Tribhuvan University

Academia International College

Final Year Project Report
On
Movies Recommendation System (Movieplex)
[CSC 412]

Under the supervision of

"Mr. Ganesh Prasad Bhatta"

Submitted By

Aashish Bari (T.U. Exam Roll No. 26474/077)

Ashish Kumar Tharu (T.U. Exam Roll No. 26481/077)

Elin Bikram Oli (T.U. Exam Roll No. 26489/077)

Submitted To

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

January, 2025

Tribhuvan University

Academia International College

Final Year Project Report
On
Movies Recommendation System (Movieplex)
[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

Submitted by

Aashish Bari (T.U. Exam Roll No. 26474/077)

Ashish Kumar Tharu (T.U. Exam Roll No. 26481/077)

Elin Bikram Oli (T.U. Exam Roll No. 26489/077)

Submitted to

Department of Computer Science and Information Technology Academia International College

Institute of Science and Technology

Tribhuvan University

January, 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision by Mr. Aashish Bari (26474/077), Mr. Ashish Kumar Tharu (26481/077), and Mr. Elin Bikram Oli (26489/077) entitled "Movies Recommendation System (Movieplex)" be accepted as fulfilling in partial requirements for the degree of Bachelors of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

……

Mr. Ganesh Bhatta

Project Supervisor

Department of Computer Science & IT

Academia International College

Gwarko, Lalitpur

Tribhuvan University

Department of Computer Science and Information Technology

Academia International College

Certificate of Approval

This is to certify that this project prepared by Mr. Aashish Bari, Mr. Ashish Kumar Tharu and Mr. Elin Bikram Oli entitled "Movies Recommendation System (Movieplex)" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. Mr. Ganesh Prasad Bhatta Project Supervisor Department of Computer Science and IT Academia International College	…….. Mr. Bishwas Mathema HOD/Program Coordinator Department of Computer Science and IT Academia International College
…….. Internal Examiner Academia International College	…….. External Examiner Central Department of CSIT Tribhuvan University

Acknowledgement

We would like to thank to everyone who assisted us in finishing this report. We would like to deeply thank our project supervisor, Mr. Ganesh Bhatta, for his leadership and assistance in guiding us to plan our work and write this report. We are also thankful to the Academia International College team for giving us the support to finish this project. The B.Sc. CSIT department's teaching and non-teaching staff deserve special appreciation for their steady support and help. We appreciate the thoughtful analysis, we received from the panels and supervisors throughout our project presentation, which helped us to develop our abilities. Finally, we would like to express our thankfulness to our friends for their support and encouragement during the project. We sincerely appreciate everyone's assistance and efforts.

With respect,

Aashish Bari (26474/077)

Ashish Kumar Tharu (26481/077)

Elin Bikram Oli (26489/077)

Abstract

This project presents a Movies Recommendation System designed to enhance user experience by suggesting movies based on user preferences and choices. The system implements K-Nearest Neighbors (KNN) algorithm, to recommend similar movies based on the selected movie's genre, plot, and other features. The system also uses TF-IDF (Term Frequency-Inverse Document Frequency) to analyze movie descriptions (plot keywords) and calculates the similarity between movies using Cosine Similarity. Users can receive recommendations based on selected movies or genres, with an changing number of suggestions according to their preferences.

The system further enhances the recommendation process by integrating the OMDB API to fetch additional movie details, such as the director, cast, plot, IMDb ratings, and posters, providing users with a complete overview of each recommended movie. This platform is developed using Streamlit, features a user-friendly interface that allows users to explore recommendations dynamically. The system supports both content-based recommendations and genre-based recommendations, for various user preferences.

Keywords: K-Nearest Neighbors (KNN), TF-IDF (Term Frequency-Inverse Document Frequency), Cosine Similarity, Movie Recommendation, OMDB API, Content-Based Filtering, Streamlit.

Table of Contents

List of Abbreviations

API	Application Programming Interface
CBF	Content-Based Filtering
CF	Collaborative Filtering
CSS	Cascading Style Sheets
HF	Hybrid Filtering
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
IMDB	Internet Movie Database
KNN	K-Nearest Neighbor
NLP	Natural Language Processing
OMDB	Open Movie Database
OS	Operating System
TF-IDF	Term Frequency-Inverse Document Frequency
TMDB	The Movie Database
UI	User Interface
UML	Unified Modelling Language

List of Figures

List of Tables

Chapter 1: Introduction

1.1. Introduction

The Movie Recommender System is a customized system made to make choosing movies easier in a time when streaming services offer a multiple number of options. Using a content-based filtering, the system examines user's movies taste, IMDb ratings, and genres to suggest films. Its advanced features and easy-to-use interface are designed to improve the movie selection process and make it quick and easy for users to locate movies they like.

1.2. Problem Statement

Users frequently struggle to discover a movie that suits their interests due to the abundance of options accessible on digital streaming services. Current recommendation systems usually don't include audience feedback well and aren't personalized. By offering personalized recommendations of user evaluations, the Movie Recommender System tackles these problems and guarantees more appropriate and knowledgeable movie recommendations.

1.3. Objectives

The project aims to meet the following objectives:

1. To use different algorithms to recommend movies based on similarities in features like genres and IMDb ratings.
2. To allow users to get recommendation of movies based on their input (either a selected movie or a genre preference).
3. To give additional movie information such as directors, cast, plot summaries, IMDb ratings, and posters, for better user experience.
4. To build an interactive and user-friendly application for user to select their preferences and explore the recommendations.
5. To offer options for movie-based and genre-based recommendations to address needs of the users.

1.4. Scope and Limitation

1.4.1 Scope

This project has many Scopes. They are as follows:

1. Personalized Recommendations: The system should use a content-based filtering approach to recommend movies based on user-selected features like genres and IMDb ratings or user selected movies.
2. Interactive Interface: The users can easily interact with the system, select their preferences, and view recommendations.
3. Enhanced Movie Details: The system should provide additional movie information, such as directors, cast, plot summaries, and posters, for better user experience.
4. Multiple Recommendation Modes: Users can be able choose between movie-based and genre-based recommendations, offering diverse features.
5. Dynamic Recommendations: The system should accept real-time input for features like the number of recommendations, preferred genres, and IMDb rating levels, to provide dynamic and customizable results.

1.4.2 Limitations

1. The system relies heavily on the preloaded dataset of movies, which may not cover all movies available on streaming platforms.
2. The system does not consider collaborative filtering or hybrid approaches, limiting recommendations to feature similarity rather than collective user behavior.
3. The retrieval of additional movie information relies on the OMDB API, which may cause delays or fail if the API key is invalid, the API limit is reached, or the movie details are unavailable.
4. The system does not include advanced personalization features, such as learning from user interactions or using user ratings to fine-tune recommendations.
5. The system uses a fixed set of recommendation algorithms (KNN and content-based filtering) which doesn't adapt over time based on user behavior or preferences.

6. The recommendation system relies solely on the "plot keywords" column. If this column is sparse, inconsistent, or missing in some movies, the quality of recommendations will be poor.
7. Not all movies may have posters available through the OMDb API, leading to incomplete movie details in some cases.

1.5. Development Methodology

The development of this project is done using the Incremental Delivery Methodology, which divides the project into more manageable, increments. Every increment builds on the one before it, adding additional features and functionalities. Early feedback and iterative improvement are made possible by the system's incremental delivery. This approach supports in reaching milestones, doing phased functionality testing, and improving the system in response to user interactions.

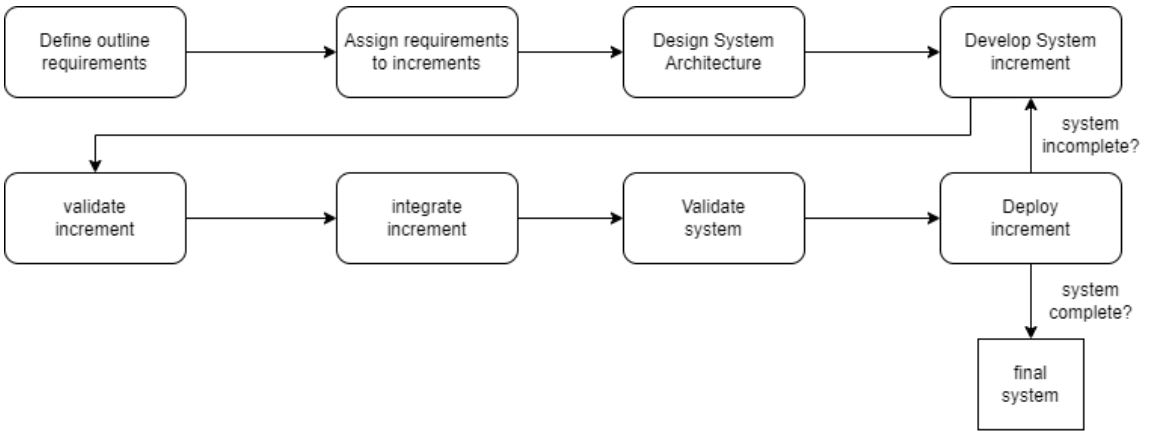


Figure 1. 1 Incremental Delivery Diagram

1.6. Report Organization

The report on "Movie Recommendation System" consists of six chapters. The project's constructive development is followed in each chapter. An outline of our project is provided in Chapter 1. It suggests the key ideas that will be covered in the ensuing chapters. Likewise, the theoretical literature review is typically found in chapter 2. It provides awareness into identifying potential hypotheses, approaches, and gaps in the current study. In order to assess, model, and develop information, Chapter 3 examines the system.

Additionally, it provides enough details for the study to be repeated. It discusses the goals of each experiment and tackles the issues raised in chapter 1. An overview of the system design and the algorithm employed in its development is found in Chapter 4. System testing, which covers the execution of a program or system with the goal of identifying errors, is covered in Chapter 5. It also covers the analysis of code and its execution in a variety of settings and circumstances. The importance of the "Movies Recommendation System" is explained in Chapter 6, along with potential future suggestions to improve the project.

Chapter 2: Background Study and Literature Review

2.1 Background Study

In recent years, the entertainment business has seen a significant shift toward digitalization, with a increasing number of people using online streaming services to watch movies. The huge number of movies available online can frequently lead to confusion with customers struggling to find a film that matches their preferences. This issue can be addressed by developing a Movie Recommender system that can make individualized recommendations to consumers based on their viewing history and preferences.

Movie recommendation systems have grown in popularity in recent years due to their capacity to deliver personalized recommendations to consumers, hence improving the user experience. These systems use machine learning algorithms and data analysis approaches to analyze user behavior and preferences and suggest movies that are most likely to be of user's interest. They consist of different factors such as movie genre, actors, ratings, and user feedback to provide recommendations.

The development of a Movie Recommender system requires skills in a variety of fields, including machine learning, data analysis, and software development. The system must be able to collect and analyze user data in real time, and generate personalized recommendations. Furthermore, the system must be capable of processing large amounts of data, as well as user-friendly and simple to browse.

A Movie Recommender system provides several benefits to both users and movie streaming businesses. Consumers will be able to easily find movies that match their preferences, resulting in an enjoyable viewing experience. Streaming services, on the other hand, will be able to provide customers with personalized recommendations, resulting in increased user engagement.

In conclusion, the development of a Movie Recommender system has the potential to change the entertainment business by offering personalized recommendations to consumers, hence improving the entire user experience. With the increase in demand for online streaming services, the development of such a system has become a necessity for movie streaming services to remain competitive in the market.

2.2 Literature Review

The entertainment industry has changed dramatically in recent years as a result of technological breakthroughs. As a result, several movie streaming services have arisen, providing consumers with a wide library of movies to choose from. However, with so many alternatives, it can be difficult to locate a film that matches one's tastes. To overcome this issue, movie recommendation systems have been created. According to the study by S. Rajarajeswari & colleagues [1], Recommender systems, often known as information filtering tools, employ big data to recommend things that the user loves based on their preferences and interests. Furthermore, they assist in matching people with similar tastes and interests. As a result, recommender systems now account for a significant portion of websites and e-commerce apps. suggestion systems are those that use suggestion algorithms such as collaborative filtering and content-based filtering. According to Rishabh Ahuja, Arun Solanki, and Anand Nayyar [2], Movie recommender system is built using the K-Means Clustering and K-Nearest Neighbor algorithms. The movielens dataset is taken from Kaggle dataset. The system is implemented in python programming language. The proposed work dealt with the introduction of various concepts related to machine learning and recommendation system. In that work, various tools, techniques and algorithms such as K-Means Clustering, KNN, Collaborative Filtering, Content-Based Filtering have been used to build recommender systems. According to a journal on Content based movie recommendation [3], the goal of a movie recommendation system is to suggest movies to different users based on their preferences. This allows the user to save time searching the internet for movies from the thousand already available options. A content-based recommendation system specifies the objects that can be recommended to the user. Based on a data set, it predicts what movies a user would enjoy based on the attributes found in previously liked films. Recommendation systems can suggest movies based on one or more attributes. Several elements are examined when building a movie recommendation system, such as the film's genre, director, and cast.

According to an article based on Content-based movie recommendation system [4], Recommendations are developed based on user similarities (Collaborative Filtering) or a user's individual behavior (Content-Based Filtering). A recommendation system is a data tool that supports consumers in selecting what they want from a large number of available objects. The recommendation algorithm recommends movies based on the user's choices. It supports the user in selecting the best option from the movies available. Recommendation

algorithms are used by popular platforms such as Netflix, YouTube, and Amazon to enhance consumer experience and profitability. Various sites offer recommendations, including movies, products, music, employment, and people. All of this is based on recommendation system. According to SRS Reddy & colleagues [5], Movie recommendation algorithms typically anticipate what movies a user would like based on the characteristics of previously liked movies. Such recommendation systems are useful for businesses that collect data from a big number of clients and want to efficiently deliver the finest recommendations available. Many elements can be considered when building a movie recommendation system, such as the film's genre, actors, and even the director. In conclusion, movie recommender systems have grown in popularity in recent years as a result of the enormous library of movies available on streaming platforms. To assure the system's success, engineers must address data security concerns, improve the user experience, and solve the cold-start issue.

Chapter 3: System Analysis

3.1 System Analysis

System analysis is the process of studying a system or organization to better understand its components, how they interact, and how they might be improved. It is a holistic approach that examines the system as a whole and determines the relationships between its components. The purpose of systems analysis is to identify problems and inefficiencies in the current system and provide solutions for improvement.

3.1.1 Requirement Analysis

In Requirement Analysis, we've analyzed and validated the requirements, recorded and monitored the implementation throughout the project.

i. Functional Requirements

1. Movie Recommendation: Based on the similarities between the genre if the movie users searched, the system must generate a list of recommended movies of similar genre for each user.
2. Feature Extraction: The system will be able to extract important features from the movies (like genres, actors, and directors) for the user during recommendation.
3. Scalability: The system can be able to handle a growing number of users in efficient manner.
4. User Feedback Collection: The system should allow users to give feedback about the system, to improve it in future.

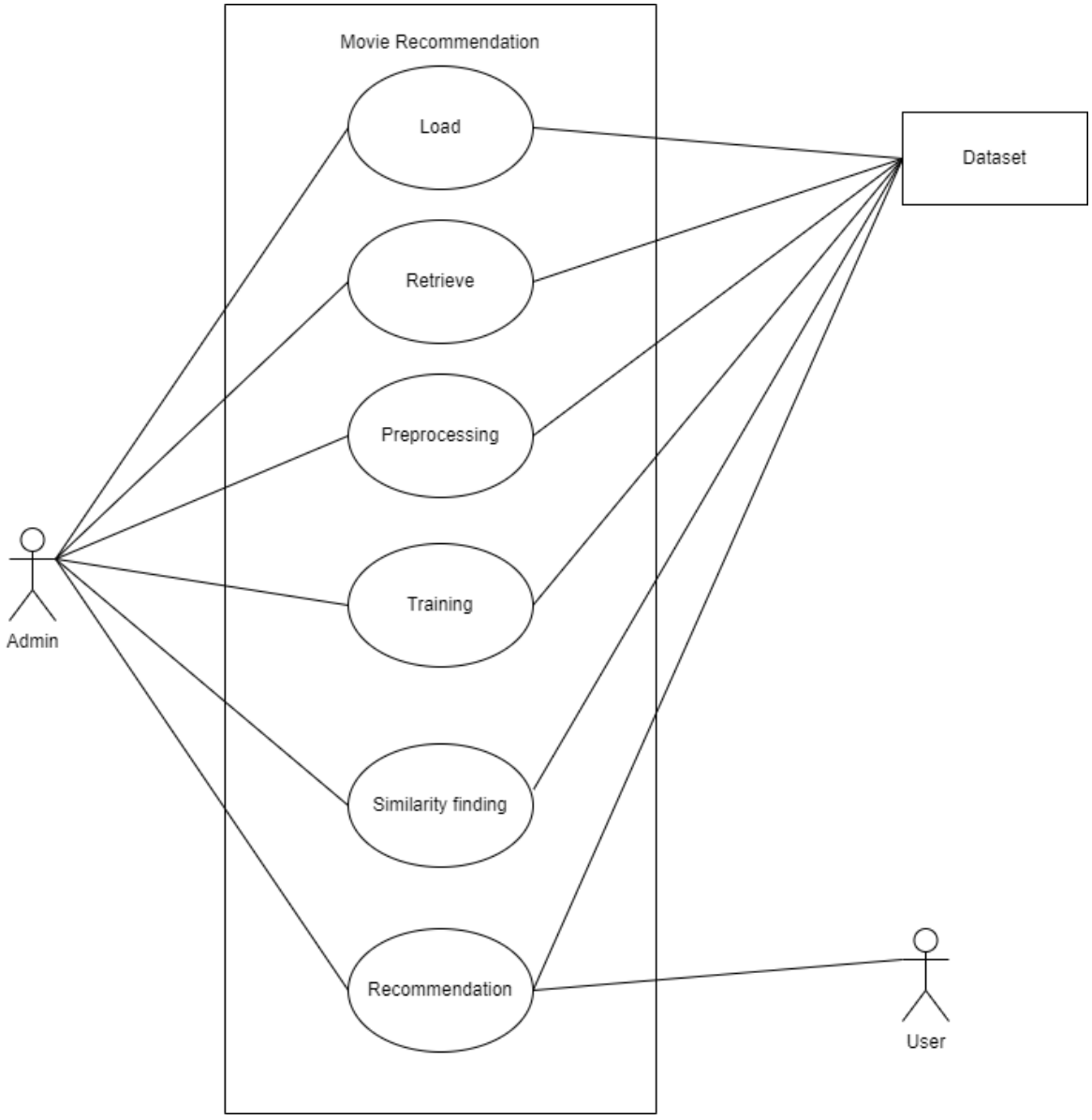


Figure 3. 1 Use Case Diagram

ii. Non-Functional Requirements

The points below focus on the non-functional requirement of the system proposed.

1. User friendly: User friendly generally means easy to use. The system is not complex. It is easy to locate different tools and options in our system.
2. Reliability: The system is reliable. The system takes data from a dataset.
3. Easy access: Our project is a web-based application. So, it can be accessed by anyone, through internet connection.

3.1.2 Feasibility Analysis

Feasibility studies aim to objectively and rationally detect the strengths and weakness of an existing or proposed system, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for the success.

3.1.2.1 Technical Feasibility

The technical feasibility of the project is high, as the required hardware and software resources are widely available and accessible. Requirements of our system can be categorized as:

Hardware Requirements:

- Processor: Multi-core CPU (e.g., Intel i5).
- Memory: 8GB RAM for handling large datasets and complex computations.
- Storage: SSD
- Reliable Internet Connection for data fetching, software updates, and cloud service access.
- Cloud backup services to safeguard data.

Software Requirements:

- Operating System: Windows 10, Linux, or MacOS
- Text Editor (VS-code)
- HTML, CSS, JS, PHP
- Python programming language
- Streamlit: For building the web interface
- Requests: For making HTTP requests to the OMDB API.
- Pillow: For handling and displaying images.
- NumPy: For numerical computations, specifically for the Euclidean distance calculation.
- Pandas: For data manipulation, reading CSV/JSON files, and processing the dataset.
- Scikit-learn: For TF-IDF vectorization and cosine similarity in the content-based filtering system. And K-Nearest Neighbors (KNN) in the KNN-based recommender.

- JSON: For parsing and loading JSON files.
- OMDB API Key: Ensure you have an API key to fetch movie data.

3.1.2.2 Operational Feasibility

Operational feasibility measures how well a proposed system can solve the defined problem, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase. The system can be designed to be dependable, maintainable, usable, sustainable, and reasonably priced. Therefore, this system is operationally feasible.

3.1.2.3 Economic Feasibility

Economic feasibility analyses the project's costs and revenue in an effort to determine whether it is possible to complete or not. There will not be any necessary equipment to be bought. However, the project will require a domain, hosting, and, most likely, API, all of which can be purchased and configured with a suitable plan. Even if some features are added, there will be no cost because no additional equipment is required. This system is cost-effective because the team already has everything they need. Extensive databases can be maintained as the number of users of the app increases.

3.1.2.4 Schedule Feasibility

It is the most important for the completion of the project on time. The project that we are proposing will too be completed within time constraints.

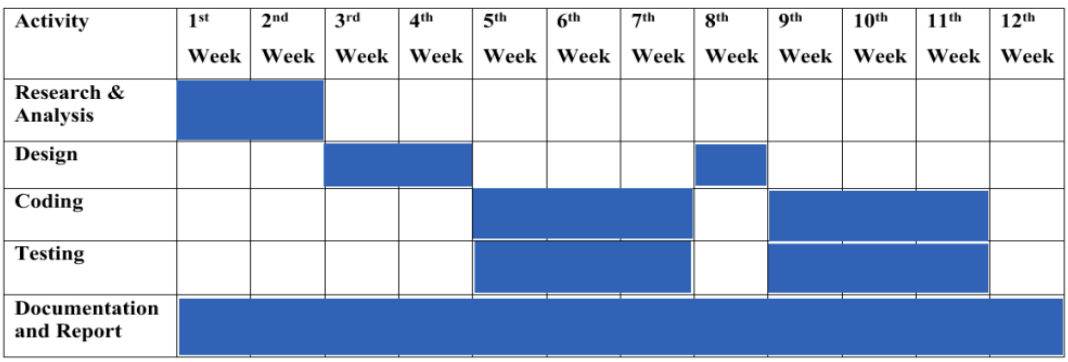


Figure 3. 2 Gantt Chart

3.1.3 Analysis

3.1.3.1 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. A class diagram is used not only to visualize, describe, and document various aspects of a system, but also to create executable code for the software application. A class diagram describes a class's attributes and operations, as well as the system's constraints. The purpose of a class diagram is to represent the static view of an application. Class diagrams are the only diagrams that can be directly mapped to object-oriented languages, so they are commonly used during construction. UML diagrams, such as activity diagrams and sequence diagrams, can only show the application's sequence flow; however, class diagrams are slightly different. It is the most widely used UML diagram in the coding community.

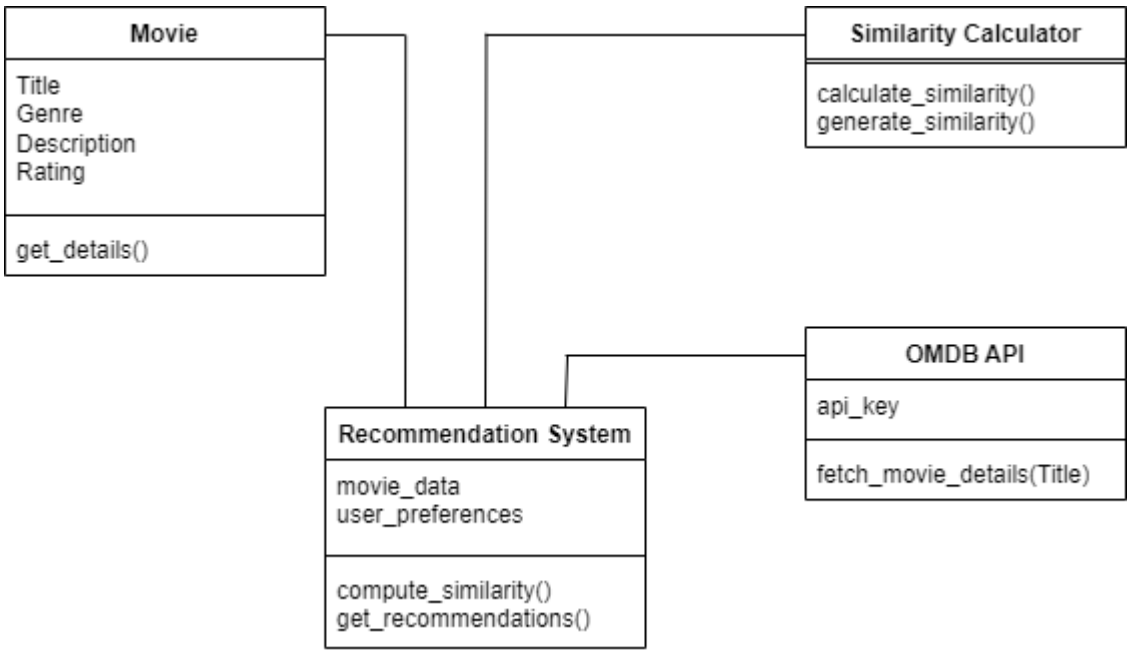


Figure 3. 3 Class Diagram of Movie Recommendation system

3.1.3.2 Sequence Diagram

Sequence diagrams can be valuable reference tools for businesses and other organizations. We use a sequence diagram for following reasons:

- To represent detailed information of UML use case.
- To represent the logic of a complex procedure, function, or operation.

- To observe how tasks are transferred between objects or process components.
- To plan and understand the detailed functionality of a system.

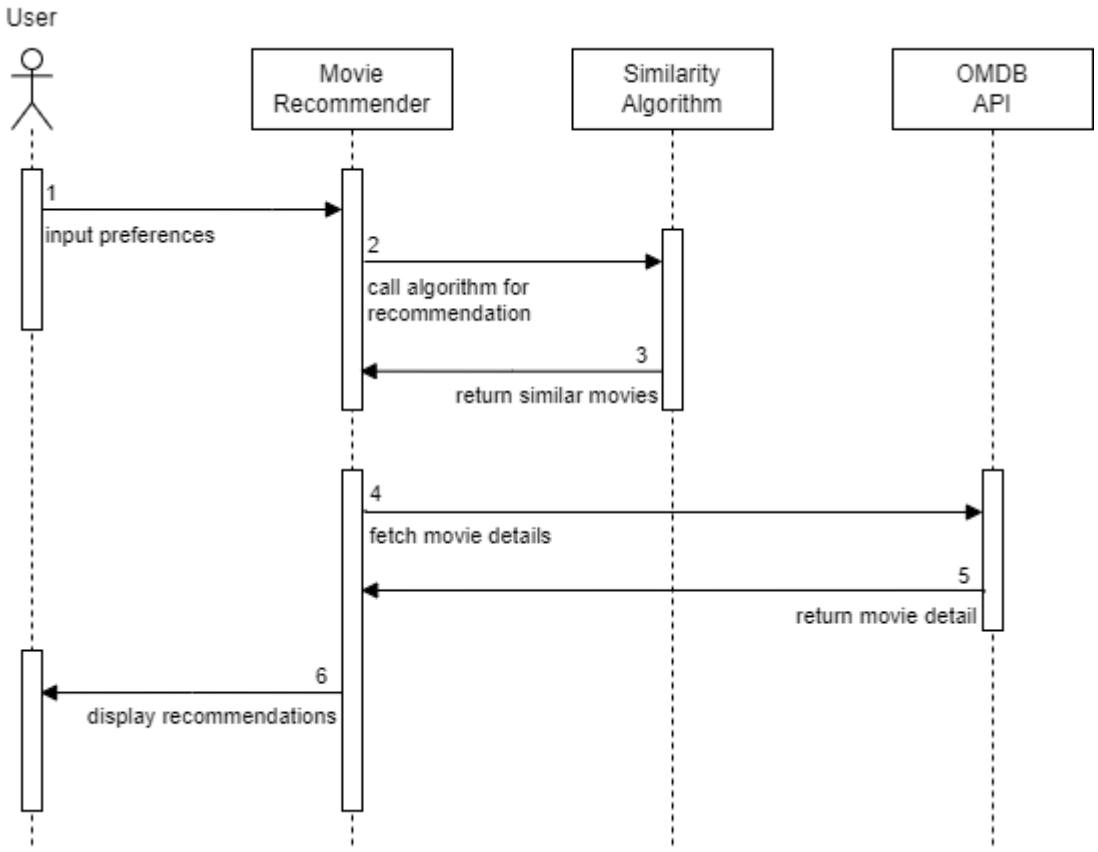


Figure 3. 4 Sequence Diagram of Movie Recommendation System

3.1.3.3 Activity diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. An activity diagram shows the steps involved in how a system works, helping us understand the flow of control. The activity can be considered a system operation. The control flow is transferred from one operation to another. This flow may be sequential, branched, or concurrent. Activity diagrams shows all types of flow control using various elements such as fork, join, and so on. The primary purpose of activity diagrams is to capture the dynamic behavior of the system.

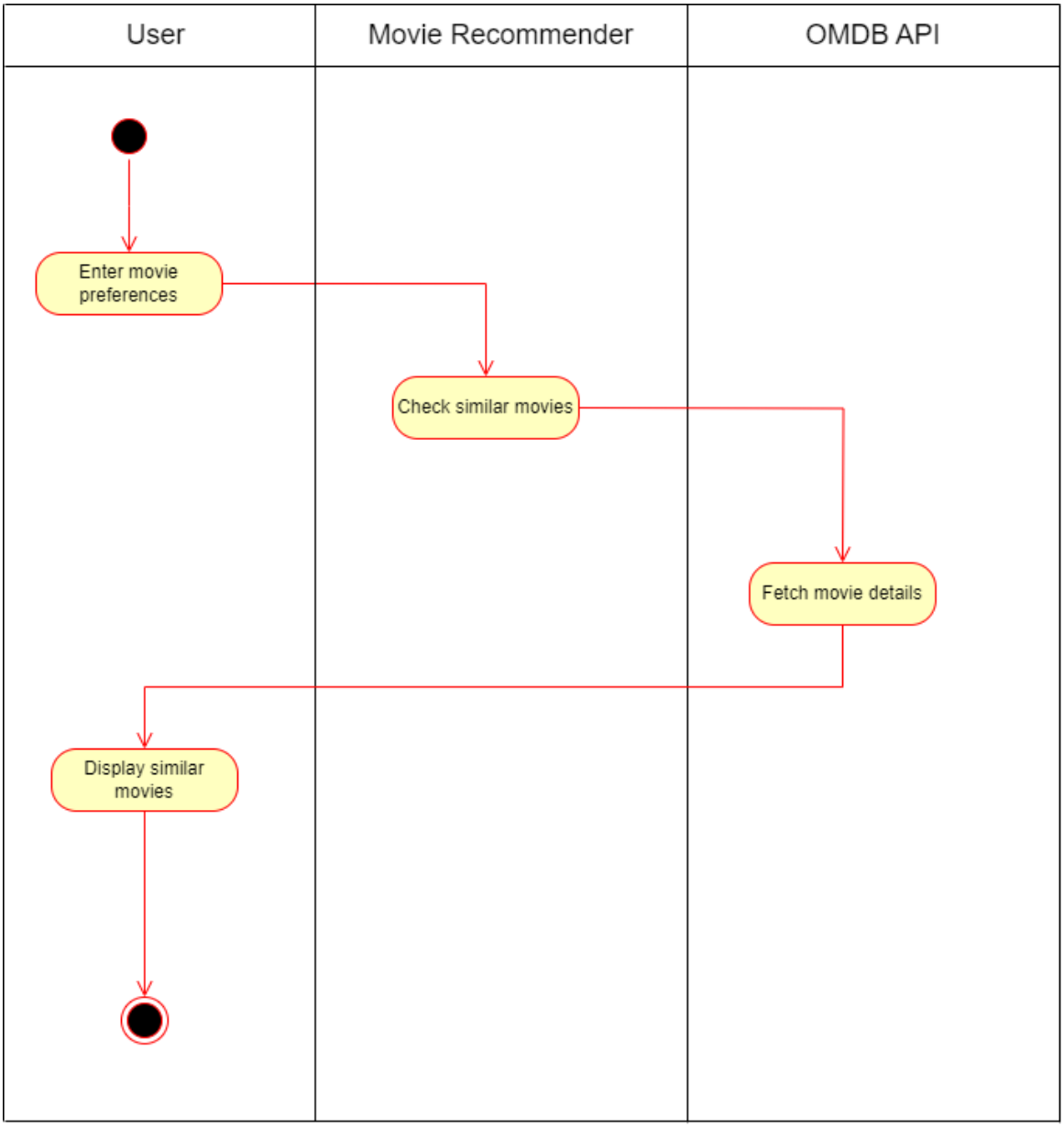


Figure 3. 5 Activity Diagram of Movie Recommendation System

Chapter 4: System Design

4.1 Design

System design is the process of representing architecture, interfaces, components that are included in the system. i.e., system design can be seen as the application of system theory to product development.

4.1.1 Refinement of Class Diagram

The UML diagrams have been refined to include a more detailed description of each system component, making it easier to understand how the system works overall. Refined UML diagrams include class, sequence, and activity diagrams for various system modules.

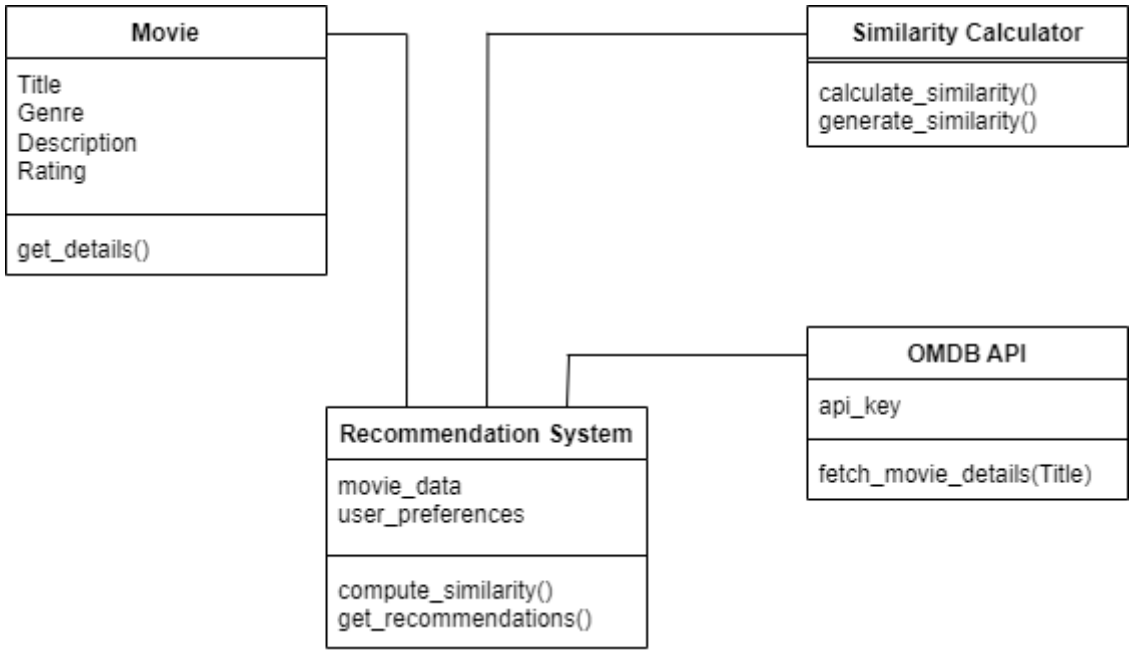


Figure 4. 1 Refined Class Diagram

4.1.2 Refinement of Sequence Diagram

A refined sequence diagram is a type of UML (Unified Modeling Language) diagram that depicts the interactions of objects in a system or process. It shows the sequence of messages sent between system objects or components, as well as the order in which these interactions take place.

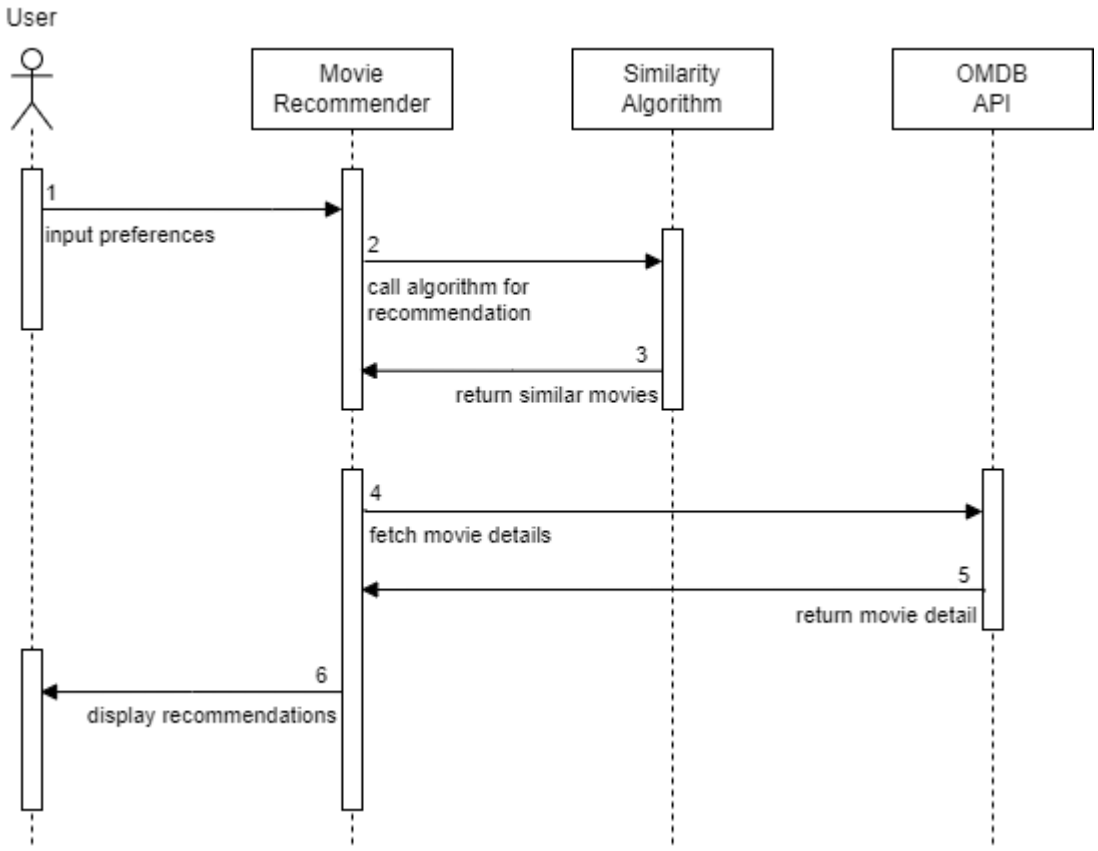


Figure 4. 2 Refined Sequence Diagram

4.1.3 Refinement of Activity Diagram

A refined activity diagram is a more detailed representation of the activities and actions involved in a specific process or workflow. It expands on the basic activity diagram by including additional details and steps, allowing for a more in-depth understanding of the process.

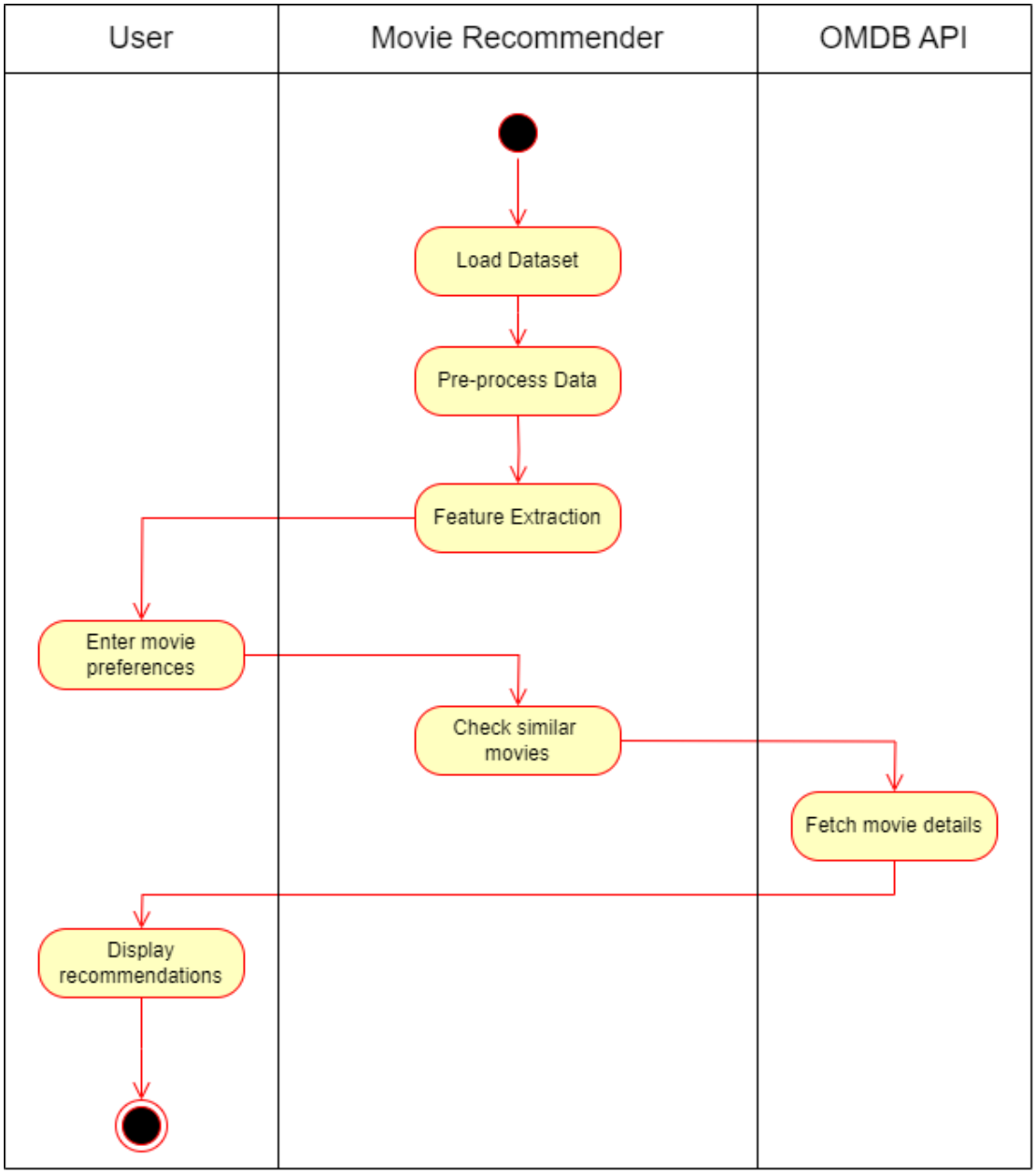


Figure 4. 3 Refined Activity Diagram

4.1.4 Component Diagram

A component diagram is a type of UML diagram that depicts the structural relationships and dependencies among the components of a software system. It represents how software components are linked to each other and interacts with each another within a system. Individual modules, libraries, executables, and other system parts can all be represented as components.

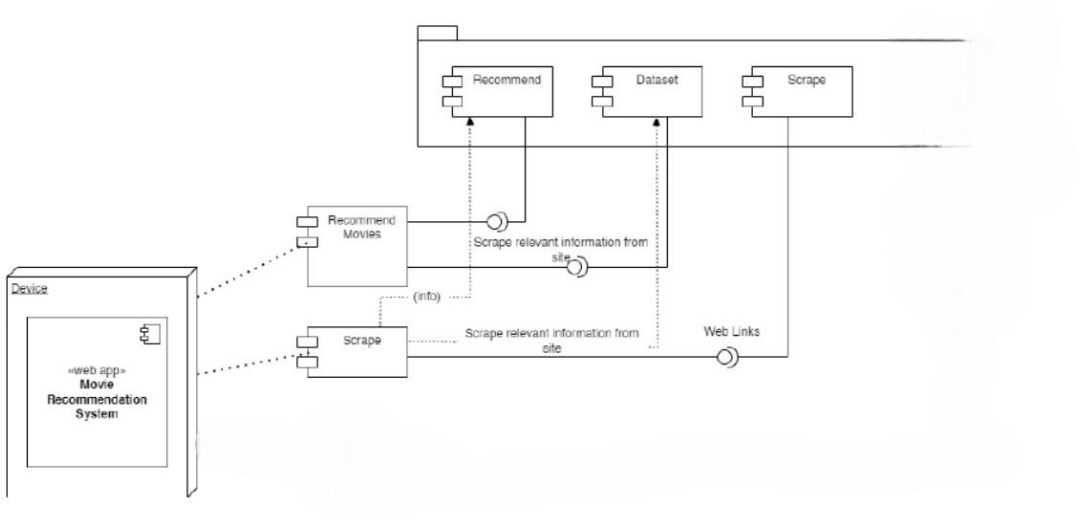


Figure 4. 4 Component Diagram

4.2 Algorithm Details

- **K-Nearest Neighbors (KNN) Algorithm:**

KNN is a distance-based algorithm that classifies data points based on their proximity to other data points. The core idea is that similar data points (e.g., movies with similar features) are likely to belong to the same class (e.g., genre). The algorithm works by calculating the distance between the test point (e.g., a selected movie) and all the other points (movies) in the dataset, and selecting the K nearest neighbors to make a classification.

Steps Involved in the Algorithm:

1. Input Data:

   The system receives movie feature data (data), where each movie is represented as a vector of features (e.g., genre, rating). The target represents the corresponding genre or category for each movie.

2. Test Point:

   The test point is the movie for which recommendations are to be generated. Its feature vector (e.g., genres, IMDb rating) is used to find similar movies.

3. Distance Calculation:

   Euclidean distance is used to measure the similarity between the test point and each point in the dataset. The Euclidean distance between two points p1 = (x1, x2, …, xn) and p2 = (y1, y2, …, yn) is calculated as:

$$distance(p1, p2) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

4. Finding K Nearest Neighbors:

   The distances from the test point to all data points are calculated. These distances are then sorted in ascending order, and the K nearest neighbors (i.e., the K data points with the smallest distances) are selected.

5. Classification:

   The categories (e.g., genres) of the K nearest neighbors are retrieved. The category that appears the most among the K neighbors is assigned to the test point as the predicted genre or category.

6. Output:

   The system outputs a set of recommended movies that belong to the most frequent genre among the nearest neighbors.

- **TF-IDF Algorithm:**

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical technique used to evaluate the importance of a word in a document relative to a collection of documents. It helps to represent the content of each movie (in the form of keywords or plot descriptions) as numerical vectors, where the importance of each word in the description is calculated. This allows us to compare movies and measure their similarity based on the textual content.

Steps Involved in the Algorithm:

1. Preprocessing the Text Data:

   It contains following steps:

- Tokenization: Each movie's textual features (like plot keywords, genres, director, and actors) are concatenated into a single string for each movie.
- Stop-word Removal: Common English words (like "the", "a", "and") are excluded from the analysis using the stop_words="english" parameter in TfidfVectorizer.
- Lowercasing: The text is automatically treated case-insensitively as the TfidfVectorizer inherently handles it.
- Stemming/Lemmatization: Not explicitly implemented in the code, but could be added if required to reduce words to their root forms.

2. Calculating Term Frequency (TF):

   Term Frequency (TF) for each word is computed as the frequency of a word within each movie's concatenated features string (which contains plot keywords, genres, director's name, and actors). This is handled implicitly by the TfidfVectorizer. It is calculated for each word in the document as:

$$TF(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

   Example: If a word "action" appears 3 times in a movie's plot description, and the total words in the description are 100, then:

$$TF(action) = \frac{3}{100} = 0.03$$

3. Calculating Inverse Document Frequency (IDF):

   IDF is calculated by TfidfVectorizer to assign a lower weight to words that appear frequently across the dataset (e.g., "action" or "comedy") and higher weight to less frequent, distinguishing words. It is calculated for each word as:

$$IDF(t) = log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ term\ t}\right)$$

   Example: If the word "action" appears in 50 out of 100 movie descriptions:

$$IDF(action) = log\left(\frac{100}{50}\right) = 0.3010$$

4. Calculating TF-IDF:

   TF-IDF is automatically computed by TfidfVectorizer for each word across all movies. The result is a matrix representing the relative importance of each word within the context of each movie's features (plot, genre, director, and actors).The formula is:

$$TF - IDF(t) = TF(t) \times IDF(t)$$

   Example: If a word "action" has a TF of 0.03 and an IDF of 0.3010, then:

$$TF - IDF(action) = 0.03 \times 0.3010 = 0.00903$$

   Words with higher TF-IDF values are considered more important in the document.

5. Creating TF-IDF Vectors

   The movies are represented as vectors of TF-IDF values in the form of a sparse matrix (tfidf_matrix), where each movie is associated with a vector that represents the weighted importance of its words. These vectors are used to represent the content of the movies numerically, which can later be compared using similarity metrics (like Cosine Similarity) to find similar movies.

- **Cosine-Similarity Algorithm:**

Cosine Similarity is a metric used to measure how similar two vectors (or texts) are by calculating the cosine of the angle between them. It is commonly used in text analysis and recommender systems to compare the similarity between two documents.

Cosine similarity ranges from -1 to 1, where:

- 1 indicates that the vectors are identical (i.e., very similar).
- 0 indicates that the vectors are orthogonal (i.e., completely dissimilar).
- -1 indicates that the vectors are diametrically opposite (i.e., completely opposite).

Cosine similarity is calculated using the dot product of two vectors and the magnitude (length) of each vector.

The formula for cosine similarity between two vectors is:

$$Cosine\ Similarity(A, B) = \frac{A \cdot B}{\|A\|\ \|B\|}$$

Where,

- A·B is the dot product of the two vectors.
- ‖A‖ is the magnitude (length) of vector A.
- ‖B‖ is the magnitude (length) of vector B.

Steps Involved in the Algorithm:

1. Convert Text into TF-IDF Vectors:
   The TfidfVectorizer is used to convert the concatenated features (plot, genre, director, actors) of each movie into TF-IDF vectors.

2. Compute Cosine Similarity Matrix:
   Cosine similarity between all movie vectors is computed using linear_kernel(), which measures the cosine of the angle between two vectors in a multi-dimensional space. The similarity between each pair of movies is stored in a matrix (cosine_sim).

3. Select the Movie and Find its Similarities
   The movie title passed through the URL query parameter (movie_param) is used to identify the corresponding index of the movie in the dataset. The cosine similarity scores between this movie and all others are retrieved.

4. Sort Similarity Scores and Select Top N Movies:
   The cosine similarity scores are sorted in descending order to find the most similar movies. The top N movies (based on the similarity score) are selected for recommendation.

5. Return and Display the Recommended Movies:
   The most similar movies are displayed as recommendations, and additional details (like IMDb link, poster, director, actors, plot, and IMDb rating) are fetched from the OMDB API using the fetch_movie_info_from_api() function.

Chapter 5: Implementation and Testing

5.1 Implementation Overview

5.1.1 Programming language Tools

For the Content-Based Movie Recommendation System project, the primary language used for the backend is Python, and for the frontend: HTML, CSS, PHP, JavaScript is used for the web interface.

Python libraries: For the computation and analysis we need certain python libraries which are used to perform analytics. Packages such as Streamlit, Numpy, Pillow(PIL), Requests, etc. are needed.

Streamlit: A Python framework for creating interactive web applications, used to design the user interface with widgets like dropdowns, sliders, and image displays, making the recommendation system user-friendly.

Scikit-learn: It is a machine learning library which is used for implementing algorithms in the recommendation systems. The algorithms are: Content-Based Filtering (TF-IDF & Cosine Similarity) and K- Nearest Neighbors (KNN).

Pandas: It is used to read, manipulate, and manage the movie dataset (such as movie_metadata.csv or JSON files). It allows the system to process, filter, and display movie data efficiently.

NumPy: NumPy is used for handling arrays and numerical operations, especially in calculating similarity scores between movies. It is critical for efficient mathematical calculations (e.g., cosine similarity).

Requests: A library for handling HTTP requests, utilized to fetch movie details, including posters and metadata, from the OMDb API.

Pillow (PIL): An image processing library, used to load and display movie posters retrieved from the API in the application.

JSON: A built-in Python library for reading and writing JSON data, used to manage movie datasets and API responses effectively.

Operator: A built-in Python utility library, used for sorting and indexing operations in the KNN algorithm, ensuring efficient data handling.

OMDb API: An external API service, integrated to provide detailed movie metadata, such as cast, director, plot, ratings, and posters, enriching the user experience.

5.1.2 Modules Description

This project implements following modules:

1. Recommendation Module: This module is used to recommend movies according to the user input (i.e. movie name or genres).
2. Scrape Module: This module is used to scrape movie information using OMDb API.

5.2 Testing

System testing examines every component of an application to make sure that they work as a complete and unite whole. Although each test serves a different purpose, they all aim to ensure that all system elements are properly integrated and perform their assigned functions. The testing process is actually carried out to check that if the product does operations exactly it is designed to do. In the testing stage, the following goals are attempted to be achieved:

- To affirm the quality of the project.
- To find and eliminate any residual errors from previous stages.
- To validate the software as a solution to the original problem.
- To provide operational reliability of the system.

Testing Methodologies

There are different types of testing methods and techniques used as part of the software testing methodology. Some of the key testing methodologies are:

5.2.1 Unit Testing

Unit testing is the first level of testing where you test the smallest functional unit of code. It is the process of ensuring that individual code of the software is functional and work as intended. In a test-driven environment, developers usually build and run tests before handing over the program or feature to the testing team. Unit testing can be done manually, however automating the process will take less amount of time and increase test coverage. Unit testing will also make debugging easier because identifying issues early means they will take less time to resolve. Pytest, PyUnit, Behave, etc are the python test frameworks used for unit testing. Test Left is also a solution that enables advanced testers and developers for the quickest test automation tool built into any IDE.

Table 5. 1 Test Cases for Interface

S.No.	Test Cases	Feature	Test Description	Step to Execute	Input Data	Expected Results	Remarks
1	TC-IV-1	User Registration and login	Verify that the user can register and then login.	• Register your details. • Login your detail.	Registration Data: Username: aashish Email: aashishbari@gmail.com Password: aashish Login Data: Email: aashishbari@gmail.com Password: aashish	User can be able to register their information and then can login.	Pass
2	TC-IV-2	Load application interface.	Verify that the application loads properly without any errors.	• Launch the application. • Wait for the interface to load.	Application: Movie Recommendation System	The interface should load with all UI elements	Pass
3	TC-IV-3	Movie-based recommendation	Verify that movie recommendation	• Open the application. • Select "Movie-	Recommendation Type: Movie-Based	The system displays movie recomme	Pass

		selection	s are displayed based on a selected movie.	based" from the recommendation dropdown. • Choose a movie from the movie list. • Set the number of recommended movies (e.g., 5). • Click "Submit" to generate recommendations.	Movie Selected: Inception Number of Recommendations: 5	ndations based on the selected movie with detailed information (director, cast, plot, etc.)	
4	TC-IV-4	Genre-based recommendation selection	Verify that genre-based recommendations are displayed based on selected genres.	• Open the application. • Select "Genre-based" from the recommendation dropdown. • Choose one or more genres from the genre selection list. • Set an IMDb rating threshold (e.g., 7).	Recommendation Type: Genre-Based Selected Genres: Action, Sci-Fi IMDb Rating: 7 Number of Recommendations: 5	The system displays movie recommendations based on the selected genres and IMDb rating.	Pass

				• Set the number of recommended movies (e.g., 5). • Click "Submit" to generate recommendations.			
5	TC-IV-5	recommend movies based on fetched data.	Verify that movie recommendations are displayed based on a fetched data.	• Open the application. • Click on a movie title that is recommended by the admin.	Fetched Movie: The Dark Knight	The system displays movie recommendations based on the clicked movie title with detailed information (director, cast, plot, etc.)	Pass

Table 5. 2 Test Cases for Model Accuracy

S.No.	Test Cases	Feature	Test Description	Steps to Execute	Input Data	Expected Results	Remarks
1	TC-MA-1	Valid movie name	Ensure that when a valid movie name is provided, the system correctly returns	• Provide the movie title "Avatar" to the system. • Check the recommended list of movies. • Manually verify if the	Movie Title: Avatar	Recommendations should closely match movies with similar themes, directors, or genres	Pass

			similar movie recommendations based on plot keywords.	recommended movies are related to "Avatar" in terms of genre, plot, or similarity		to "Avatar".	
2	TC-MA-2	Movie Not in the Dataset	Ensure that the system handles cases where a movie is not present in the dataset gracefully.	• Provide the movie title that is not in the dataset, to the system. • Check the response of the system.	Movie Title: Nonexistent Movie	The system should display a message like "No matching movie found".	Pass
3	TC-MA-3	No Movie Selected	Test the behavior when no movie is provided in the query parameter.	• Do not provide any movie name to the system. • Check the response of the system.	No Input	The system should prompt the user to input a movie name with a message like "No movie selected", indicating that a query parameter is needed.	Pass

Table 5. 3 Test Cases for Performance

S.No.	Test Cases	Feature	Test Description	Steps to Execute	Input Data	Expected Results	Remarks
1	TC-P-1	Response Time for a Movie Recommendation	Test the time taken to provide recommendations for a movie query.	• Provide a valid movie name (e.g., "Inception"). • Measure the time it takes to return movie recommendations	Movie Title: Iron Man	The response time should be as soon as possible.	Pass
2	TC-P-2	Load Time for Movie Info from OMDB API	Test the time taken to fetch movie details (like director, cast, etc.) from the OMDB API.	• Provide a valid movie name. • Measure the time taken for fetching movie details from the OMDB API.	Movie Title: Iron Man	The response time for OMDB API requests should be as soon as possible.	Pass
3	TC-P-3	Movie Not Found in Dataset	Test the behavior when the queried movie is not present in the dataset	Provide a movie name that is not present in the dataset.	Movie Title: Non-Existent movie	The system should display an appropriate message like "Movie not found" and handle the error gracefully.	Pass

5.2.2 System Testing

System testing is a black box testing method used to evaluate the completed and integrated system, as a whole, to ensure it meets specified requirements. The functionality of the software is tested from end-to-end and is typically conducted by a separate testing team than the development team before the product is pushed into production.

Table 5. 4 Test Cases for Web Scraping

S.No.	Test Cases	Feature	Test Description	Steps to Execute	Input Data	Expected Results	Remarks
1	TC-WS-1	Information Scrapping	Extract the relevant information about movies from OMDb Api.	Check the contents	Movie Title: Iron Man	Scrapped the relevant and needed information from the Api.	Pass

5.3 Result Analysis

The system was tested using unit testing and found to be effective in carrying out its intended functions. The results demonstrated that the project was able to accomplish its goals, although there is still potential for development in terms of expanding the system's capabilities and improving community participation.

5.3.1 Evaluating Accuracy

In machine learning, accuracy is a standard parameter for measuring the effectiveness of a model. Accuracy is the fraction of correctly classified data in the dataset. To calculate accuracy, divide the dataset into two parts: training and test sets. The training set trains the model, while the test set evaluates its performance.

In classifier model the most common measure to evaluate accuracy are:

- Precision: Precision is the fraction of true positives among all the positive predictions made by the model. It measures how accurate the model is when predicting positive instances. The formula for precision is:
  Precision = True Positives / (True Positives + False Positives)
- Recall: Recall is the fraction of true positives among all the actual positive instances in the dataset. It measures how well the model is able to identify positive instances. The formula for recall is:

Recall = True Positives / (True Positives + False Negatives)

- F1 score: The F1 score is the harmonic mean of precision and recall. It provides a single score that balances the tradeoff between precision and recall. The F1 score ranges from 0 to 1, where a score of 1 represents perfect precision and recall, and 0 represents the worst performance. The formula for F1 score is:
  F1 score = 2 * (Precision * Recall) / (Precision + Recall)

For Movie-Based Recommendation (Getting input from the user):

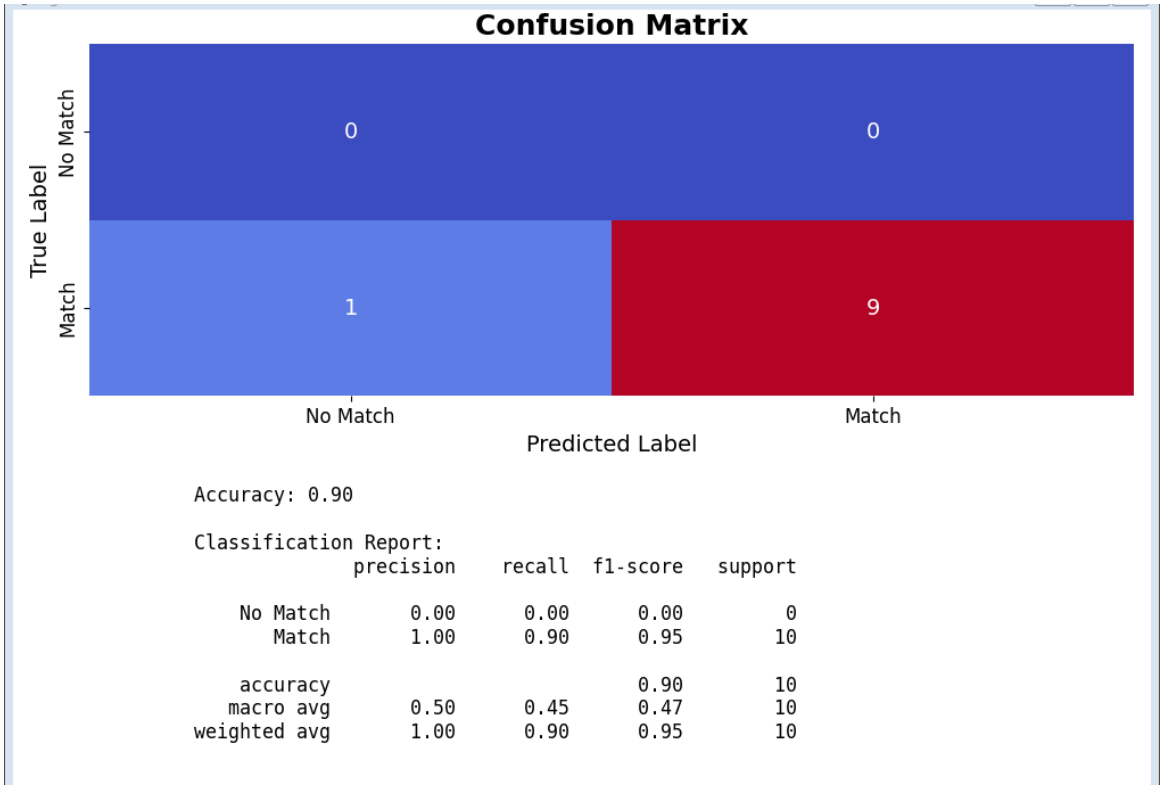
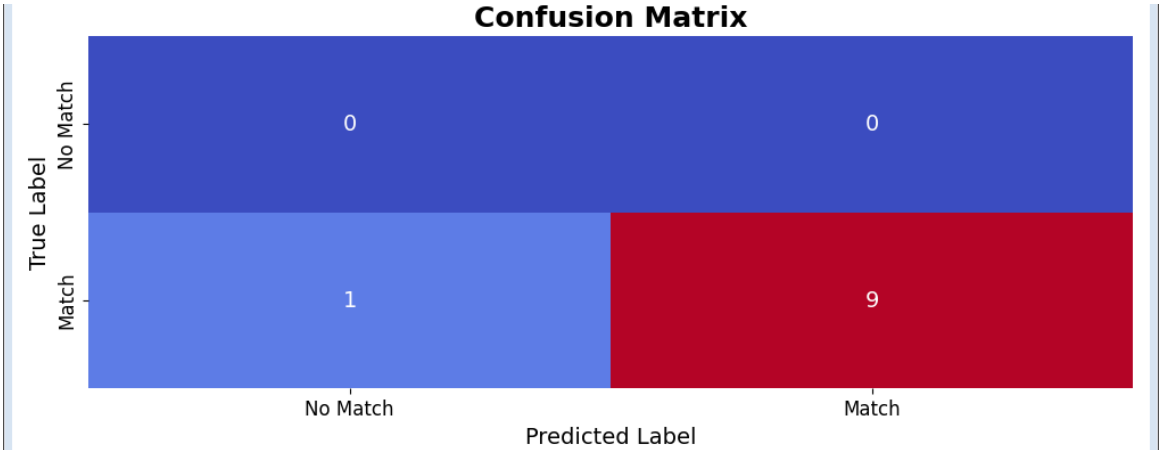


Figure 5. 1 Evaluating Accuracy for Movie-Based Recommendation (Getting input from the user)

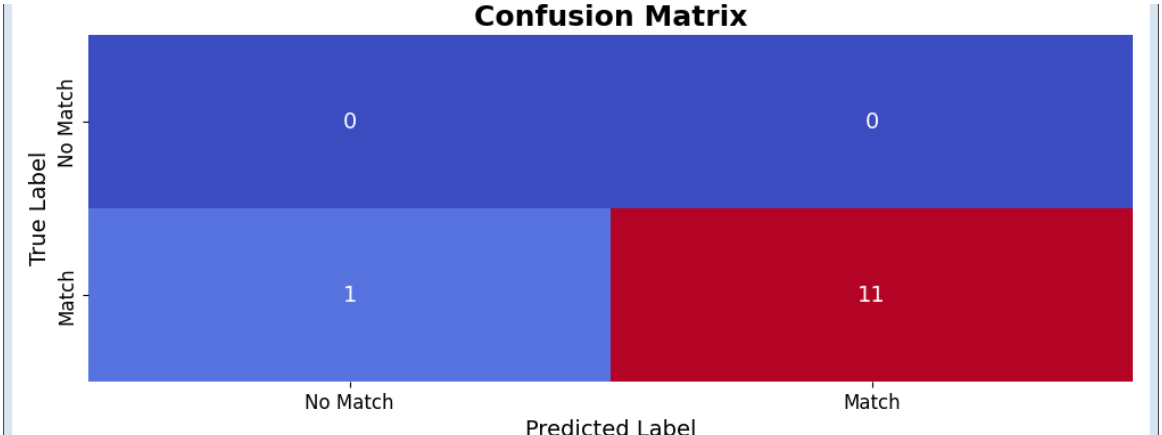
For Genre Based Recommendation:




```
Accuracy: 0.90

Classification Report:
              precision    recall  f1-score   support

    No Match       0.00      0.00      0.00         0
       Match       1.00      0.90      0.95        10

    accuracy                           0.90        10
   macro avg       0.50      0.45      0.47        10
weighted avg       1.00      0.90      0.95        10
```

Figure 5. 2 Evaluating Accuracy for Genre-Based Recommendation

For Movie-Based Recommendation (Fetching movie-name from the database):




```
Accuracy: 0.92

Classification Report:
              precision    recall  f1-score   support

    No Match       0.00      0.00      0.00         0
       Match       1.00      0.92      0.96        12

    accuracy                           0.92        12
   macro avg       0.50      0.46      0.48        12
weighted avg       1.00      0.92      0.96        12
```

Figure 5. 3 Evaluating Accuracy for Movie-Based Recommendation (Fetching movie-name from the database)

Chapter 6: Conclusion and Recommendations

6.1 Conclusion

The Movie Recommendation System successfully recommends movies based on user preferences. By combining movie data with the K-Nearest Neighbors (KNN) algorithm, TF-IDF algorithm and Cosine Similarity Algorithm, the system can recommend movies based on genre, IMDb rating, or a specific reference movie. The system obtains detailed movie information about the movie from the OMDb API, including posters, cast, and plot, for better user experience. The interactive interface, created using Streamlit, provides a seamless user experience, allowing users to easily input their preferences and explore movie options. Overall, the project demonstrates the value of combining data science techniques and web development tools to create an engaging and functional movie recommendation system.

6.2 Future Recommendations

In the future, we would like to improve the suggestion process by:

1. Display Streaming Availability: Integrating a simple feature that shows where users can stream the recommended movies (like Netflix, Amazon Prime, etc.) would be a helpful addition. This can be done by using APIs from streaming platforms or a movie database with streaming availability.
2. Advanced Recommendation Algorithms: Implement user-based or item-based collaborative filtering to recommend movies based on similar user behavior or movie ratings. Combine collaborative and content-based filtering to create hybrid recommendation models, increasing recommendation accuracy.
3. Improved Frontend/UI/UX: Add more filtering options like year of release, IMDB ratings, actor/actress, etc., to give users more control over the recommendations. Allow users to re-rank their movie suggestions based on different criteria.
4. Cloud Deployment and Maintenance: Consider moving parts of your project to serverless platforms to reduce infrastructure costs and improve scalability. Integrate automated testing to run every time a change is made, ensuring that your system remains reliable as it grows.

References

[1] S. N. S. S. M. K. S. P. &. P. U. S. Rajarajeswari, "Movie Recommendation System," 03 May 2019. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-5953-8_28. [Accessed 2024].

[2] R. Ahuja, A. Solanki and A. Nayyar, "Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor," July 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8776969/authors. [Accessed 2024].

[3] N. Pradeep, K. K. R. Mangalore, B. Rajpal, N. Prasad and R. Shastri, "International Journal of Research in Industrial Engineering," *Content based movie recommendation system*, vol. 9, no. 4, pp. 337-348, 2020.

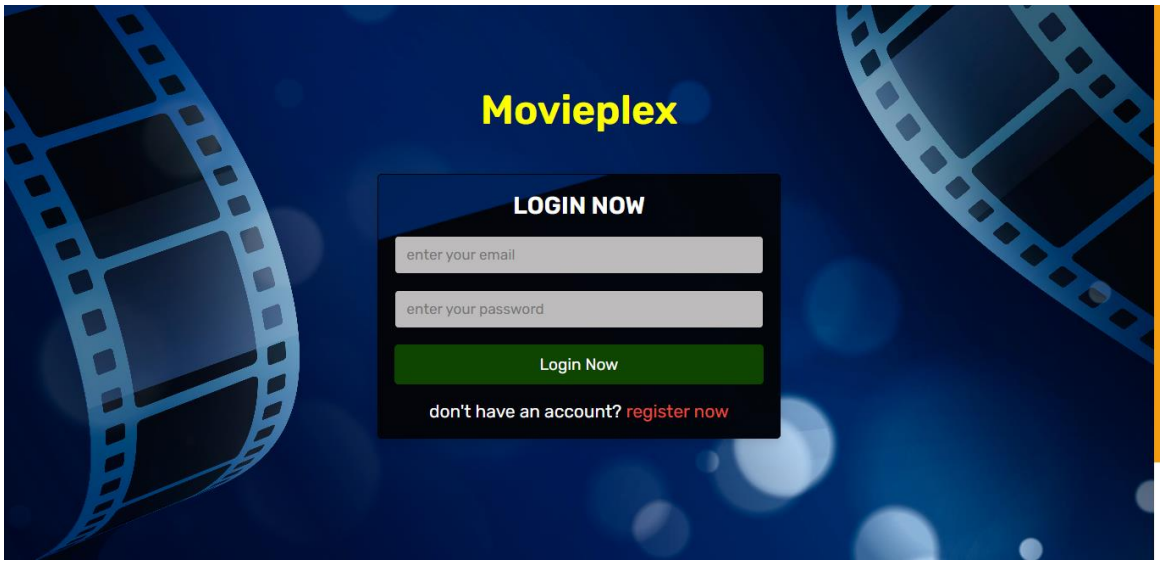
[4] D. D. G. C. P. L. S. Sridhar, "Content-Based recommendation System," *Content-Based Movie Recommendation System Using MBO with DBN*, no. 2022, pp. 1-2, 2022.

[5] S. N. S. K. S. A. &. B. V. SRS Reddy, "Content-Based Movie Recommendation System Using Genre Correlation," 5 November 2018. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-1927-3_42.

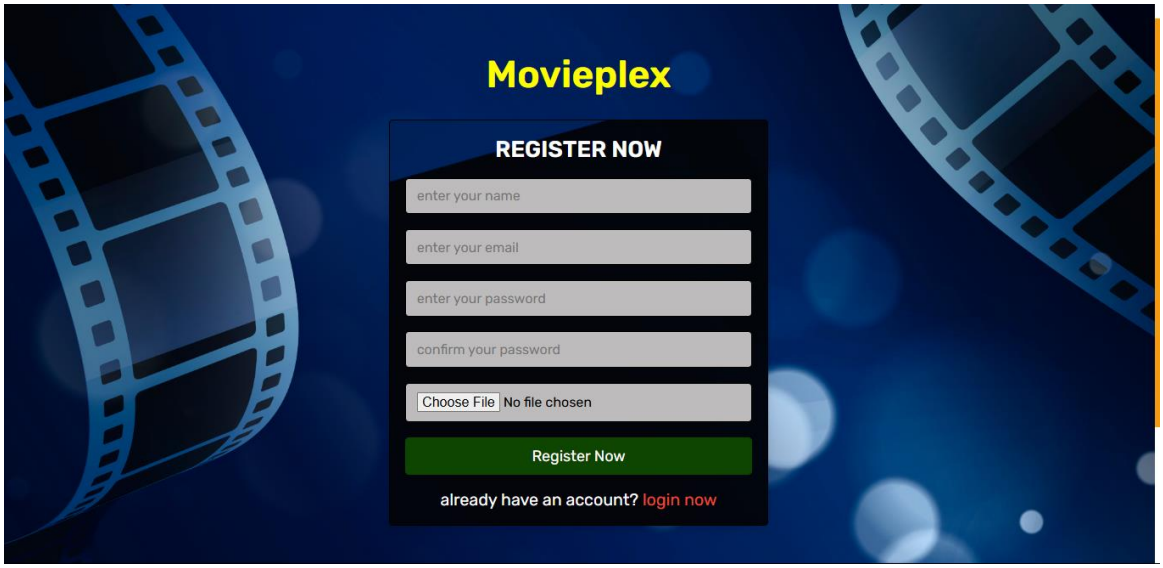
Appendices

(Screenshots)

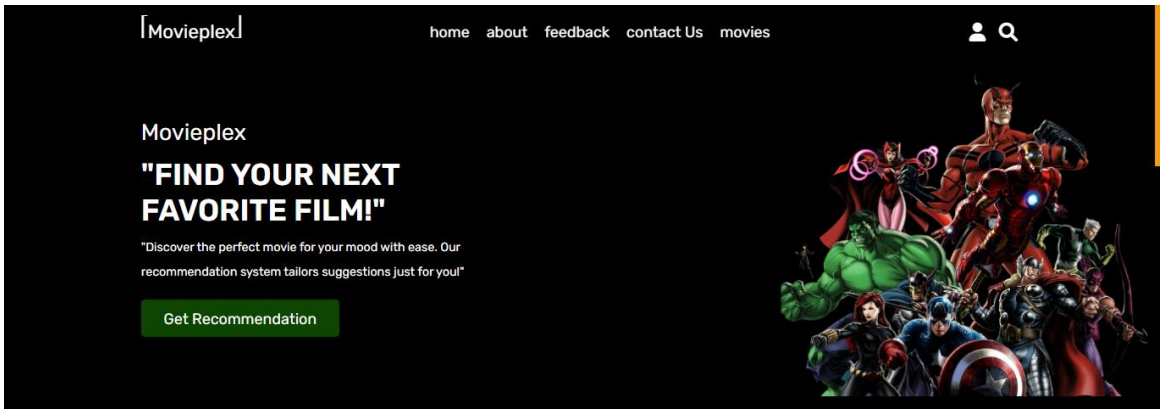
Login Page:

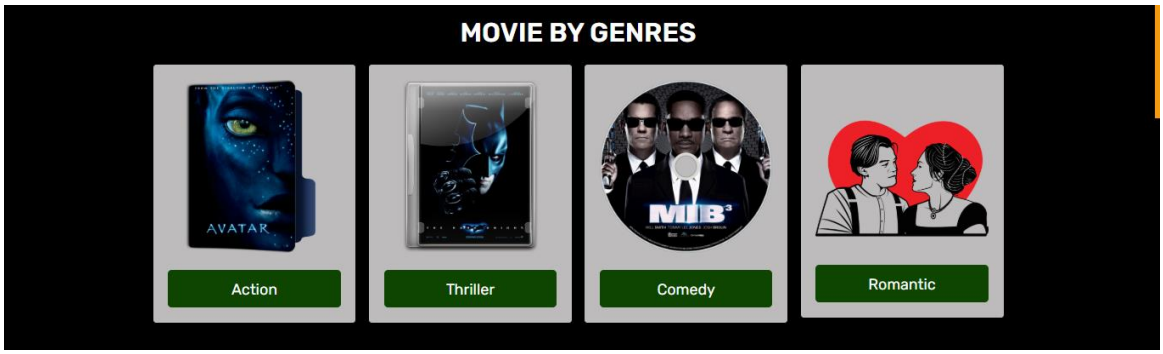


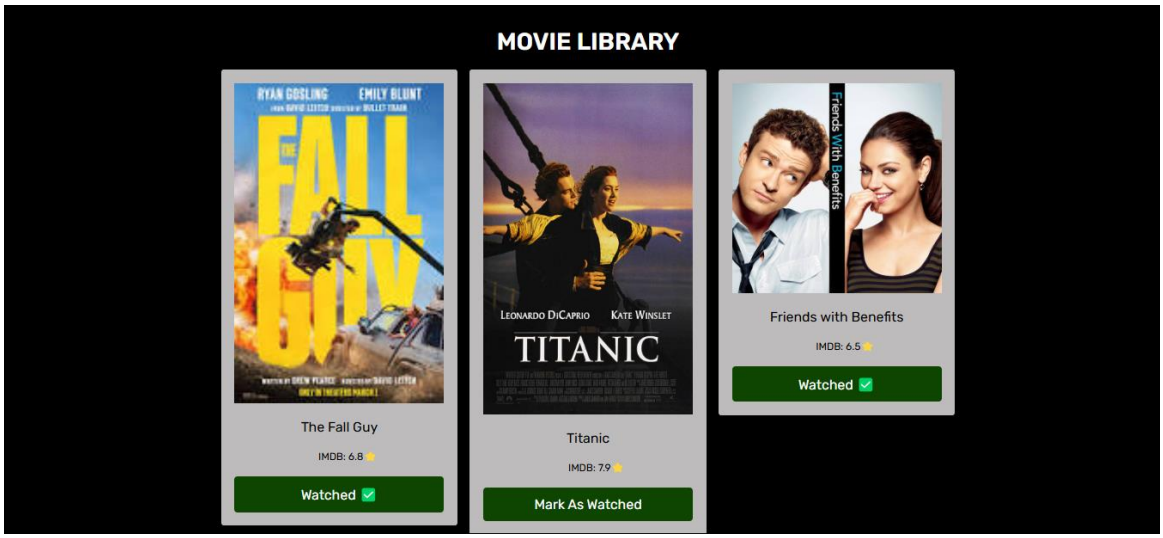
Registration Page:

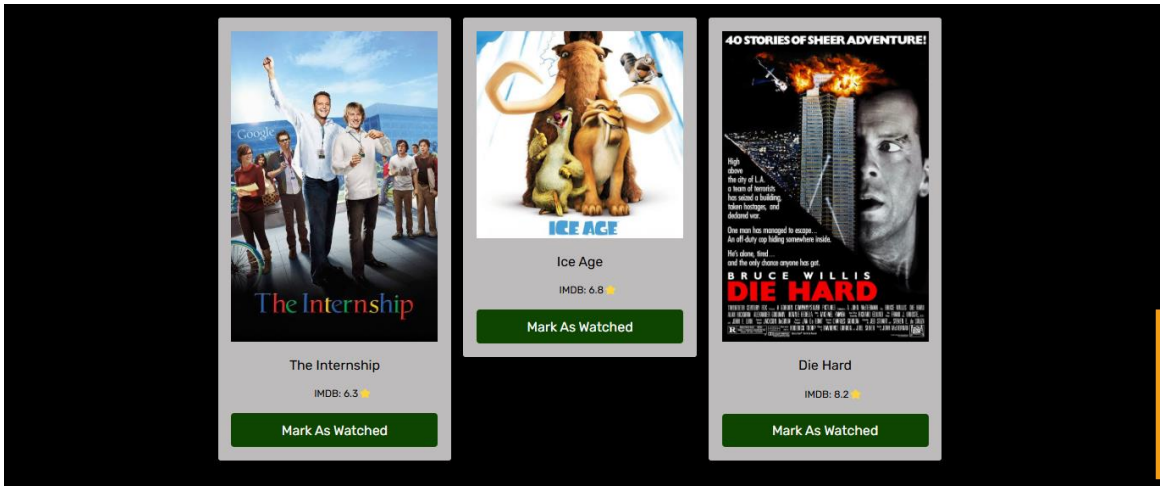


Home Page:

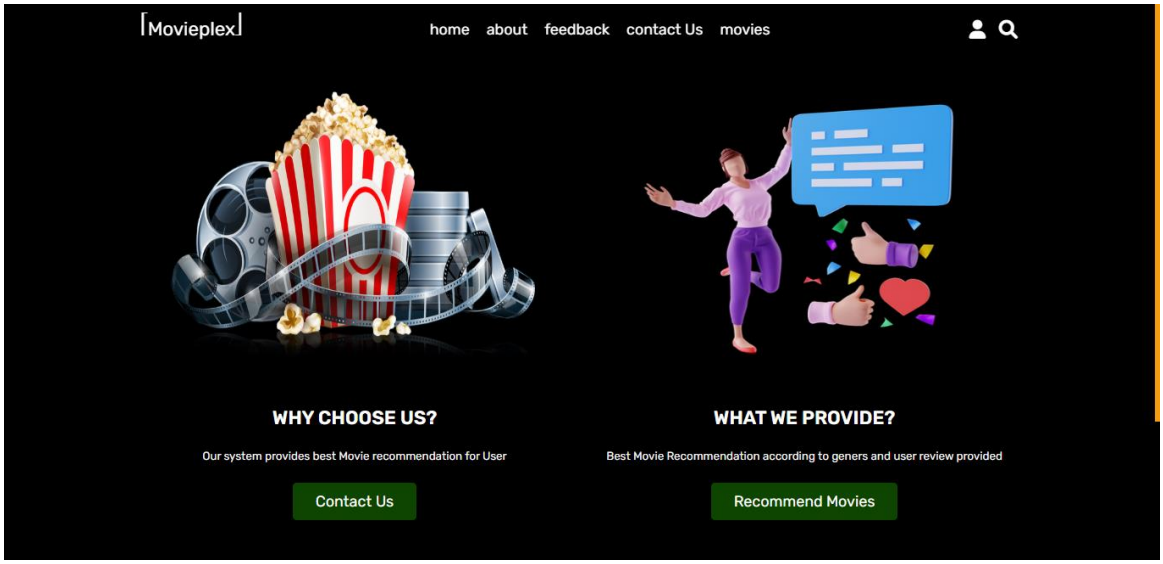

Movieplex
home
about
feedback
contact Us
movies
Movieplex
"FIND YOUR NEXT FAVORITE FILM!"
"Discover the perfect movie for your mood with ease. Our recommendation system tailors suggestions just for you!"
Get Recommendation


MOVIE BY GENRES
AVATAR
Action
Thriller
MIB
Comedy
Romantic


MOVIE LIBRARY
RYAN GOSLING
EMILY BLUNT
The Fall Guy
IMDB: 6.8
Watched
LEONARDO DiCAPRIO
KATE WINSLET
TITANIC
Titanic
IMDB: 7.9
Mark As Watched
Friends With Benefits
Friends with Benefits
IMDB: 6.5
Watched

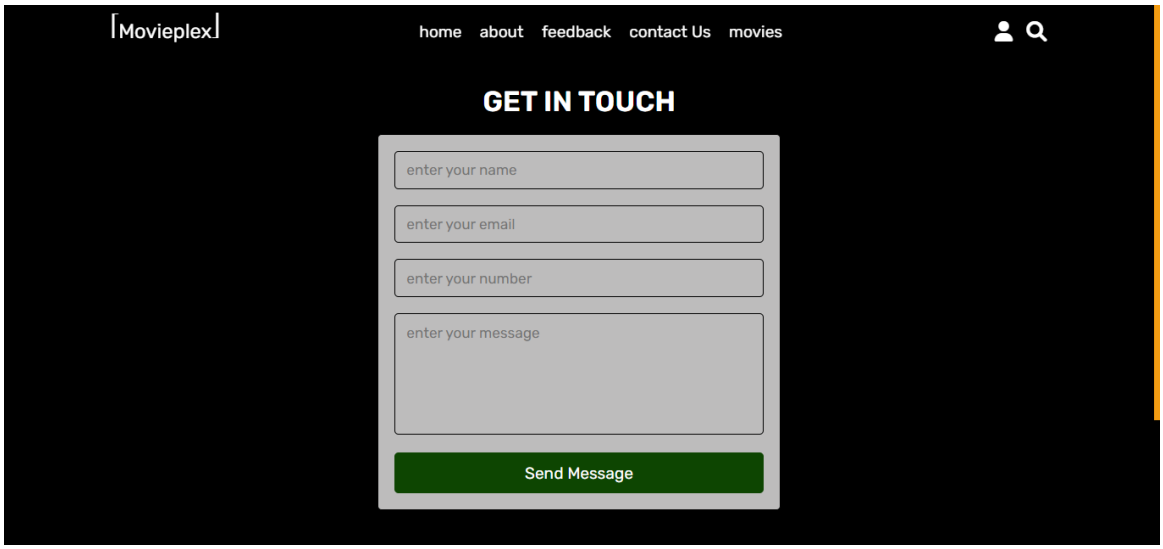

The Internship
The Internship
IMDB: 6.3
Mark As Watched
ICE AGE
Ice Age
IMDB: 6.8
Mark As Watched
40 STORIES OF SHEER ADVENTURE!
BRUCE WILLIS
DIE HARD
Die Hard
IMDB: 8.2
Mark As Watched

About Page:

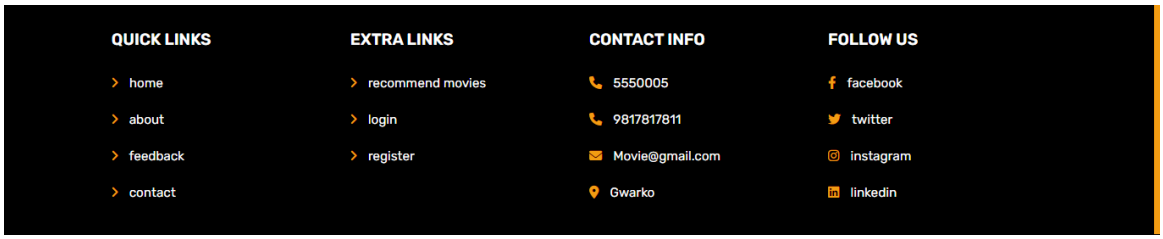

Movieplex
home
about
feedback
contact Us
movies
WHY CHOOSE US?
Our system provides best Movie recommendation for User
Contact Us
WHAT WE PROVIDE?
Best Movie Recommendation according to geners and user review provided
Recommend Movies

Website Feedback Page:

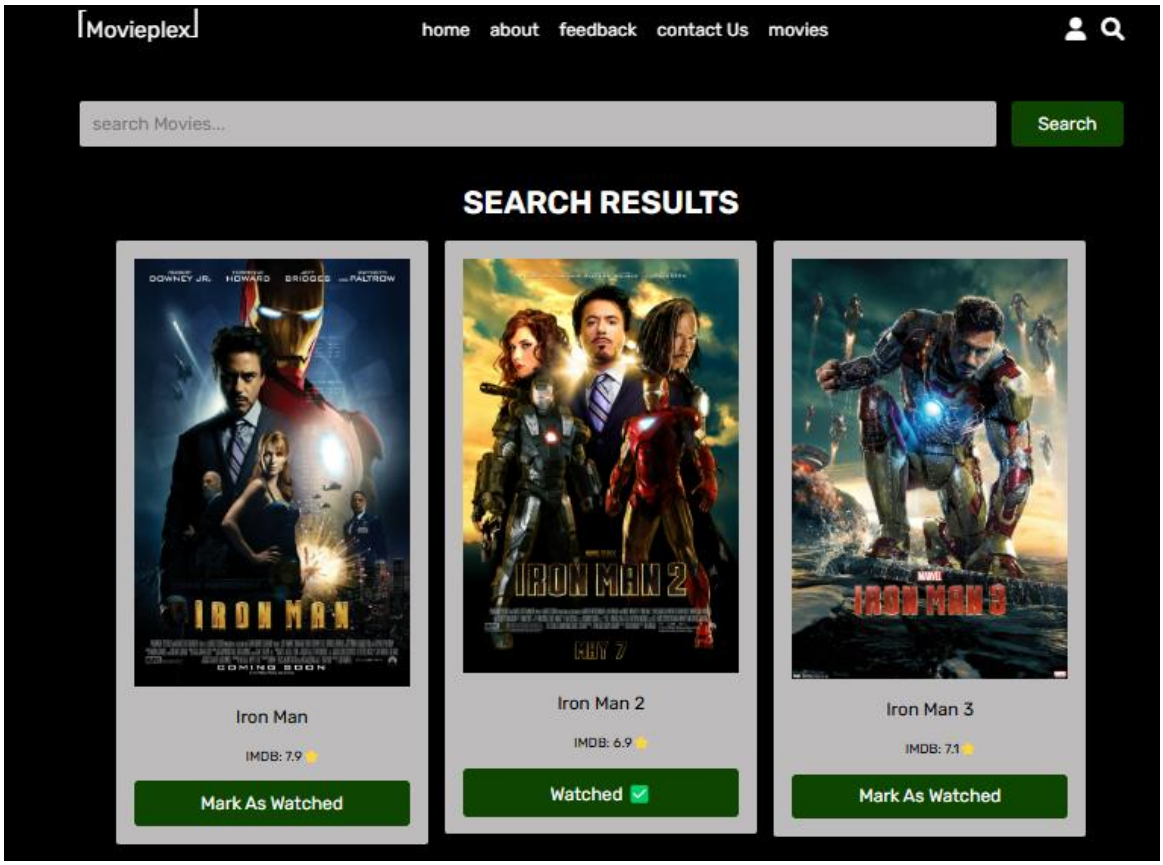
Contact Us Page:



Footer:

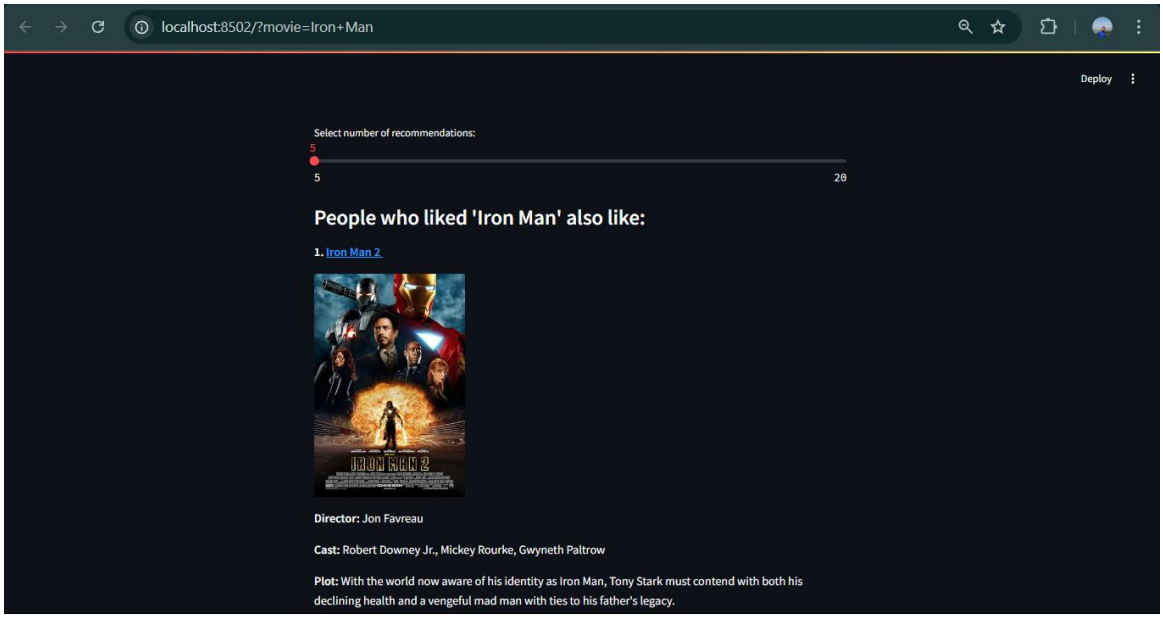


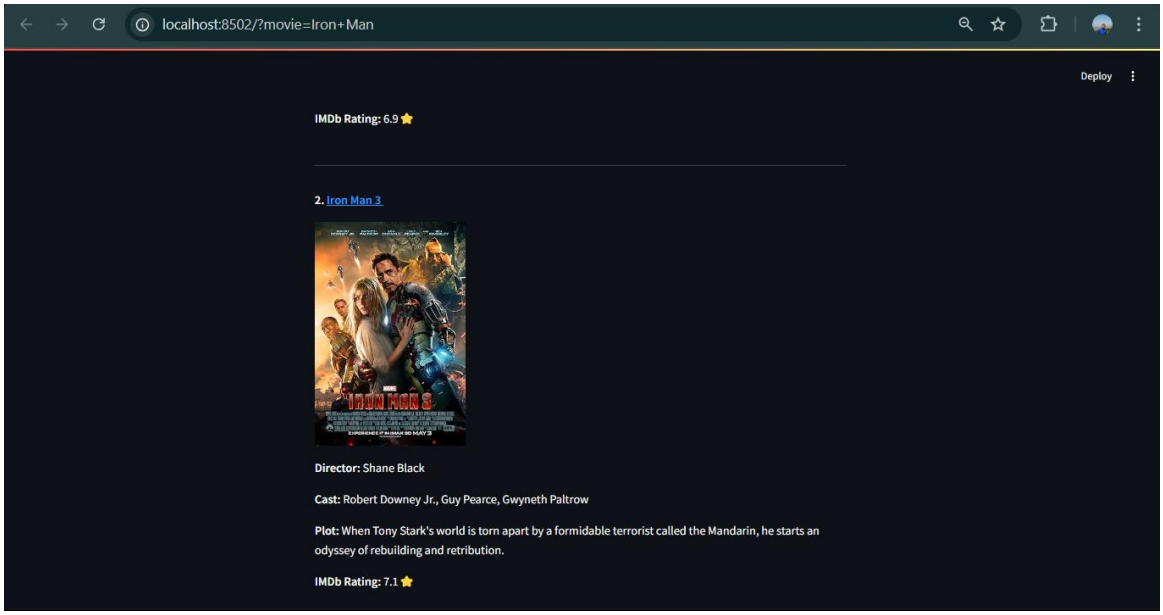
Search Page:



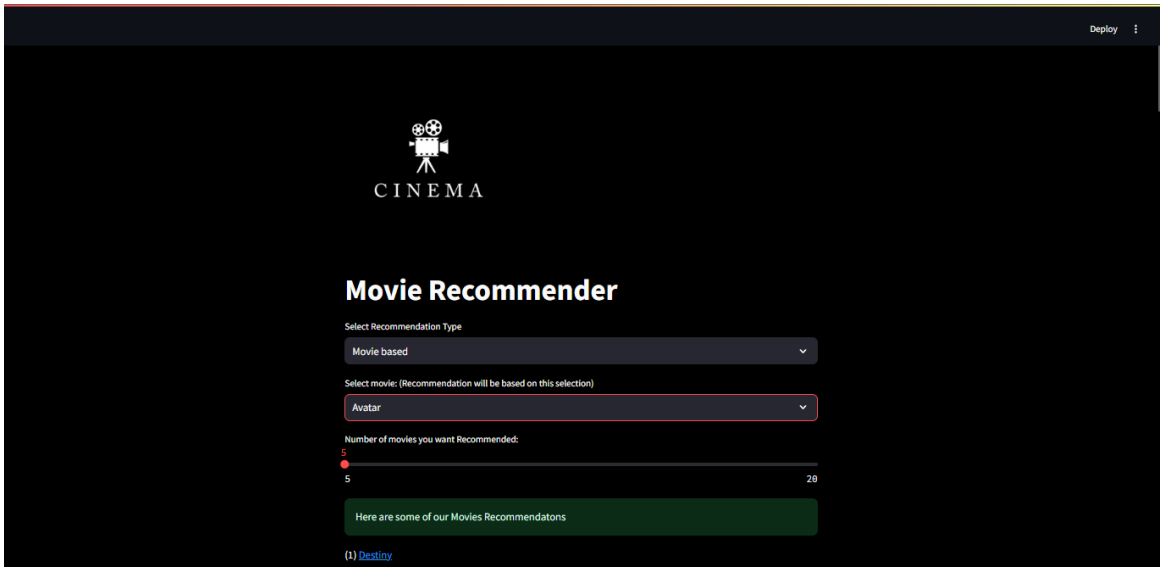
Recommendation Systems:

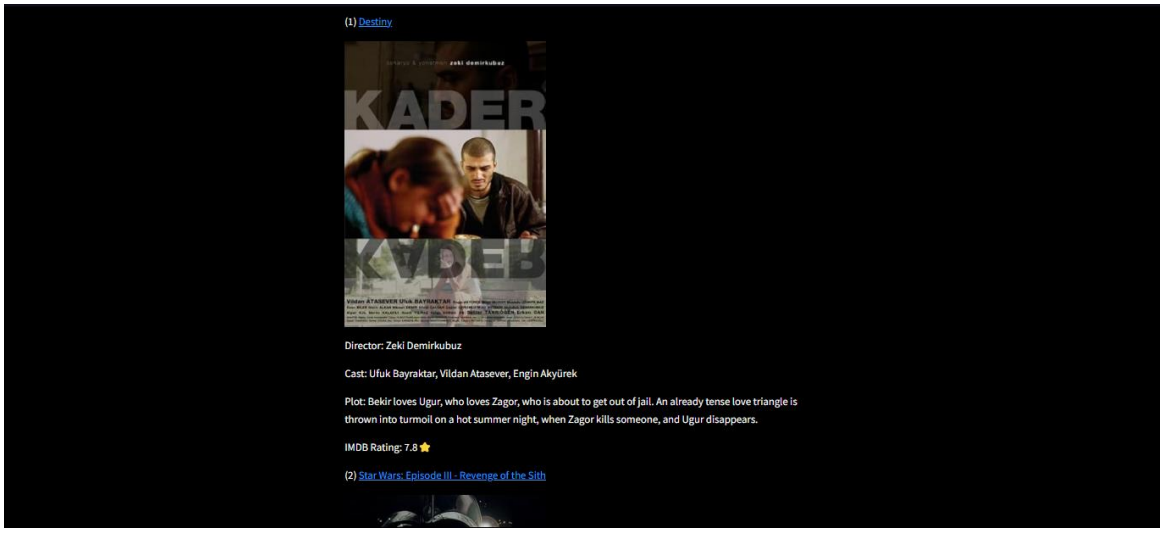
Recommending movies based on the movie you have watched:





Recommending movies based on movie name:





Recommending movies based on Genre: