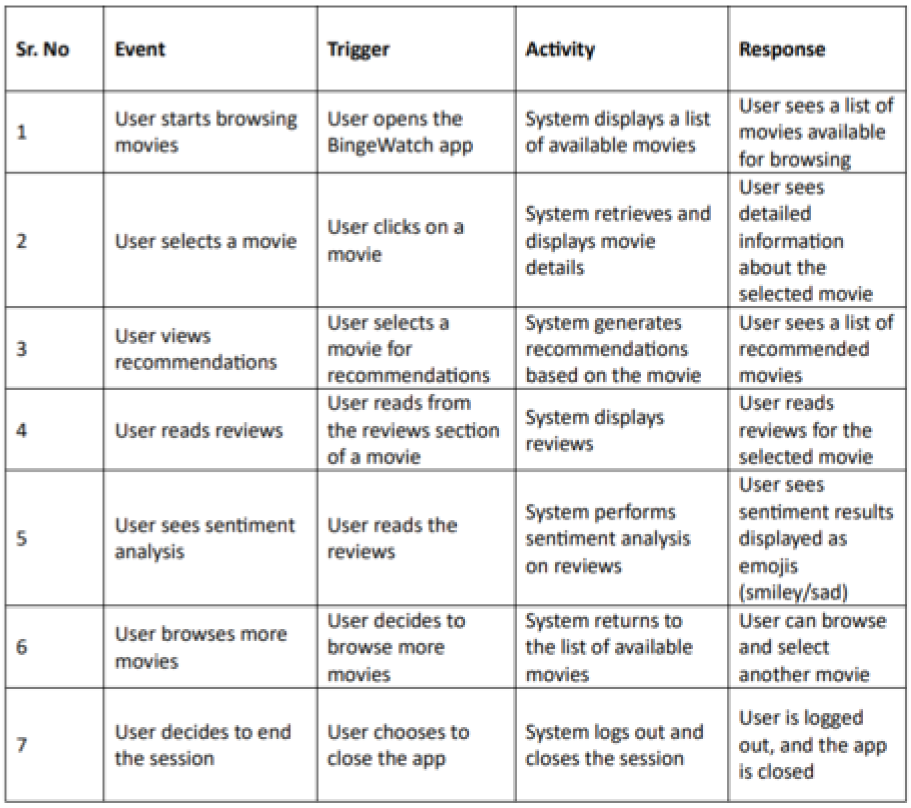
Unit Testing was employed to scrutinize individual components, verifying that each function and feature operated as intended in isolation. Integration Testing focused on examining the seamless interaction between different components, ensuring the overall integrity of the system when combined.

System Testing assessed the application as a whole, validating its compliance with specified requirements and testing end-to-end scenarios for cohesive system behavior. User Acceptance Testing (UAT) involved actual users engaging with the system to confirm its usability, effectiveness, and overall user satisfaction.

Performance Testing was conducted to evaluate the responsiveness, stability, and scalability of MyExpensePal under various conditions, ensuring optimal performance during peak usage. Security Testing was a critical aspect, aiming to identify and rectify vulnerabilities to safeguard user data and prevent unauthorized access.

Regression Testing was consistently implemented to detect unintended side effects resulting from updates or modifications, ensuring that new changes did not adversely impact existing functionalities. The combination of these methodologies contributed to the delivery of a robust and reliable MyExpensePal system, meeting high-quality standards.

7.2 Test Cases



Sr. No	Event	Trigger	Activity	Response
1	User starts browsing movies	User opens the BingeWatch app	System displays a list of available movies	User sees a list of movies available for browsing
2	User selects a movie	User clicks on a movie	System retrieves and displays movie details	User sees detailed information about the selected movie
3	User views recommendations	User selects a movie for recommendations	System generates recommendations based on the movie	User sees a list of recommended movies
4	User reads reviews	User reads from the reviews section of a movie	System displays reviews	User reads reviews for the selected movie
5	User sees sentiment analysis	User reads the reviews	System performs sentiment analysis on reviews	User sees sentiment results displayed as emojis (smiley/sad)
6	User browses more movies	User decides to browse more movies	System returns to the list of available movies	User can browse and select another movie
7	User decides to end the session	User chooses to close the app	System logs out and closes the session	User is logged out, and the app is closed

Table 7.1: Test case used in BingeWatch

This table depicts the user activities for the BingeWatch project, detailing each event, trigger, activity, and system response.

Chapter 8

Limitation and Future Enhancement

8.1 Limitation

Despite the strides made in developing BingeWatch, there are certain limitations to acknowledge. The software's effectiveness may be contingent on user familiarity with digital platforms, potentially excluding individuals with limited technological proficiency. Additionally, the reliance on predefined reviews for sentiment analysis may not capture the full spectrum of user opinions and emotions, potentially limiting the accuracy of the sentiment feedback. Furthermore, BingeWatch's compatibility is restricted to systems that meet specific hardware requirements, such as a processor with 2.40 GHz speed and a minimum of 8 GB RAM, potentially excluding users with less capable hardware. Acknowledging these limitations provides insights for future enhancements and ensures transparency about the current scope of BingeWatch. Periodically updating the application to incorporate advanced features based on user feedback and market trends will be essential to overcoming these limitations.

8.2 Future Enhancement

BingeWatch, as a forward-looking movie recommendation system, envisions several future enhancements. One key area of development involves the introduction of more sophisticated recommendation algorithms, incorporating real-time data analysis to provide even more personalized content suggestions. To streamline the user experience, the application aims to integrate automated sentiment analysis using advanced natural language processing techniques, reducing the reliance on predefined reviews and capturing a wider range of user feedback.

Looking towards a more collaborative future, BingeWatch aims to introduce features that facilitate shared viewing experiences and recommendations for groups of users with similar tastes. This includes functionalities catering to families, friends, or communities with shared interests. Additionally, the application plans to integrate educational resources within its interface, promoting awareness and understanding of diverse genres and cinematic techniques through articles, videos, and interactive content. BingeWatch's future enhancements aim to continually align with user feedback, industry trends, and the overarching strategic objectives of the application. Through regular updates and continuous improvement, the platform seeks to maintain long-term success and high levels of user satisfaction.

References

[1] Prashant Chaudhary. Movie recommender systems: Concepts, methods, challenges, and future directions. *National Center for Biotechnology Information*, 2022.

[2] Michael Schrage. *Recommendation Engines.* The MIT Press Essential Knowledge series, 2020.

[3] Prashant Chaudhary. Movie recommender system using sentiment analysis. 2021.