**CHAPTER 1**

**INTRODUCTION**

**1.1 Overview**

A recommendation system or recommendation engine is a model used for information filtering where it tries to predict the preferences of a user and provide suggests based on these preferences. These systems have become increasingly popular nowadays and are widely used today in areas such as movies, music, books, videos, clothing, restaurants, food, places and other utilities.

These systems collect information about a user's preferences and behaviour, and then use this information to improve their suggestions in the future. Movies are a part and parcel of life. There are different types of movies like some for entertainment, some for educational purposes, some are animated movies for children, and some are horror movies or action films. Movies can be easily differentiated through their genres like comedy, thriller, animation, action etc.

Other way to distinguish among movies can be either by releasing year, language, director etc. Watching movies online, there are a number of movies to search in our most liked movies. Movie Recommendation Systems helps us to search our preferred movies among all of these different types of movies and hence reduce the trouble of spending a lot of time searching our favourable movies.

So, it requires that the movie recommendation system should be very reliable and should provide us with the recommendation of movies which are exactly same or most matched with our preferences. A large number of companies are making use of recommendation systems to increase user interaction and enrich a user's shopping experience.

**1.2 Problem Statement**

The goal of the project is to recommend a movie to the user. Providing related content out of relevant and irrelevant collection of items to users of online service providers.

**1.3 Objective of Project**

• Improving the Accuracy of the recommendation system

• Improve the Quality of the movie Recommendation system

• Improving the Scalability.

• Enhancing the user experience.

**1.4 Scope of Project**

The objective of this project is to provide accurate movie recommendations to users. The goal of the project is to improve the quality of movie recommendation system, such as accuracy, quality and scalability of system than the pure approaches. This is done using Hybrid approach by combining content based filtering and collaborative filtering, To eradicate the overload of the data, recommendation system is used as information filtering tool in social networking sites.

Hence, there is a huge scope of exploration in this field for improving scalability, accuracy and quality of movie recommendation systems Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffers with poor recommendation quality and scalability issues.

**1.5 Methodology for Movie Recommendation**

The hybrid approach proposed an integrative method by merging fuzzy k means clustering method and genetic algorithm based weighted similarity measure to construct a movie recommendation system. The proposed movie recommendation system gives finer similarity metrics and quality than the existing Movie recommendation system but the computation time which is taken by the proposed recommendation system is more than the existing recommendation system.

This problem can be fixed by taking the clustered data points as an input dataset The proposed approach is for improving the scalability and quality of the movie recommendation system. We use a Hybrid approach, by unifying Content-Based Filtering and Collaborative Filtering, so that the approaches can be profited from each other. For computing similarity between the different movies in the given dataset efficiently and in least time and to reduce computation time of the movie recommender engine we used cosine similarity measure.

The success of a recommendation system often depends on the quality and relevance of the underlying data, as well as the chosen algorithm. Regularly updating and refining the system based on user feedback is essential for continuous improvement.

**CHAPTER 2**

**LITERATURE SURVEY**

Over the years, many recommendation systems have been developed using either collaborative, content based or hybrid filtering methods. These systems have been implemented using various big data and machine learning algorithms.

**2.1 Movie Recommendation System by K-means Clustering and K-Nearest Neighbour**

A recommendation system collect data about the user’s preferences either implicitly or explicitly on different items like movies. An implicit acquisition in the development of movie recommendation system uses the user’s behaviour while watching the movies.

On the other hand, a explicit acquisition in the development of movie recommendation system uses the user’s previous ratings or history. The other supporting technique that are used in the development of recommendation system is clustering. Clustering is a process to group a set of objects in such a way that objects in the same clusters are more similar to each other than to those in other clusters. K-Means Clustering along with K-Nearest Neighbour is implemented on the movie lens dataset in order to obtain the best-optimized result. In existing technique, the data is scattered which results in a high number of clusters while in the proposed technique data is gathered and results in a low number of clusters.

The process of recommendation of a movie is optimized in the proposed scheme. The proposed recommender system predicts the user’s preference of a movie on the basis of different parameters. The recommender system works on the concept that people are having common preference or choice. These users will influence on each other’s opinions. This process optimizes the process and having lower RMSE.

**2.2 Movie Recommendation System Using Collaborative Filtering**

Collaborative filtering systems analyze the user's behaviour and preferences and predict what they would like based on similarity with other users. There are two kinds of collaborative filtering systems; user-based recommender and item-based recommender.

**2.2.1 Use-based filtering:** User-based preferences are very common in the field of designing personalized systems. This approach is based on the user's likings. The process starts with users giving ratings (1-5) to some movies. These ratings can be implicit or explicit. Explicit ratings are when the user explicitly rates the item on some scale or indicates a thumbs-up/thumbs-down to the item. Often explicit ratings are hard to gather as not every user is much interested in providing feedbacks. In these scenarios, we gather implicit ratings based on their behaviour. For instance, if a user buys a product more than once, it indicates a positive preference.

In context to movie systems, we can imply that if a user watches the entire movie, he/she has some likeability to it. Note that there are no clear rules in determining implicit ratings. Next, for each user, we first find some defined number of nearest neighbours. We calculate correlation between users' ratings using Pearson Correlation algorithm. The assumption that if two users' ratings are highly correlated, then these two users must enjoy similar items and products is used to recommend items to users.

**2.2.2 Item-based filtering**: Unlike the user-based filtering method, item based focuses on the similarity between the item’s users like instead of the users themselves. The most similar items are computed ahead of time. Then for recommendation, the items that are most similar to the target it.

**CHAPTER 3**

**SYSTEM REQUIREMENTS SPECIFICATION**

System Requirement Specification (SRS) is a fundamental document, which forms the foundation of the software development process. The System Requirements Specification (SRS) document describes all data, functional and behavioral requirements of the software under production or development. An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time (usually) prior to any actual design or development work. It's a two- way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it. It is important to note that an SRS contains functional and non-functional requirements only. It doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements.

This chapter involves both the hardware and software requirements needed for the project and detailed explanation of the specifications.

**3.1 Hardware Requirements**

• A PC with Windows/Linux OS

• Processor with 1.7-2.4gHz speed

• Minimum of 8GB RAM

**3.2 Software Specifications**

• Text Editor (VS-code/WebStorm)

• Anaconda distribution package (PyCharm Editor)

• Python libraries

**3.3 Functional Requirements**

User Registration and Authentication: Users should be able to create accounts securely, log in, and the system must ensure robust authentication mechanisms to protect user data.

Recommendation Engine: The heart of the system lies in its recommendation engine. Depending on the chosen approach, be it collaborative filtering, content-based filtering, or a hybrid method, the engine should generate accurate and personalized book recommendations for users.

User Interface: An intuitive and user-friendly interface is essential for users to interact seamlessly with the recommendation system. The interface should allow users to explore recommended books, provide feedback, and navigate effortlessly through the application.

Feedback Mechanism: Users should have the ability to provide feedback on recommended books. This feedback loop is crucial for refining the recommendation algorithms over time and improving the overall accuracy of the system.

**3.4 Non-Functional Requirements**

• Usability

• Availability

• Efficiency

• Flexibility

**3.5 Software Requirements**

3.5.1 Anaconda Distribution:

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system. The anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.3.

3.5.2 Python Libraries:

NumPy: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Pandas: Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Data frame.

Pandas: It is a powerful and versatile library that simplifies tasks of data manipulation in Python. Pandas is built on top of the NumPy library and is particularly well suited for working with tabular data, such as spreadsheets or SQL tables. Its versatility and ease of use make it an essential tool for data analysts, scientists, and engineers working with structured data in Python.

Streamlit: It is a popular Python library designed for creating web applications with minimal effort, making it particularly accessible for data scientists and developers. With Streamlit, users can transform data scripts into interactive and shareable web apps with just a few lines of code. Its simplicity lies in its declarative syntax, allowing users to focus on data visualization and functionality rather than intricate web development details.

**CHAPTER 4**

**SYSTEM ANALYSIS AND DESIGN**

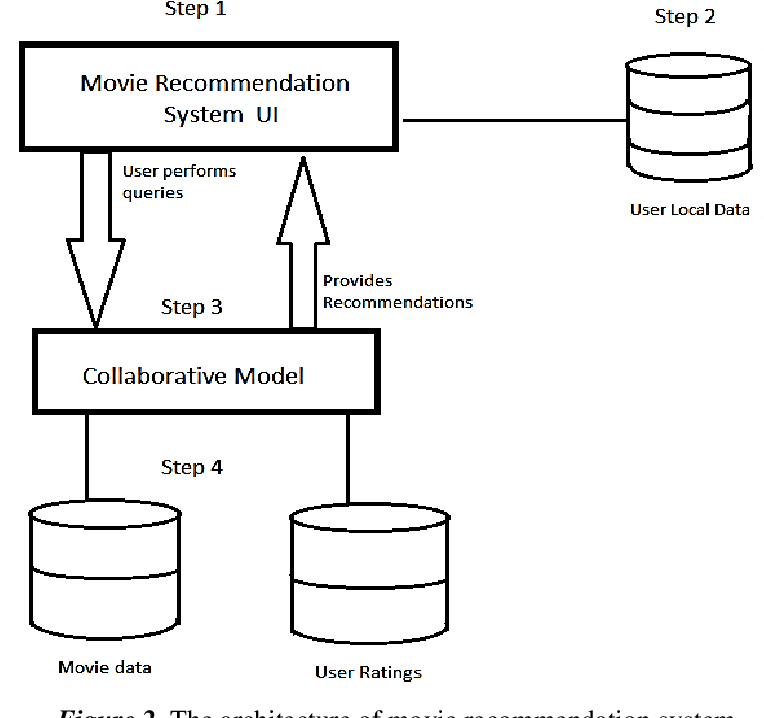
**4.1 System Architecture and Proposed System**

The proposed movie recommendation system employs a scalable and modular system architecture to ensure efficient performance and adaptability. At its core, the system utilizes a microservices architecture, where distinct components handle specific functionalities such as user authentication, recommendation algorithms, and content management. The front-end interface is designed using a responsive web framework, providing users with an intuitive and seamless experience across various devices.

The recommendation engine incorporates collaborative filtering and content-based algorithms, dynamically adjusting recommendations based on user preferences and feedback. A robust backend, built on a cloud infrastructure, manages the movie database, user profiles, and system analytics. Additionally, the system integrates with external APIs for real time updates on movie releases and social media sharing features, enhancing the overall user engagement and satisfaction. The modular and scalable architecture enables easy maintenance, updates and future expansion, ensuring the movie recommendation system’s adaptability to evolving user needs and technological advancements.

For each different individual use different list of movies are recommended, as user login or enters the user id based on two different approaches used in the project each will recommend the set of movies to the particular user by combining the both the set of movie based on the user the hybrid model will recommend the single list of movie to the user.

This architecture provides a comprehensive approach to building a movie recommendation system, combining user interaction, data processing, and machine learning to deliver personalized movie recommendations. The flexibility to incorporate additional features and external data sources allows for continuous improvement and adaptation to user preferences over time.



**Fig. 4.1 Architecture for hybrid approach**

**4.2 Activity Diagram**

An activity diagram is a type of UML (Unified Modeling Language) diagram that illustrates the flow of activities and actions within a system. In the context of a movie recommendation system, an activity diagram can help visualize the different activities and interactions that take place. Here's an explanation of an activity diagram for a movie recommendation system.

Start: The diagram typically starts with a rounded rectangle, indicating the beginning of the process. In this case, it could represent the initiation of the movie recommendation system.

User Login: The user begins by logging into the system. This involves interacting with the user interface to provide login credentials.

User Registration (Optional): If the user is not registered, there might be a branch to a registration process. This step involves entering necessary information to create a user profile.

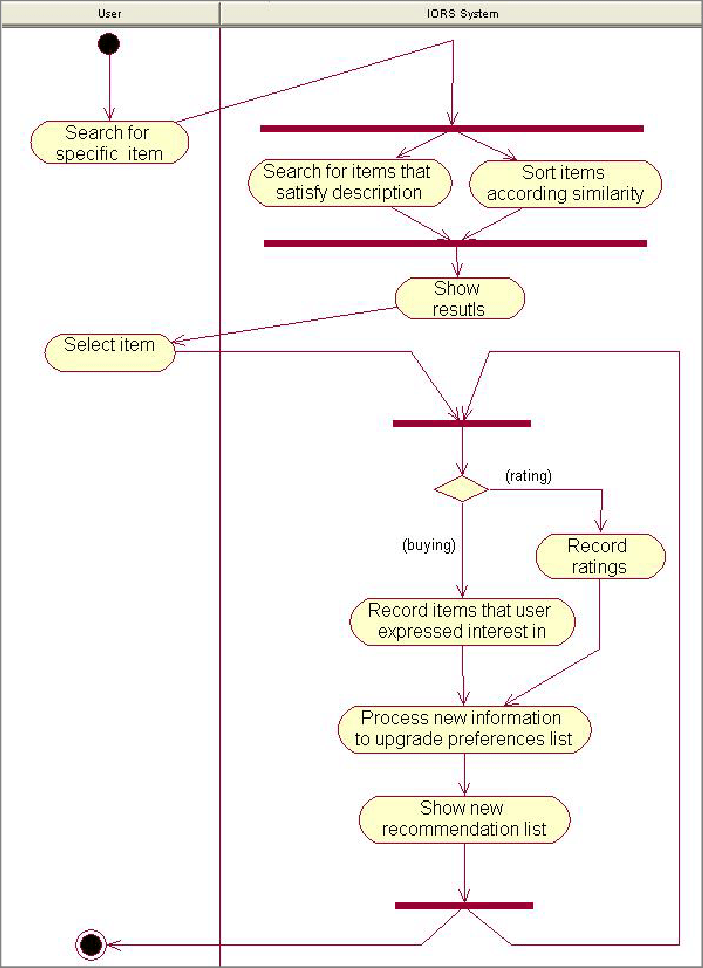
Home Page: After logging in, the user is directed to the home page of the movie recommendation system.

Browse Movies: The user has the option to browse through available movies. This activity involves interacting with the system to view a list of movies or search for specific genres, actors, or titles.

Rate Movies: Users can rate movies they have watched. This step involves providing a rating for a movie, which contributes to the system's understanding of the user's preferences.

Get Recommendations: Based on the user's viewing history and ratings, the system generates movie recommendations. This involves complex algorithms and data processing to suggest movies that align with the user's preferences.

Display Recommendations: The system displays the recommended movies to the user. The user can then choose to view more details or proceed with watching a recommended movie.



**Fig. 4.2 Activity Diagram**

**4.3 Data-Flow Diagram**

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. It is a graphical tool, useful for communicating with users, managers and other personnel. it is useful for analyzing existing as well as proposed system.

It provides an overview of-

• What data is system processes.

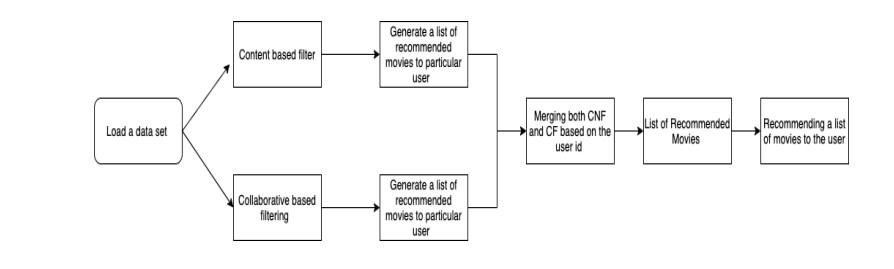
• What transformation are performed.

• What data are stored.

• What results are produced, etc.

Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

Initially load the data sets that are required to build a model the data set that are required in this project are movies.csv, ratinfg.csv, users.csv all the data sets are available in the Kaggle.com. Basically, two models are built in this project content based and collaborative filtering each produce a list of movies to a particular user by combining both based on the user id a single final list of movies are recommended to the particular user.



**Fig. 4.3 Data Flow Diagram**

**CHAPTER 5**

**RESULTS AND DISCUSSION**

**5.1 Results**

Since our project is movie recommendation system. One can develop a movie recommendation system by using either content based or collaborative filtering or combining both. In our project we have developed a hybrid approach that is combination of both content and collaborative filtering. Both the approaches have advantages and dis-advantages. In content based filtering it based on the user ratings or user likes only such kind of movie will recommended to the user.

Advantages: It is easy to design and it takes less time to compute.

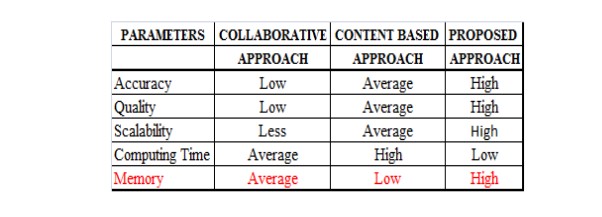
Dis-advantages: The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

In Collaborative filtering the recommendation is comparison of similar users.

Advantages: No need domain knowledge because the embeddings are automatically learned. The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.

Dis-advantages: The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item. This issue is often called the cold-start problem.

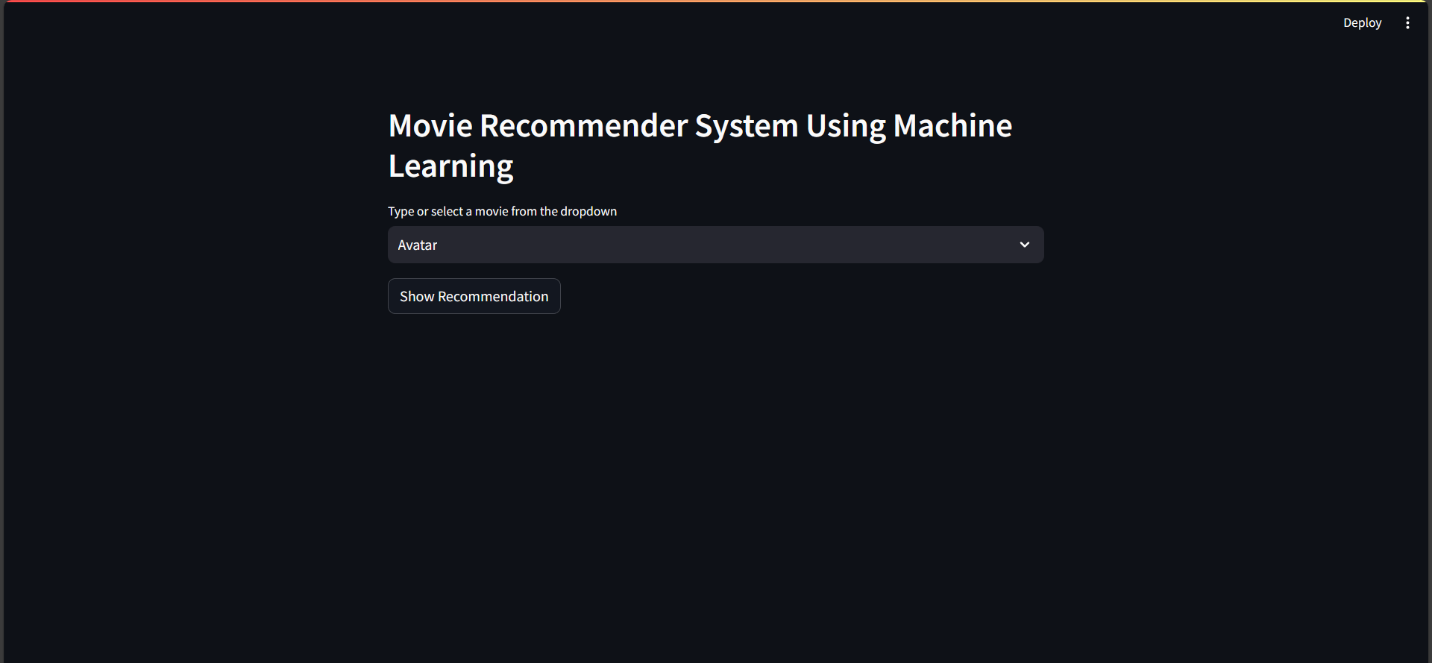
The hybrid approach will resolves all these limitations by combining both content and collaborative filtering and also takes advantages of both the approaches, content based approach and collaborative filtering.



**Table 5.1 Results**

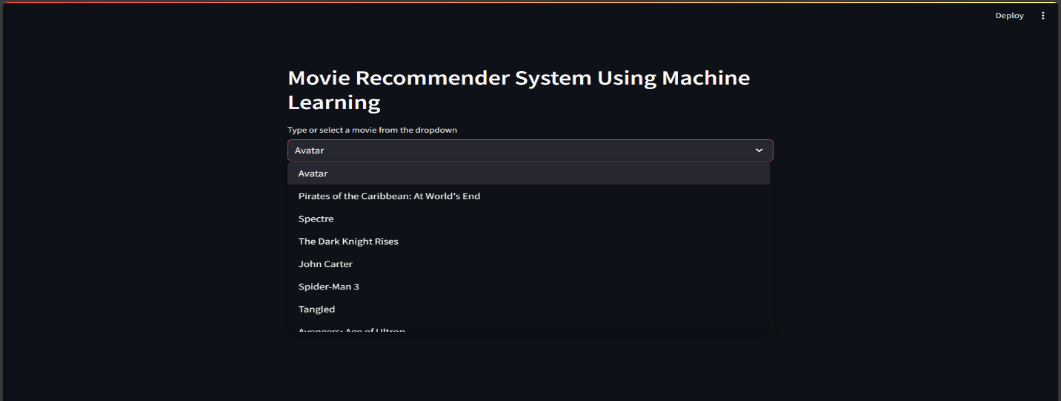
**5.2 Screenshot of the Result:**

**5.2.1** Designing the interface for a movie recommendation system involves creating a user-friendly and visually appealing platform that allows users to easily discover and explore movie recommendations. Here are some key elements and features you might consider incorporating into the interface:

****

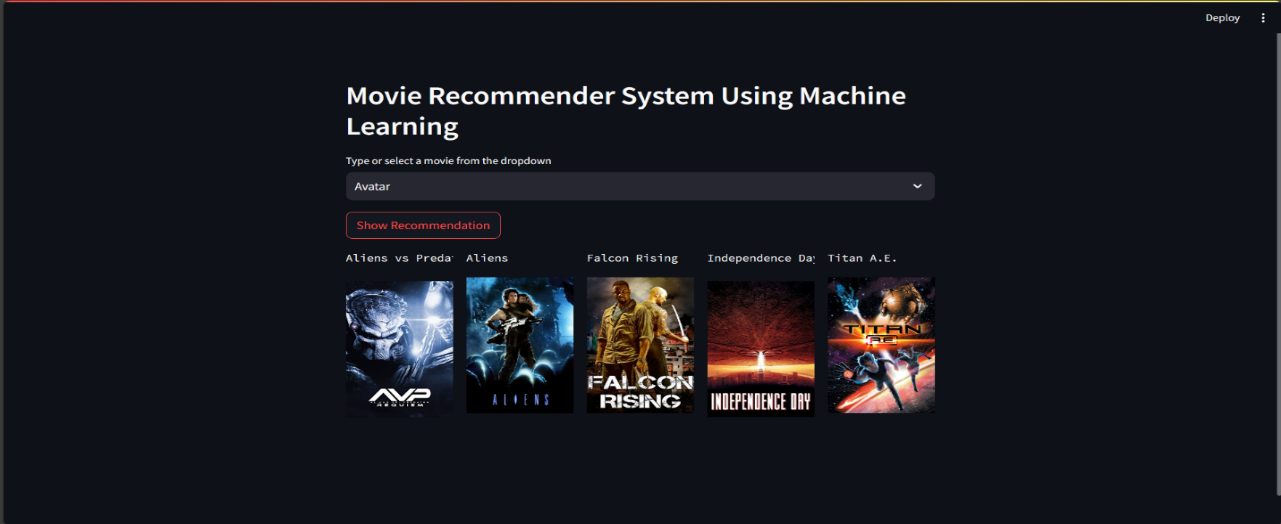
**Fig 5.1 Home Page**

**5.2.2** Once the user id is entered the list of recommended movies are displayed. In the real-world, ratings are very sparse and data points are mostly collected from very popular movies and highly engaged users. We wouldn’t want movies that were rated by a small number of users because it’s not credible enough. Similarly, users who have rated only a handful of movies should also not be taken into account.

****

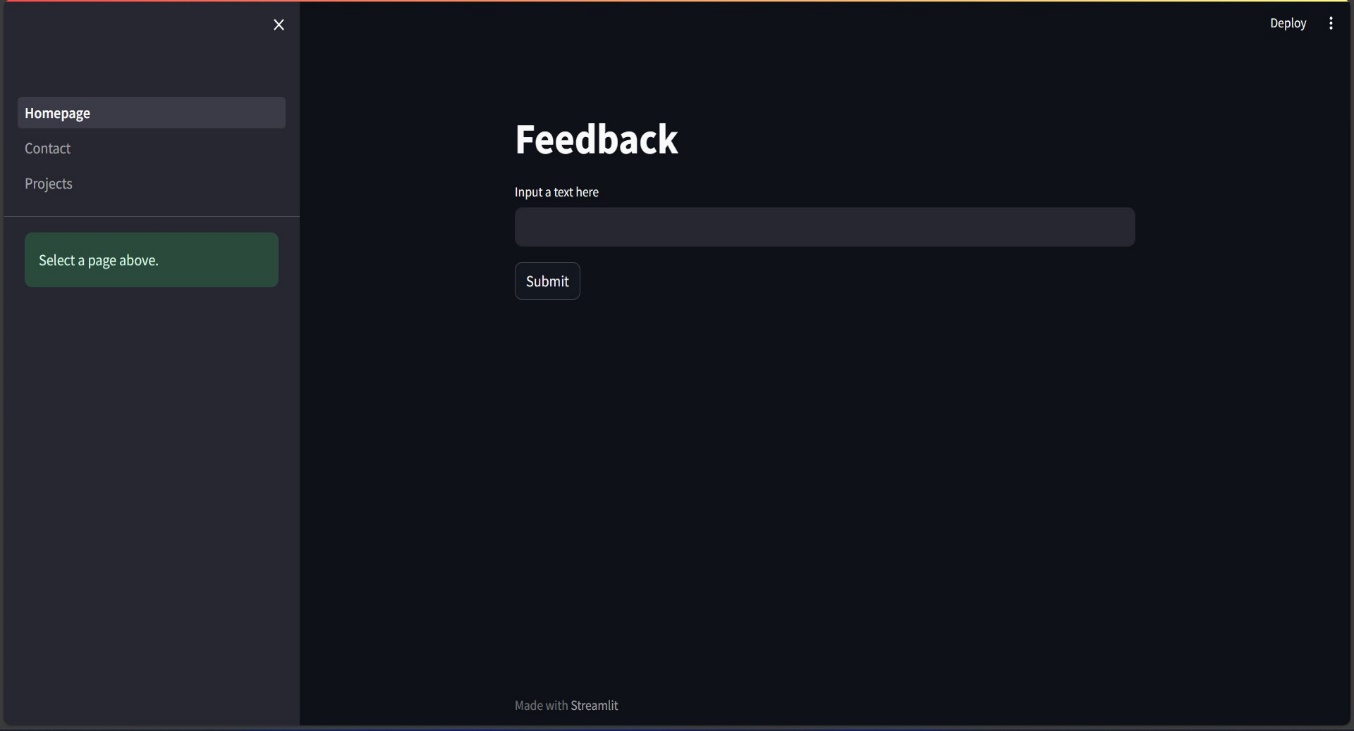
**Fig. 5.2 List of Movies**

**5.2.3** Once the user id is entered the list of recommended movies are displayed.

****

**Fig. 5.3 Recommendations**

**5.2.4** This is the Homepage of the project where user can submit the feedback of his experience.



**Fig 5.4 Feedback Page**

**CHAPTER 6**

**TESTING**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is supposed to do. In the testing stage following goals are tried to achieve:

● 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.

**6.1 Testing Methodologies:**

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

**Unit Testing**: Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process. Test Left is a tool that allows advanced testers and developers to shift left with the fastest test automation tool embedded in any IDE.

**Integration Testing:** After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

**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.

**Performance Testing:** Performance testing is performed to determine how well the system can perform in terms of responsiveness under all kinds of load. The web application is tested to see if it can sustain huge amount of requests providing higher throughput under different loads. I have simulated multiple hits on various pages of the application to evaluate the overall performance.

**CHAPTER 7**

**CONCLUSION AND FUTURE SCOPE**

**7.1 Conclusion:**

In this project, to improve the accuracy, quality and scalability of movie recommendation system, a Hybrid approach by unifying content based filtering and collaborative filtering; using Singular Value Decomposition (SVD) as a classifier and Cosine Similarity is presented in the proposed methodology. Existing pure approaches and proposed hybrid approach is implemented on three different Movie datasets and the results are compared among them. Comparative results depicts that the proposed approach shows an improvement in the accuracy, quality and scalability of the movie recommendation system than the pure approaches. Also, computing time of the proposed approach is lesser than the other two pure approaches.

Movie recommendation systems have proven to be effective in addressing the overwhelming abundance of content available to users. By narrowing down choices and offering tailored suggestions, these systems have played a pivotal role in improving content discovery. As a result, users can now explore a diverse range of movies that align with their tastes and preferences, leading to a more enjoyable and personalized viewing experience.

**7.2 Future Scope**

In the proposed approach, It has considered Genres of movies but, in future we can also consider age of user as according to the age movie preferences also changes, like for example, during our childhood we like animated movies more as compared to other movies. There is a need to work on the memory requirements of the proposed approach in the future. The proposed approach has been implemented here on different movie datasets only. It can also be implemented on the Film Affinity and Netflix datasets and the performance can be computed in the future.

Looking ahead, the future of movie recommendation systems holds exciting prospects. Here are some potential avenues for further development:

Integration of Emerging Technologies: Explore the integration of emerging technologies such as augmented reality (AR) or virtual reality (VR) to create immersive and interactive movie recommendation experiences. This could revolutionize the way users engage with content.

Enhanced Personalization: Future systems can delve deeper into user preferences by incorporating more advanced user profiling techniques, sentiment analysis, and real-time data. This can lead to even more accurate and personalized recommendations.

Cross-Platform Integration: Develop recommendation systems that seamlessly integrate across various platforms, including streaming services, social media, and offline media. This can provide users with a consistent and personalized experience across their entertainment ecosystem.

Ethical and Inclusive Recommendations: Focus on ensuring that recommendation algorithms are ethically designed, avoiding biases and promoting inclusivity. Strive to provide diverse and culturally relevant movie suggestions to cater to a global audience.

Collaborative Filtering Improvements: Refine collaborative filtering algorithms to handle sparse data more effectively and enhance their scalability. This can lead to better recommendations, especially for new or niche content.

Feedback Mechanisms: Implement robust feedback mechanisms to allow users to provide explicit feedback on recommendations, helping the system continuously learn and adapt to evolving user preferences.

Hybrid Recommendation Systems: Explore the combination of different recommendation approaches, such as content-based and collaborative filtering, to create hybrid models that leverage the strengths of each method.

In summary, the future of movie recommendation systems lies in continuous innovation, leveraging emerging technologies, and refining algorithms to provide users with an increasingly personalized and enjoyable cinematic experience. As technology advances, the potential for these systems to transform the entertainment landscape remains vast.

**BIBLIOGRAPHY**

1. <https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109>
2. <https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109>
3. <https://www.kaggle.com/code/rounakbanik/movie-recommender-systems>