**A Project Report on  
Movie Recommendation System based on User Preferences**

*Submitted to the*

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD**

*In partial fulfilment of the requirement for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**BY**

|  |  |
| --- | --- |
| ANAGONDI SHIVA PRASAD | (19WJ1A0519) |
| BADEMPET PAVAN | (19WJ1A0531) |
| CHEERALA PRASHANTH | (19WJ1A0561) |

Under the Esteemed Guidance Of

**Mr. K. Chandra Shekar**

**Associate Professor**

**Logo, company name

Description automatically generated**

**Department of Computer Science Engineering**

**GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMOUS)**

**School of Engineering & Technology**

**Ibrahimpatnam, R.R District- 501506**

**2022**



Date: 02-01-2023

**Department of Computer Science Engineering**

**CERTIFICATE**

This is to certify that this project report entitled **“Movie Recommendation System based on User Preferences”** by **Anagondi Shiva Prasad (19WJ1A0519), Badempet Pavan (19WJ1A0531), Cheerala Prashanth (19WJ1A0561)** was submitted in partial fulfilment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the **Jawaharlal Nehru Technological University Hyderabad** during the academic year 2022-2023, is a Bonafede record of work carried out under our guidance and supervision.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**INTERNAL GUIDE PROJECT CO-ORDINATOR HOD CSE**

**Mr. K. Chandrashekar Mr. B. Samirana Acharya Dr. J. Rajeshwar**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**EXTERNAL EXAMINER**



**ACKNOWLEDGEMENT**

We wish to express our sincere thanks to **Dr. Rishi Sayal, Associate Director, GNITC** for providing us with the conducive environment for carrying through our academic schedules and Project with ease.

We would like to say our sincere thanks to **Dr. J. Rajeshwar, Professor & Head, Department of CSE, GNITC** for providing seamless support and right suggestions are given in the development of the project.

We would like to say our sincere thanks to **Mr. B. Samirana Acharya, Assistant Professor, Department of CSE**, Project Coordinator, for providing seamless support and right suggestions are given in the development of the project.

We have been truly blessed to have a wonderful internal guide M**r. K. Chandrashekar, Associate Professor, Department of CSE**,for guiding us to explore the ramification of our work and we express our sincere gratitude towards him for leading me through the completion of Project.

Finally, we would like to thank our family members for their moral support and encouragement to achieve goals.

**ANAGONDI SHIVA PRASAD (19WJ1A0519)**

**BADEMPET PAVAN (19WJ1A0531)**

**CHEERALA PRASHANTH (19WJ1A0561)**

**ABSTRACT**

A Recommendation System is a filtering program whose primary goal is to predict the “rating” or “preference” of a user towards a domain-specific element. In our project, this domain-specific element is a movie. Hence the main focus of our recommendation system is to provide a total of ten movie recommendations to users who searched for a movie that they like. These results are based on tags of the movie that has been searched. Content based filtering is a technique that is used to recommend movies.

Apart from providing recommendations the system also provides posters, trailers/relevant videos of the Movies along with Release Date, Budget, Collection, Popularity, Similarity between selected Movie, Overview of Movie and More.

The System uses the concept of vectorization based on common features and uses Cosine Similarity with respect to each other vectors to determine the most similar movies.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVATIONS** | 1  4  5 - 7  8 |
| 1. | **CHAPTER 1: INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   1.3 EXISTING SYSTEM  1.3.2 LITERATURE SURVEY  1.4 PROPOSED SYSTEM | 9 - 17  9  10  11  12 - 16  17 |
| 2. | **CHAPTER 2: PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES  2.3 TECHNIQUE USED OR ALGORITHM USED  2.3.1 EXISTING TECHNIQUE  2.3.2 PROPOSED TECHNIQUE | 18 - 22  18  19 - 20  19 - 20  21 - 22  21  22 |
| 3. | **CHAPTER 3: REQUIREMENTS ENGINEERING**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS  3.4 FUNCTIONAL REQUIREMENTS  3.5 NON-FUNCTIONAL REQUIREMENTS | 23 - 26  23  23  24  25  26 |
| 4. | **CHAPTER 4: SYSTEM DESIGN**  4.1 general  4.2 UML DIAGRAMS  4.2.1 USE CASE diagram  4.2.2 CLASS DIAGRAM  4.2.3 OBJECT DIAGRAM  4.2.4 STATE DIAGRAM  4.2.5 ACTIVITY DIAGRAM  4.2.6 SEQUENCE DIAGRAM  4.2.7 COLLABORATION DIAGRAM  4.2.8 COMPONENT DIAGRAM  4.2.9 DEPLOYMENT DIAGRAM  4.3 DATA FLOW DIAGRAM  4.4 SYSTEM ARCHITECTURE | 27 - 38  27  28 - 36  28  29  30  31  32  33  34  35  36  37  38 |
| 5. | **CHAPTER 5: DEVELOPMENT TOOLS**  5.1 general | 39 - 41  39 - 41 |
| 6. | **CHAPTER 6: IMPLEMENTATION**  6.1 GENERAL | 42 - 58  42 - 58 |
| 7. | **CHAPTER 7: SNAPSHOTS**  7.1 SNAPSHOTS | 59 - 60  59 - 60 |
| 8. | **CHAPTER 8: SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING  8.3.1 UNIT TESTING  8.3.2 FUNCTIONAL TESTING  8.3.3 SYSTEM TESTING  8.3.4 PERFORMANCE TESTING  8.3.5 INTEGRATION TESTING  8.3.6 ACCEPTANCE TESTING  8.3.7 BUILD THE TEST PLAN  8.4 Test cases  8.4.1 TEst cases (Internal)  8.4.2 TEST CASES (EXTERNAL) | 61 - 63  61  61  61 - 63  61  62  62  62  62  63  63  64 - 65  64  65 |
| 9. | **CHAPTER 9: FUTURE ENHANCEMENT**  9.1 FUTURE ENHANCEMENTS | 66 - 66  66 - 66 |
| 10. | **CHAPTER 10: CONCLUSION AND REFERENCES**  10.1 CONCLUSION  10.2 REFERENCES | 67 - 69  67  68 - 69 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 4.2.1 | Use Case Diagram | 28 |
| 4.2.2 | Class Diagram | 29 |
| 4.2.3 | Object diagram | 30 |
| 4.2.4 | State diagram | 31 |
| 4.2.5 | Activity diagram | 32 |
| 4.2.6 | Sequence diagram | 33 |
| 4.2.7 | Collaboration Diagram | 34 |
| 4.2.8 | Component Diagram | 35 |
| 4.2.9 | Deployment Diagram | 36 |

**LIST OF SYMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single class. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is like another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processes. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard, sensors, etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | ML | Machine Learning |
| 2. | GUI | Graphical User Interface |
| 3. | NLP | Natural Language Processing |
| 4. | NLTK | Natural Language Toolkit |
| 5. | AI | Artificial Intelligence |
| 6. | DL | Deep Learning |

**CHAPTER 1**

**INTRODUCTION**

1

1.Introduct

**1.1 GENERAL**

A recommendation system is a type of suggesting system which makes suggestions based on the user’s liking. These systems can be applied to various data. These systems can retrieve, and filter data based on user's preferences to give suggestions or recommendations in the upcoming period. To watch a movie the first step is to select a movie that matches the user’s liking. Users often waste a lot of time selecting a movie to watch. Here comes the need for a recommendation system. It can recommend popular movies based on their rating, but what makes the system useful is its ability to recommend movies based on users’ liking and preferences. The purpose of this system is to search for content that would be interesting to an individual and provide user with those suggestions. Since the number of users and the movies are increasing day by day, computing the recommended movies list in a single node machine takes a very large time. When we deal with huge volumes of data coming from various sources and in a variety of formats as we see in the case of movies where there is a huge amount of data to be computed and then recommended to a user, it involves many aspects that must be taken into consideration while recommending movies to the user. Our recommending system uses cosine similarity which is a type of content-based filtering method to recommend similar movies to the user. Additional information about the searched movie will also be provided. Additional information is provided with posters of the Movies along with Release Date, Budget, Collection, Popularity, Similarity between selected Movie and an Overview as well.

Types of Recommendation Systems:

* Content-based Recommendation System
* Collaborative Filtering Based Recommendation System
* Hybrid Recommendation System

**1.2 OBJECTIVE**

To as to why or what is the need for developing? What kind of objectives that the system is to supposed to suffice? The two major categories of recommender approaches are collaborative and content-based filters (while Hybrid Recommendation System is a combination of two). Content-based filtering makes use of a set of definite and distinctive features of an item to recommend added items with identical characteristics, whereas, collaborative filtering denotes creating systems from client’s previous action, following that the system is implemented to predict results that the client may possibly be attracted to. Collaborative Filtering suffers when the user is new and hasn’t rated any movie yet, so we have no idea which kind of user is like this user this is what you call a cold start problem. While in content-based user just needs to input one movie or be asked to take a survey on the kind of movies they might like this could be based on known features like genres. It then works without a hassle and won’t require huge computing power and not even succumb to scalability or data sparsity.

* 1. **EXISTING SYSTEM:**
* Recommender systems are information altering tools that aspire to predict the rating for users and items, predominantly from big data to recommend their likes. Collaborative Movie recommendation Systems provide a mechanism to assist users in classifying users with similar interests. This makes recommender systems essentially a central part of websites and e-commerce applications.
* Existing System focuses on the movie recommendation systems whose primary objective is to suggest a recommender system through data clustering and computational intelligence. A novel recommender system has been used which makes use of k-means clustering.

**DRAWBACKS:**

* The slow rate of convergence.
* It has trouble clustering data where clusters are of varying sizes and density
* Centroids can be dragged by outliers, or outliers might get their own cluster instead of being ignored. Consider removing or clipping outliers before clustering.

**1.4 LITERATURE SURVEY**

**Title:** A Comprehensive Survey on Movie Recommendation Systems

**Author:** R. Lavanya, Utkarsh Singh, Vibhor Tyagi

**Year:** 2021

**Description:**

Internet technology has occupied an important part of human lives. Users often face the problem of the available excessive information. Recommendation system (RS) are deployed to help users cope up with the information explosion. RS is mostly used in digital entertainment, such as Netflix, prime video, and IMDB, and e-commerce portals such as Amazon, Flipkart, and eBay. The two traditional methods namely, collaborative filtering (CF) and content-based approaches consist of few limitations individually. Some fundamental issues faced by movie recommendation systems such as scalability, cold start problem, data sparsity and practical usage feedback and verification based on real implementation are still neglected. Other issues that require significant research attention are accuracy and time complexity problem, which could make RS, a bad candidate for real-world recommendation systems. This literature survey aims to consolidate and structurally categorize all the major drawbacks present in the most common and popular commercial movie recommendation systems.

**Title:** Career Recommendation Systems using Content based Filtering

**Author:** Tanya V Yadalam, Vaishnavi M Gowda

**Year:** 2020

**Description:**

Presently recommendation frameworks are utilized to take care of the issue of the overwhelming amount of information in every domain and enables the clients to concentrate on information that is significant to their area of interest. One domain where such recommender systems can play a significant role to help college graduates to fulfil their dreams by recommending a job based on their interest and skillset. Currently, there are a plethora of websites which provide heaps of information regarding employment opportunities, but this task is extremely tedious for students as they need to go through large amounts of information to find the ideal job. Simultaneously, existing job recommendation systems only take into consideration the domain in which the user is interested while ignoring their profile and skillset.

**Title:** Recommendation of Indian Cuisine Recipes based on Ingredients

**Author:** Nilesh Nilesh, Madhu Kumari, Pritom Hazarika, Vishal Raman

**Year:** 2019

**Description:**

There are lots of varieties of Indian cuisine available with same ingredients. In India, Traditional cuisines consist of wide varieties due to locally available spices, herbs, vegetables, and fruits. In this paper, we purposed a method that recommends recipes of Indian cuisine on the basis of available ingredients and liked cuisine. For this work, we did web scraping to make a collection of recipes' varieties and after that apply the content-based approach of machine learning to recommend the recipes. This system gives the recommendation of Indian Cuisines based on ingredients.

**Title:** The Cosine Similarity Technique for Removing the Redundancy Sample

**Author:** Worasak Rueangsirarak, Teeravisit Laohapensaeng, Suppakarn Chansareewittay, Anusorn Yodjaiphet

**Year:** 2019

**Description:**

The k-nearest neighbor algorithm is one of the basic and simple classification algorithms that share a common limitation of the algorithm which requires more computation cost when the size of training data is enlarged. To solve this problem, a new method applied to the cosine similarity for reducing the size of the training data set is proposed. This method reduces the data points that close to a decision boundary and retains the important points which affect classification accuracy. For the data far from the decision boundary and not affect the classification, these points will be removed from the training data set. The proposed method is evaluated its accuracy and reduction performance on the state of the art mechanisms, categorized as prototype selection algorithms. The 20 real-world data set are used to evaluate the proposed method.

**Title:** Recommendation system for property search using content based filtering method

**Author:** Tessy Badriyah, Sefryan Azvy, Wiratmoko Yuwono, Iwan Syarif

**Year:** 2018

**Description:**

Development of technology causes many business industries to migrate from offline business systems to the e-commerce world. One of the most popular e-commerce frequented by potential buyers is the property site. Considering that the property is one of the essential requirements for living, and furthermore it is also one of the most prized assets one can have. In this research, we develop a web-based recommendation system in choosing a property using content-based filtering method. The recommendation system provides property information based on user behavior by searching advertising content previously searched by the user. Each time the user selects the contents of the ad to display, this information will be stored into the database to be processed further in order to provide a recommendation.

* 1. **PROPOSED SYSTEM**
* The main purpose to develop a movies recommendation system is to provide users with recommendations that are not based on popularity or purely rating but based on the movies that the user likes. This will lead to a highly personalized recommendation, which will increase the accuracy of the recommendation system.
* This system will help in recommending movie to users based on Content using Cosine Similarity. Making it a more stable and faster model. This model will be giving filtered movies from a large set of movies. Upon recommendation of movies the user won’t have to surf the internet for finding a movie that he/she likes as all the information needed will be provided on a single platform. The user won’t have to rely on friends for a movie suggestion as the recommendation system will provide the user with the top ten movies that are most like the searched movie.

**ADVANTAGES: -**

* The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.
* When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.
* With cosine Cosine Similarity not being affected by Sparse Data it becomes a great tool for comparison between objects.
* It is fairly quicker and stable

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

The main purpose to develop a movies recommendation system is to provide users with recommendations that are not based on popularity or purely rating but based on the movies that the user likes. This will lead to a highly personalized recommendation, which will increase the accuracy of the recommendation system. The additional information of the searched movie will help the user make an informed decision while selecting a movie. The user won’t have to surf the internet for finding a movie that he/she likes as all the information needed will be provided on a single platform. The user won’t have to rely on friends for a movie suggestion as the recommendation system will provide the user with the Top 10 movies that are most like the searched movie.

**2.2 METHODOLOGIES**

**2.2.1 MODULES:**

**This project necessarily includes 6 Major Modules**

**1: Data Collection**

Appropriate data sets are shortlisted and downloaded from Kaggle. To keep the data up to date there is a constant revision of dataset with minimal to no changes to code. Additionally, TMDB (The Movie Database) API is used to fetch other data and movie posters. Two datasets were collected one constituted the Movies Information while the other was filled with the Credits.

**2: Data Analysis**

Analyzing the data is examining data to identify patterns and make sense of them in the context of your problem at the same time understanding the outliers, measure of central tendency like mean, median, mode. Analysis of data it’s size, rows, columns, null values etc. Similar components in two datasets.

**3: Data Preprocessing**

The movies data set and the credits dataset which were taken from Kaggle are processed in Jupyter Notebooks to clean the data. It included Removing irrelevant features from your dataset, handling missing data & duplicate data. Merging both the datasets. Performed necessary transformations.

**Step 4: Model Training**

Among many tags available the focus should be on those words with high frequency (not including stop words) but prior to that there is a need for necessary stemming to avoid using inflectional morphemes which might cause a bit of duplicate similarity. NLTK (Natural Language Toolkit) library can be used to perform various transformations on text. It includes a PorterStemmer class which enables stemming on inflectional morphemes to be counted as a same unit. Then the Count Vectorizer Class to make tags based on word frequency. Then Cosine Similarity is performed on those vectors to find the Cosine Angle to recommend the top ten movies which are like the searched movie.

**Step 5: Creating web app service**

For the system to be useful and easy to use, the GUI must be good. This would help the user to communicate with the software. For this to be accomplished we first made a pickle of the model and then used Streamlit for creating a web application.

**Step 6: Containerization & Deploying on Cloud**

Further to deploy the model on the internet such that it becomes readily available for real-time usage we used Docker to first create images and composed up into the Docker Registry for building containers then created a registry in the Azure Portal and pushed the latest then with the help of Azure App Service we were able to deploy it.

**2.3 TECHNIQUE USED OR ALGORITHM USED**

**2.3.1 EXISTING TECHNIQUE: -**

**k-means Clustering Algorithm**

* In the existing system K-means as clustering algorithm was used. Initially k-means clustering algorithm is applied to Movies dataset for clustering of users into different clusters. The clusters are selected randomly at ﬁrst then users are inspected one by one by calculating the differences in their ratings and the centric of the clusters, and if their difference is smallest, then the user gets allocated to the cluster to which they are closest. However, at this moment does not assure that each user has been assigned to the real cluster with a minimum difference of centric. So, each user’s distance is compared to its cluster mean and with other clusters mean and relocate the users according to the smallest distance from any cluster’s mean.

**2.3.2 PROPOSED TECHNIQUE:** -

**Cosine Similarity Technique**

* The movies are recommended based on a simple algorithm called Cosine Similarity. Cosine similarity is a measure used to determine the similarity between two items. Mathematically it can be determined as the cosine angle between two vectors in a three-dimensional plane. We can also check the Euclidean distance between the two vectors to determine how different or similar they are from each other. In our case, one of the vectors is the movie that is searched and the rest of the movies in the database are checked as the second vector. The top ten movies which have the least Euclidean distance corresponding to the searched movie are shown as recommendations. Cosine Similarity is a type of Content-based filtering approach. The attributes of a thing are termed as “content”. Based on these attributes we can classify whether the two things are similar or not. The attributes can be words specified in the database such as genre, cast names, director names, overview, and so on. If the attributes match or have a high similarity, then the two movies can be classified as similar movies. The intuition behind this sort of recommendation system is that if a user liked a particular movie or show, he/she might like a movie or a show similar to it.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

The Requirements engineering (RE) refers to the process of defining, documenting, and maintaining requirements in the engineering design process.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system does and not how it should be implemented.

# Processor : Core i3 Processor

# Ram : 4GB DDR3 RAM

# Hard Disk : 500 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

**SOFTWARE REQUIREMENTS**

* Operating System : Linux 3.0/ Windows 7
* Back End : Azure, Python
* Version Control : GitHub
* Front End : Streamlit, Python

**3.4 FUNCTIONAL REQUIREMENTS**

Recommendation Systems are considered as one of effective knowledge management engines that helps us filter out unwanted data and provide targeted data based on the feedbacks from old data and similar data from user’s search. Many Recommendation Systems have been introduced till date following different approaches for the computation like Content Based Filtering, Collaborative Filtering and Hybrid models for recommendation.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**The major non-functional Requirements of the system are as follows**

**Usability**

The system is designed with completely automated process hence there is no or less user intervention.

**Reliability**

The system is more reliable because of the qualities that are inherited from the chosen platform Azure. The code built by using python is more reliable.

**Performance**

This system is developing in the high-level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

**Supportability**

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is built into the system.

**Implementation**

The system is implemented in web environment using Streamlit framework. Azure is used as the supporting back-end for hosting.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modeling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**4.2 UML DIAGRAMS**

**4.2.1 USE CASE DIAGRAM**

A picture containing text

Description automatically generated

**EXPLANATION:**

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.**4.2.2 CLASS DIAGRAM**

**Text

Description automatically generated**

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

**4.2.3 OBJECT DIAGRAM**

Graphical user interface, application, website

Description automatically generated

**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**4.2.4 STATE DIAGRAM**

**Graphical user interface, application

Description automatically generated**

**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration, and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**4.2.5 ACTIVITY DIAGRAM**

**Text

Description automatically generated**

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**4.2.6 SEQUENCE DIAGRAM**

**Diagram

Description automatically generated with medium confidence**

**EXPLANATION:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

**4.2.7 COLLABORATION DIAGRAM**

Text

Description automatically generated with medium confidence

**EXPLANATION:**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

**4.2.8 COMPONENT DIAGRAM**

**Graphical user interface, application

Description automatically generated**

**EXPLANATION**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicate dependencies.

**4.2.9 DEPLOYMENT DIAGRAM**

**Text

Description automatically generated**

**EXPLANATION:**

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

**4.3.1 Data Flow Diagram**

|  |  |
| --- | --- |
| **Level 0** | **Level 1** |
|  | **Text  Description automatically generated** |

**4.3.2 SYSTEM ARCHITECTURE:**

**A picture containing application

Description automatically generated**

**Fig. Proposed Methodology**

**CHAPTER 5**

**DEVELOPMENT TOOLS**

**5.1 GENERAL**

**Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

## History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

#### Importance of Python

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
* **Python is Interactive** − You can sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### Features of Python

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are

Listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to bytecode for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Libraries used in python:**

* NumPy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as data frames.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



Figure: NumPy, Pandas, Matplotlib, Scikit-learn

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**CODING:**

#Importing dependences

import numpy as np

import pandas as pd

#Creating a Pandas DataFrame by reading the dataset using read\_csv from Pandas Library

movies = pd.read\_csv('C:\\Users\\pavan\\Desktop\\Project\\Movie Recommendation System based on User Preferences\\Datasets\\tmdb\_movies.csv')

credits = pd.read\_csv('C:\\Users\\pavan\Desktop\\Project\\Movie Recommendation System based on User Preferences\\Datasets\\tmdb\_credits.csv')

#For better access converting all column names to lowercase

movies= movies.rename(columns=str.lower)

credits= credits.rename(columns=str.lower)

#Merging of two DataFrames on title

movies = movies.merge(credits,on='title')

#renaming column name of movie\_id\_x to movie\_id for easy access

movies.rename(columns = {'movie\_id\_x':'movie\_id'}, inplace = True)

#Creating a new data frame with only the attributes which are having a significance in our model

movies = movies[['movie\_id','title','overview','genres','keywords','cast','crew']]

#Data Preprocessing

# Determining the total no of null values in each attribute

movies.isnull().sum()

#If the total count seems insignificant or negligible then you could drop those records

movies.dropna(inplace=True)

#Determing the total number of duplicate records

movies.duplicated().sum()

#Preprossing to convert list of dictionaries into List of elements

#But to do that we have to first make it a list from a string

#This can be done via importing ast library

# From: '[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": #"Fantasy"}, {"id": 878, "name": "Science Fiction"}]'

# To: '["Action","Adventure","Fantasy",...]'

#Helper function to convert from string to list of dictionaries and from list of dictionaries to list of #elements

def convert(obj):

L = []

for i in ast.literal\_eval(obj):

L.append(i['name'])

return L

#Sending genres,keywords, object to convert function

movies['genres'] = movies['genres'].apply(convert)

movies['keywords'] = movies['keywords'].apply(convert)

#Helper function to convert from string to list of dictionaries and from list of dictionaries to list of #elements and only select 3 Main Actors

def convert3(obj):

L=[]

counter=0

for i in ast.literal\_eval(obj):

if counter!=3:

L.append(i['name'])

counter+=1

else:

break

return L

#Sending cast object to convert3 function

movies['cast'] = movies['cast'].apply(convert3)

#Helper function to Only Selecting director from crew

def fetch\_director(text):

L = []

for i in ast.literal\_eval(text):

if i['job'] == 'Director':

L.append(i['name'])

return L

#Sending crew object to fetch\_director function

movies['crew'] = movies['crew'].apply(fetch\_director)

#In order to make things easy convert the “Overview” into List

movies['overview'] = movies['overview'].apply(lambda x : x.split())

#Inorder to reduce confusion associated with tags we will be transforming all spaces in between names

def collapse(L):

L1 = []

for i in L:

L1.append(i.replace(" ",""))

return L1

#Sending the cast, crew, genres, keywords objects to collapse function

movies['cast'] = movies['cast'].apply(collapse)

movies['crew'] = movies['crew'].apply(collapse)

movies['genres'] = movies['genres'].apply(collapse)

movies['keywords'] = movies['keywords'].apply(collapse)

#Creating a new attribute by concatenating overview, genre, keywords, cast, crew attributes

movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']

#Creating a new DataFrame with just movie\_id, title, tags attributes

new\_df = movies[['movie\_id','title','tags']]

#Now converting list into a string by joining the elements in the list

new\_df['tags'] = new\_df['tags'].apply(lambda x: " ".join(x))

new\_df.head()

#Converting tags into lowercase

new\_df['tags'] = new\_df['tags'].apply(lambda x:x.lower())

#Importing nltk

import nltk

#Here we are using it to remove redundancy in similar word patterns for which we have #PorterStemmer

from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

#Helper function to pass the porterstemmer with required objects for stemming

def stem(text):

y = []

for i in text.split():

y.append(ps.stem(i))

return " ".join(y)

#Passing the tags object to stem function to perform stemming

new\_df['tags'] = new\_df['tags'].apply(stem)

#Now using the Count Vectorizer function which is used to transform the data into a vector based #on the frequency(count) of each word that occurs in the data set.

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(max\_features=5000, stop\_words='english')

#Converting to NumPy array

vector = cv.fit\_transform(new\_df['tags']).toarray()

#Importing Cosine similarity from sklearn

from sklearn.metrics.pairwise import cosine\_similarity

#Cosine Similarity is performed on those vectors to find the Euclidean Distance to recommend the #top ften movies which are like the searched movie.

similarity = cosine\_similarity(vector)

#Importing the Pickle

#Making pickle of movies data frame

pickle.dump(new\_df.to\_dict(),open('movie\_list.pkl','wb'))

#Making pickle of Model

pickle.dump(similarity,open('similarity.pkl','wb'))

#Code for User Interface App.py

#Importing dependences

import streamlit as st

import pickle

import pandas as pd

import requests

#Functions for fetching the posters, trailers and other information of movies based on users’ input

def fetch\_backdrops(movie\_id):

response2 = requests.get(f"https://api.themoviedb.org/3/movie/{movie\_id}/videos?api\_key=44726ef95f4d79cb7001a4947fca7f53&language=en-US")

data2 = response2.json()

print(data2)

try:

return data2["results"][0]["key"]

except:

return "n73\_6vyq2v4"

def fetch\_poster(data):

try:

return "https://image.tmdb.org/t/p/w500/" + data['poster\_path']

except:

return "https://st3.depositphotos.com/1322515/35964/v/600/depositphotos\_359648638-stock-illustration-image-available-icon.jpg"

def fetch\_info(movie\_id,data):

try:

return [data["release\_date"],data["budget"],data["revenue"],data["popularity"]]

except:

return "No Info Available"

#Recommending with the help of cosine similarity

def recommend(movie):

index = movies[movies['title'] == movie].index[0]

distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])

recommended\_movies = []

recommended\_posters = []

recommended\_backdrops = []

recommended\_info = []

for i in distances[1:11]:

if i is not None:

movie\_id = movies.iloc[i[0]].movie\_id

recommended\_movies.append([movies.iloc[i[0]].title] + [movies.iloc[i[0]].genres] + [movies.iloc[i[0]].cast] + [movies.iloc[i[0]].crew] + [movies.iloc[i[0]].overview])

# fetching poster via api

response = requests.get(f'https://api.themoviedb.org/3/movie/{movie\_id}?api\_key=44726ef95f4d79cb7001a4947fca7f53&language=en-US')

data = response.json()

recommended\_posters.append(fetch\_poster(data))

recommended\_info.append(fetch\_info(movie\_id,data))

recommended\_backdrops.append(fetch\_backdrops(movie\_id))

return recommended\_movies, recommended\_posters, recommended\_info, distances, recommended\_backdrops

#Using the pickle of our movies data frame

movies\_list = pickle.load(open('movie\_list.pkl', 'rb'))

movies = pd.DataFrame(movies\_list)

#Using the pickle of our cosine similarity algorithm

similarity = pickle.load(open('similarity.pkl', 'rb'))

#Adding title to the GUI interface

st.title("Movie Recommendation System based on User Preferences (MRSBUP)")

selected\_movie\_name = st.selectbox('Looking for Similar Movies?', movies['title'].values)

#Function for displaying the trailer or relevant video

def display(fetch):

print(fetch)

video\_html = """

<style>

.video-container{

width: 60vw;

height: 100vh;

position: absolute;

min-width: 80%;

filter: brightness(60%);

}

iframe {

position: absolute;

top: 52.5%;

left: 60%;

width: 100vw;

height: 100vh;

transform: translate(-50%, -50%);

}

</style>

""" + f"""

<div class="video-container">

<iframe src="https://www.youtube.com/embed/{fetch}?controls=0&autoplay=1&mute=1&loop=1"></iframe>

</div>

</body>

"""

st.markdown(video\_html, unsafe\_allow\_html=True)

#OnClick of Recommend Action

if st.button('Recommend'):

my\_bar = st.progress(0)

for percent\_complete in range(100):

time.sleep(0.01)

my\_bar.progress(percent\_complete + 1)

#Calling the recommend function

names, posters, info, distances, backdrops = recommend(selected\_movie\_name)

#With the help of library specific tabs and columns creating visual fields to display

tab1, tab2, tab3, tab4, tab5, tab6, tab7, tab8, tab9, tab10= st.tabs([names[0][0], names[1][0], names[2][0], names[3][0], names[4][0],names[5][0], names[6][0], names[7][0], names[8][0], names[9][0]])

with tab1:

display(backdrops[0])

col1, col2 = st.columns(2)

with col1:

st.image(posters[0], width=350)

with col2:

st.subheader(names[0][0])

st.caption("Release Date: "+ info[0][0])

st.caption("Budget: $" + str(info[0][1]))

st.caption("Collection: $" + str(info[0][2]))

st.caption("Popularity: " +str(info[0][3]))

st.caption("Genres: " +str(names[0][1]))

st.caption("Cast: " +str(names[0][2]))

st.caption("Director: " +str(names[0][3]))

st.caption("Similarity: " +str(distances[1][1]))

st.caption("Overview: ")

st.caption(names[0][4])

with tab2:

display(backdrops[1])

col3, col4 = st.columns(2)

with col3:

st.image(posters[1], width=350)

with col4:

st.subheader(names[1][0])

st.caption("Release Date: "+ info[1][0])

st.caption("Budget: $" + str(info[1][1]))

st.caption("Collection: $" + str(info[1][2]))

st.caption("Popularity: " +str(info[1][3]))

st.caption("Genres: " +str(names[1][1]))

st.caption("Cast: " +str(names[1][2]))

st.caption("Director: " +str(names[1][3]))

st.caption("Similarity: " +str(distances[2][1]))

st.caption("Overview: ")

st.caption(names[1][4])

with tab3:

display(backdrops[2])

col5, col6 = st.columns(2)

with col5:

st.image(posters[2], width=350)

with col6:

st.subheader(names[2][0])

st.caption("Release Date: "+ info[2][0])

st.caption("Budget: $" + str(info[2][1]))

st.caption("Collection: $" + str(info[2][2]))

st.caption("Popularity: " +str(info[2][3]))

st.caption("Genres: " +str(names[2][1]))

st.caption("Cast: " +str(names[2][2]))

st.caption("Director: " +str(names[2][3]))

st.caption("Similarity: " +str(distances[3][1]))

st.caption("Overview: ")

st.caption(names[2][4])

with tab4:

display(backdrops[3])

col7, col8 = st.columns(2)

with col7:

st.image(posters[3], width=350)

with col8:

st.subheader(names[3][0])

st.caption("Release Date: "+ info[3][0])

st.caption("Budget: $" + str(info[3][1]))

st.caption("Collection: $" + str(info[3][2]))

st.caption("Popularity: " +str(info[3][3]))

st.caption("Genres: " +str(names[3][1]))

st.caption("Cast: " +str(names[3][2]))

st.caption("Director: " +str(names[3][3]))

st.caption("Similarity: " +str(distances[4][1]))

st.caption("Overview: ")

st.caption(names[3][4])

with tab5:

display(backdrops[4])

col9, col10 = st.columns(2)

with col9:

st.image(posters[4], width=350)

with col10:

st.subheader(names[4][0])

st.caption("Release Date: "+ info[4][0])

st.caption("Budget: $" + str(info[4][1]))

st.caption("Collection: $" + str(info[4][2]))

st.caption("Popularity: " +str(info[4][3]))

st.caption("Genres: " +str(names[4][1]))

st.caption("Cast: " +str(names[4][2]))

st.caption("Director: " +str(names[4][3]))

st.caption("Similarity: " +str(distances[5][1]))

st.caption("Overview: ")

st.caption(names[4][4])

with tab6:

display(backdrops[5])

col11, col12 = st.columns(2)

with col11:

st.image(posters[5], width=350)

with col12:

st.subheader(names[5][0])

st.caption("Release Date: "+ info[5][0])

st.caption("Budget: $" + str(info[5][1]))

st.caption("Collection: $" + str(info[5][2]))

st.caption("Popularity: " +str(info[5][3]))

st.caption("Genres: " +str(names[5][1]))

st.caption("Cast: " +str(names[5][2]))

st.caption("Director: " +str(names[5][3]))

st.caption("Similarity: " +str(distances[6][1]))

st.caption("Overview: ")

st.caption(names[5][4])

with tab7:

display(backdrops[6])

col13, col14 = st.columns(2)

with col13:

st.image(posters[6], width=350)

with col14:

st.subheader(names[6][0])

st.caption("Release Date: "+ info[6][0])

st.caption("Budget: $" + str(info[6][1]))

st.caption("Collection: $" + str(info[6][2]))

st.caption("Popularity: " +str(info[6][3]))

st.caption("Genres: " +str(names[6][1]))

st.caption("Cast: " +str(names[6][2]))

st.caption("Director: " +str(names[6][3]))

st.caption("Similarity: " +str(distances[7][1]))

st.caption("Overview: ")

st.caption(names[6][4])

with tab8:

display(backdrops[7])

col15, col16 = st.columns(2)

with col15:

st.image(posters[7], width=350)

with col16:

st.subheader(names[7][0])

st.caption("Release Date: "+ info[7][0])

st.caption("Budget: $" + str(info[7][1]))

st.caption("Collection: $" + str(info[7][2]))

st.caption("Popularity: " +str(info[7][3]))

st.caption("Genres: " +str(names[7][1]))

st.caption("Cast: " +str(names[7][2]))

st.caption("Director: " +str(names[7][3]))

st.caption("Similarity: " +str(distances[8][1]))

st.caption("Overview: ")

st.caption(names[7][4])

with tab9:

display(backdrops[8])

col17, col18 = st.columns(2)

with col17:

st.image(posters[8], width=350)

with col18:

st.subheader(names[8][0])

st.caption("Release Date: "+ info[8][0])

st.caption("Budget: $" + str(info[8][1]))

st.caption("Collection: $" + str(info[8][2]))

st.caption("Popularity: " +str(info[8][3]))

st.caption("Genres: " +str(names[8][1]))

st.caption("Cast: " +str(names[8][2]))

st.caption("Director: " +str(names[8][3]))

st.caption("Similarity: " +str(distances[9][1]))

st.caption("Overview: ")

st.caption(names[8][4])

with tab10:

display(backdrops[9])

col19, col20 = st.columns(2)

with col19:

st.image(posters[9], width=350)

with col20:

st.subheader(names[9][0])

st.caption("Release Date: "+ info[9][0])

st.caption("Budget: $" + str(info[9][1]))

st.caption("Collection: $" + str(info[9][2]))

st.caption("Popularity: " +str(info[9][3]))

st.caption("Genres: " +str(names[9][1]))

st.caption("Cast: " +str(names[9][2]))

st.caption("Director: " +str(names[9][3]))

st.caption("Similarity: " +str(distances[10][1]))

st.caption("Overview: ")

st.caption(names[9][4])

**CHAPTER 7**

**7.1 SNAPSHOTS**

Graphical user interface, text, application

Description automatically generated

Figure 1: Interface Before Recommending

Graphical user interface, application

Description automatically generated

Figure 2: Movie Selection

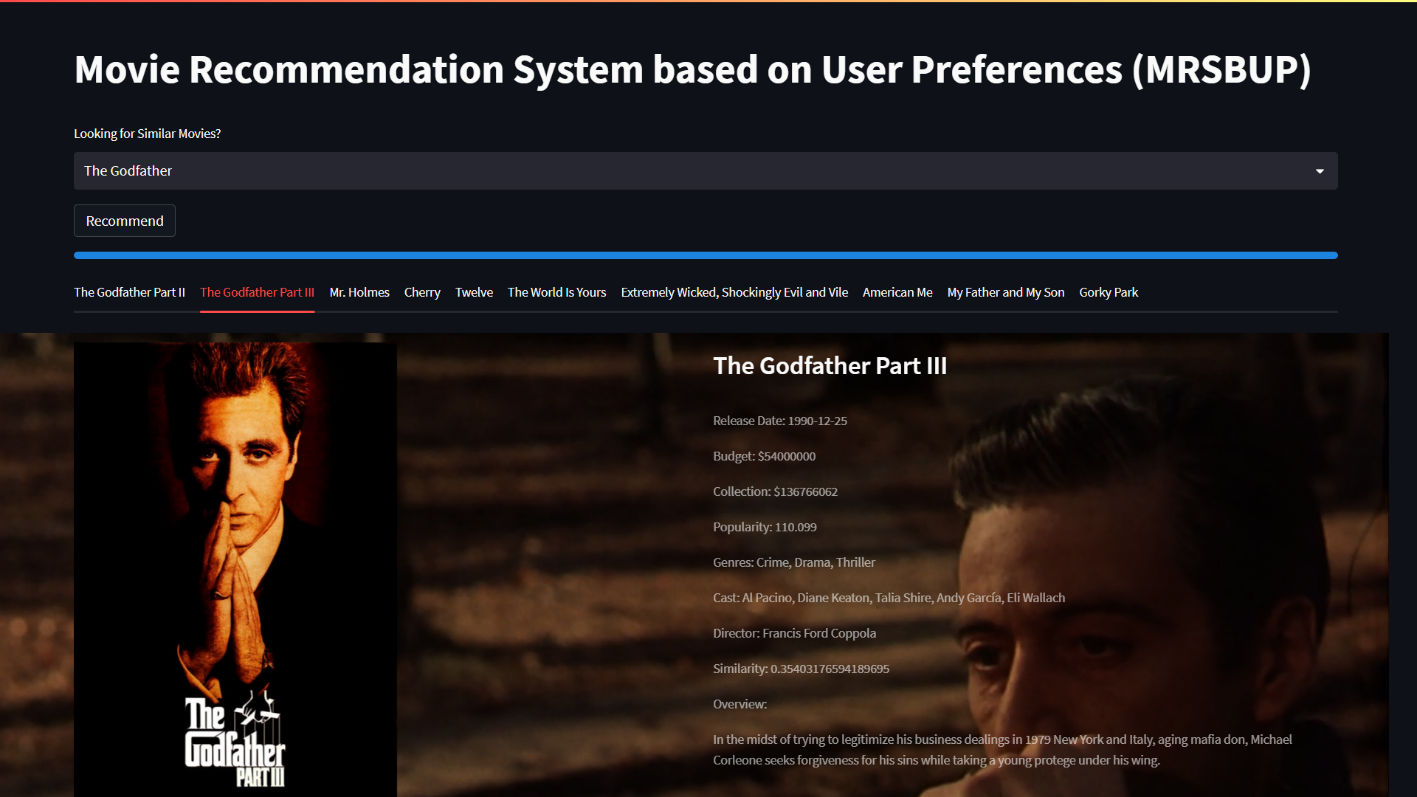


Figure 3: After Clicking Recommend

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3TYPES OF TESTS**

**8.3.1 UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 FUNCTIONAL TESTING**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

**8.3.3 SYSTEM TESTING**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 PERFORMANCE TESTING**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 INTEGRATION TESTING**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6 ACCEPTANCE TESTING**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**ACCEPTANCE TESTING FOR DATA SYNCHRONIZATION:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updating process

**8.3.7 BUILD THE TEST PLAN**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**8.4 TEST CASES**

**8.4.1 TEST CASES (INTERNAL)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **Test Description** | **Test Date** | **Expected Result** | **Actual Result** | **Remarks** |
| 1. | Recommendation System Model | 01/10/2022 | Recommend Ten Movies | Recommended Ten Movies | Pass |
| 2. | Containerization of Web Application | 20/10/2022 | Composing Up Images and Pushing to Docker Registry. | Successful Composing Up of Images and Pushing to Docker Registry with Logs. | Pass |
| 3. | Docker Container on Azure Container Registry and Deploying Image via Azure Web App | 24/10/2022 | Azure Container Registry Creation and Deploying of Web App. | Successful Azure Container Registry Creation and Deploying of Web App. | Pass |
| 4. | API Communication & Web App Service Recommendation System | 26/10/2022 | API Connection & Displaying of Ten Similar Movies with Necessary Details. | Successful API Connection & Displaying of Ten Similar Movies with Necessary Details. | Pass |
| 5. | Redeploying Web App Service | 30/11/2022 | Redeploying of Web App. | Successful Redeploying of Web App. | Pass |

**8.4.2 TEST CASES (EXTERNAL)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No.** | **Test Description** | **Test Date** | **Expected Result** | **Actual Result** | **Remarks** |
| **1.** | Web App Service Recommendation System | 01/11/2022 | Displaying of Ten Similar Movies with Necessary Details. | Displaying of Ten Similar Movies with Necessary Details. | Pass |
| **2.** | User Input: The Matrix  Recommend Button: Not Pressed | 2/11/2022 | No Action | No Action | Pass |
| **3.** | User Input: The Matrix  Recommend Button: Pressed | 2/11/2022 | Similar Movie Recommendations | Ten Movie Recommendations with Similarity Index Ranging from 0.51 to 0.19. | Pass |
| **4.** | User Input: Sherlock Holmes  Recommend Button: Pressed | 5/11/2022 | Similar Movie Recommendations | Ten Movie Recommendations with Similarity Index Ranging from 0.51 to 0.22. | Pass |
| **5.** | User Input: Harry Potter and the Philosopher's Stone  Recommend Button: Pressed | 7/11/2022 | Similar Movie Recommendation (Might Include Other Harry Potter Movies) | Ten Movie Recommendations with Similarity Index Ranging from 0.46 to 0.23. | Pass |

**CHAPTER 9**

**FUTURE ENHANCEMENT**

**9.1 FUTURE ENHANCEMENTS:**

For future we wish to make it a Content-based Hybrid Model and not just push a few details about the movie and trailers we wish to convert now microservice to a big web application. With increase in queries, we wish to reduce the time complexity by caching results. Use more search optimization techniques as well and real time API interaction with any Streaming Platform.

**CHAPTER 10**

**CONCLUSION AND REFERENCES**

**10.1 CONCLUSION**

When the user searches for a movie that he/she has already watched and liked the Movies Recommendation System will recommend the top ten movies that are most like that movie. Moreover, the system will show a poster, related background trailer or videos of those movies. All these features will save user’s time which otherwise would have been wasted on finding a movie that he/she may or may not like.

Every month several movies are being released, the movies database only gets bigger and bigger. This would help the system to provide a more accurate recommendation to the user and in turn increase customer satisfaction.

**10.2 REFERENCES**

**[1.] Authors: R. Lavanya, Utkarsh Singh, Vibhor Tyagi; A Comprehensive Survey on Movie Recommendation Systems, IEEE 2021**

**[2.] Authors: Tanya V Yadalam, Vaishnavi M Gowda, Vanditha Shiva Kumar, Disha Girish; Career Recommendation Systems using Content based Filtering; IEEE 2020**

**[3.] Authors: Nilesh Nilesh, Madhu Kumari, Pritom Hazarika, Vishal Raman; Recommendation of Indian Cuisine Recipes based on Ingredients; IEEE 2019**

**[4.] Authors: Worasak Rueangsirarak, Teeravisit Laohapensaeng, Suppakarn Chansareewittay, Anusorn Yodjaiphet; The Cosine Similarity Technique for Removing the Redundancy Sample; IEEE 2019**

**[5.] Authors: Tessy Badriyah, Sefryan Azvy, Wiratmoko Yuwono, Iwan Syarif; Recommendation system for property search using content based filtering method; IEEE 2018**

**[6.] Authors: Bagher Rahimpour Cami, Hamid A; Content-based based on Temporal Movie User Recommender Preferences System; IEEE 2017**

**[7.] Educator, StatQuest, Chapel Hill, North Carolina, United States**

**[8.] An effective collaborative movie recommender system with cuckoo search ; Rahul Katarya ; Om Prakash Verma ; Department of Computer Science Engineering, Delhi Technological University, Delhi, India.**

**[9.] Ehsan Aslanian, Mohammadreza Radmanesh, and Mahdi Jalili. Hybrid recommender systems based on content feature relationship. IEEE Transactions on Industrial Informatics, IEEE 2016**

**[10.] Minara P Anto, Mejo Antony, KM Muhsina, Nivya Johny, Vinay James, and Aswathy Wilson. Product rating using sentiment analysis. In International Con-ference on Electrical, Electronics, and Optimization Techniques, pages 3458– 3462. IEEE 2016.**

**[11.] Erik Cambria. Affective computing and sentiment analysis. IEEE Intelligent Systems, 31(2):102–107, 2016**

**[12.] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Twitter-based user mod-eling for news recommendations. In International Joint Conference on Artificial Intelligence, volume 13, pages 2962–2966, 2013.**

**[13.] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Analyzing user modeling on twitter for personalized news recommendations. In International Conference on User Modeling, Adaptation, and Personalization, pages 1–12. Springer, 2011.**

**[14.] Jesus Bobadilla, Fernando Ortega, Antonio Hernando, and Javier Alcala´. Im-proving collaborative filtering recommender system results and performance us-ing genetic algorithms. Knowledge-based systems, 24(8):1310–1316, 2011.**

**[15.] Ivan´ Cantador, Alejandro Bellog´ın, and David Vallet. Content-based recommen-dation in social tagging systems. In Proceedings of the Fourth Conference on Recommender systems, pages 237–240. ACM, 2010.**

**[16.] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recom-mender algorithms on top-n recommendation tasks. In Proceedings of the Fourth Conference on Recommender Systems, pages 39–46. ACM, 2010. ISBN 978-1-60558-906-0.**

**[17.] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6):734–749, 2005.**

**[18.] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. ACM Transactions on Information Systems, 22(1):143–177, 2004. ISSN 1046-8188.**

**[19.] Robin Burke. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4):331–370, 2002.**