MOVIE RECOMMENDATION SYSTEM

A DESIGN PROJECT REPORT

submitted by

SANTHOSH T

SHAJEETH RAHMAN S

VENGATESAN S

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

DECEMBER, 2024

MOVIE RECOMMENDATION SYSTEM

A DESIGN PROJECT REPORT

submitted by

SANTHOSH T (811722104134)

SHAJEETH RAHMAN S (811722104138)

VENGATESAN S (811722104179)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

DECEMBER, 2024

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"MOVIE RECOMMENDATION SYSTEM"** is bonafide work **SANTHOSH T (811722104134), SHAJEETH RAHMAN S (811722104138), VENGATESAN S (811722104179)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr. A Delphin Carolina Rani M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR
Department of CSE
K Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

SIGNATURE
Mrs. S Gayathri M.E,
SUPERVISOR
Assistant Professor
Department of CSE
K Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voice examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **"MOVIE RECOMMENDATION SYSTEM"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of Bachelor Of Engineering. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of Bachelor Of Engineering.

Signature

SANTHOSH T

SHAJEETH RAHMAN S

VENGATESAN S

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtness to our institution **"K RAMAKRISHNAN COLLEGE OF TECHNOLOGY"**, for providing us with the opportunity to do this project.

We are glad to credit honorable chairman **Dr. K RAMAKRISHNAN,B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering adequate duration to complete it.

We would like to thank **Dr. N VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project with full satisfaction.

We whole heartily thank **Dr. A DELPHIN CAROLINA RANI, M.E., Ph.D.,** Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing her support to pursue this project.

We express our deep and sincere gratitude and thanks to our project guide **Mrs. S GAYATHRI, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry our this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

Recommendation System is a system that seeks to predict or filter preferences according to the user's choices. Recommendation systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general, It is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly. An example of recommendation in action is when you visit Amazon and you notice that some items are being recommended to you or when Netflix recommends certain movies to you. They are also used by Music streaming applications such as Spotify and Deezer to recommend music that you might like. They gradually learn your preferences over time and suggest new products which they think you'll love. We can make this application using python language and collaborative based filtering algorithm.Collaborative filtering tackles the similarities between the users and items to perform. The next movie recommendation should be based on the user's rating to watched movies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM
BDFL	Benevolent Dictator For Life
Gunicorn	Green Unicorn
HTTP	HyperText Transfer Protocol
ORM	Object-Relational Mapping
CTR	Click-Through Rates
CF	Collaborative filtering
NCF	neural collaborative filtering

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Movie recommendation systems are widely used in online platforms to help users discover new content based on their preferences. These systems play a crucial role in enhancing user experience by providing personalized recommendations, thus increasing user engagement and satisfaction. With the growing amount of movie data, manually searching for films that match a viewer's tastes becomes increasingly challenging. Recommendation systems help alleviate this issue by analyzing users' viewing history, ratings, and demographic information, and then suggesting movies that align with their interests.

The most common techniques employed in movie recommendation systems are **collaborative filtering** and **content-based filtering**. Collaborative filtering is based on the idea that users who have rated movies similarly in the past will likely enjoy the same movies in the future. It can be divided into user-based and item-based methods, each analyzing relationships between users or items to make recommendations. In contrast, content-based filtering recommends movies based on the features of the movies themselves, such as genre, cast, director, and plot keywords

Hybrid models combine both collaborative and content-based methods to overcome the limitations of each approach. For instance, collaborative filtering may struggle with sparse data, while content-based filtering is often limited by the diversity of movie features. Hybrid models aim to leverage the strengths of both methods, resulting in better recommendations.

Recent advancements in **deep learning** and **graph-based models** have introduced more sophisticated recommendation algorithms. Techniques such as **neural collaborative filtering (NCF)** and **graph neural networks (GNN)** have enhanced the ability of movie recommendation systems to capture complex relationships between users and items

1.2 PROBLEM STATEMENT

Recommender systems are tools that aims to get the user's rating and then recommend the movies from a big set of data on the basis of the users matching interest and then classify them into different categories. The sole purpose of the whole system of this recommendation is the search for the content that it would fit into the person's interest for an individual's personal oasis. However it takes into account different factors that would create some different list of content that is specific to different categories of individual/ users . AI based algorithms that recommender systems basically used creates a list of possible different scenarios of devices and then customizing that all the interesting and matching interest/ choices of the individual categories in the end. All the results are basically based on the different activities that they have done previously such as how does the profile look what have gone through the Chrome Browser Opera browser and other Browser which includes their previously browsed history for considering the demographic traits or the possibility how they would like the movie is based on the genre, a set of predictive modelling is constructed through the data(big) which is available and then the movies are protected through the list of 2000 movies set a bunch of few selected movies are recommended using different algorithms different methods different similarity measures.

1.3 OBJECTIVE

Movie recommendation system provides the mechanism and classifying the users with the same interest and searches for the content that would be so much interesting belonging to different set of users and then creating different kind of lists and providing interesting recommendations to the individual based on the content the love. The main objective of the recommender system is to used approaches suggest demographic filtering ,content based filtering , collaborative filtering to find the set of movies with every user likes for specific set of users. The movies that have high probability of being liked by the general set of users will be displayed to the user by the recommender in the end and then in another technique we will try to find the users with different interest using the information collected through different activities an Indian in collaborative filtering will test all those users which have same type of interests to get the final set of movies to be

recommended to the users individually. So we will use different categories of recommender filtering techniques and then compare in contrast that results obtained in different methods and will try to to improve the results as h dataset for set of movies goes larger and larger above the computational bound of the system which is generally a limitation on the large dataset.

1.4 ORGANIZATION

The following is a general outline of the report's structure:

Part 1: Describes the project's overall presentation, issue proclamation venture points, and scope.

Part 2: It provides an overview of current research in the topic. It explains in full all of the research, investigations, theories, and social gatherings that took place throughout the project.

Part 3: Discusses the project's framework and plan in order to forecast the correct outcome.

Part 4: Discusses the results and provides screenshots. Part 5: Complete the project and submit a proposal for future work.

1.5 IMPLICATION

Movie recommendation systems have far-reaching implications for both users and businesses in the entertainment industry. These systems, which are built on various techniques like collaborative filtering, content-based filtering, and hybrid methods, influence how users interact with content and help content providers improve user engagement and satisfaction.

1. Personalization and User Experience: The primary implication of movie recommendation systems is the personalization of content for users. By analyzing user preferences, watching patterns, and ratings, these systems can recommend movies that match individual tastes. This results in a more satisfying and engaging user experience, as users are presented with movies they are likely to enjoy, without having to sift through large catalogs Personalized recommendations are particularly important in reducing decision fatigue, which can overwhelm users with too many choices, thereby making the discovery process more enjoyable.
2. Business Growth and Customer Retention: For streaming platforms and content providers, movie recommendation systems are essential for boosting business growth.

By suggesting relevant content, these systems can increase viewing time, customer retention, and overall platform engagement. Personalized recommendations encourage users to return to the platform for more tailored content, which helps increase subscription renewals and user loyaltyIn fact, recommendation systems contribute significantly to revenue generation, as they can drive additional views, ads, and even in-platform purchases.

3. Data Collection and Consumer Insights: Movie recommendation systems also have a significant impact on the collection and analysis of user data. By tracking user interactions, preferences, and feedback, businesses can gain valuable insights into user behavior and trends. These insights can be used not only for improving recommendation algorithms but also for refining content creation strategies. For example, platforms can identify which genres or actors attract specific demographics and develop targeted marketing campaigns or original content accordingly
4. Addressing Bias and Ethical Concerns: One of the challenges that recommendation systems face is mitigating biases in their suggestions. Systems may inadvertently reinforce stereotypes or limit exposure to diverse content by over-relying on historical data. Ensuring fairness and inclusivity in the recommendations is becoming an important area of research, especially as platforms strive to meet ethical and diversity standards
5. Scalability and Performance: As movie recommendation systems are implemented on a large scale, they must be capable of handling vast amounts of data in real-time. This scalability challenge requires constant improvements in algorithm efficiency, especially as new users and items are introduced. Techniques like matrix factorization and graph neural networks (GNN) help tackle these challenges by enhancing the efficiency of the recommendation process without compromising accuracy

In summary, movie recommendation systems have profound implications for user satisfaction, business success, data-driven strategies, and the ethical considerations of content curation. Their role is essential in driving engagement and growth in the entertainment industry, while also presenting ongoing challenges in fairness and performance that need to be addressed through innovation.

CHAPTER 2

LITERATURE SURVEY

1. Title: A Comprehensive Survey on Movie Recommendation Systems

Authors: S. Jayalakshmi, N. Ganesh, R. Cep, J. Senthil Murugan
Year:2022

This paper provides an extensive review of filtering techniques and algorithms used in movie recommendation systems, including traditional methods like collaborative filtering and modern metaheuristic-based approaches. It discusses challenges in system implementation and future research directions. The study emphasizes machine learning techniques like K-means clustering and self-organizing maps

2.Title: Analysis of Movie Recommendation Systems

Authors:Notexplicitlylisted
Year:2024

This survey reviews the latest advancements in movie recommendation systems, focusing on the integration of deep learning, reinforcement learning, and hybrid models. It also examines the challenges of scalability and real-time recommendations, along with methods for mitigating bias and improving recommendation accuracy

3.Title: Movie Recommender Systems: Concepts, Methods, Challenges.

Authors: Sambandam Jayalakshmi, Narayanan Ganesh, Robert Čep.

Year:2022

This paper reviews different recommender system models, including content-based, collaborative filtering, and hybrid models, while focusing on challenges such as information overload and system personalization. It highlights the importance of metaheuristic-based systems for improving recommendation performance

4.Title: A Comprehensive Review of Recommender Systems: Transitioning from Theory to Practice

Authors: Shaina Raza, Mizanur Rahman, Safiullah Kamawal, Armin Toroghi,

Year:2024

This survey connects theoretical research with practical applications in recommender systems, focusing on the transition from traditional to advanced techniques, such as graph-based models, deep learning, and large language models. It addresses challenges and proposes future directions for making recommender systems more scalable and trustworthy

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Existing movie recommendation systems use various techniques to suggest movies to users based on their preferences:

1. Content-Based Filtering: Recommends movies similar to those a user liked based on features like genre, director, or actors.

2. Collaborative Filtering: Suggests movies based on the preferences of users with similar tastes. Includes user-based and item-based approaches.

3. Hybrid Systems: Combines methods like content-based and collaborative filtering to improve recommendations (e.g., Netflix).

4. Deep Learning: Uses neural networks (e.g., autoencoders, RNNs) to analyze complex patterns in user behavior and movie data.

5. Context-Aware Systems: Factors in context like time, mood, or location to recommend movies dynamically.

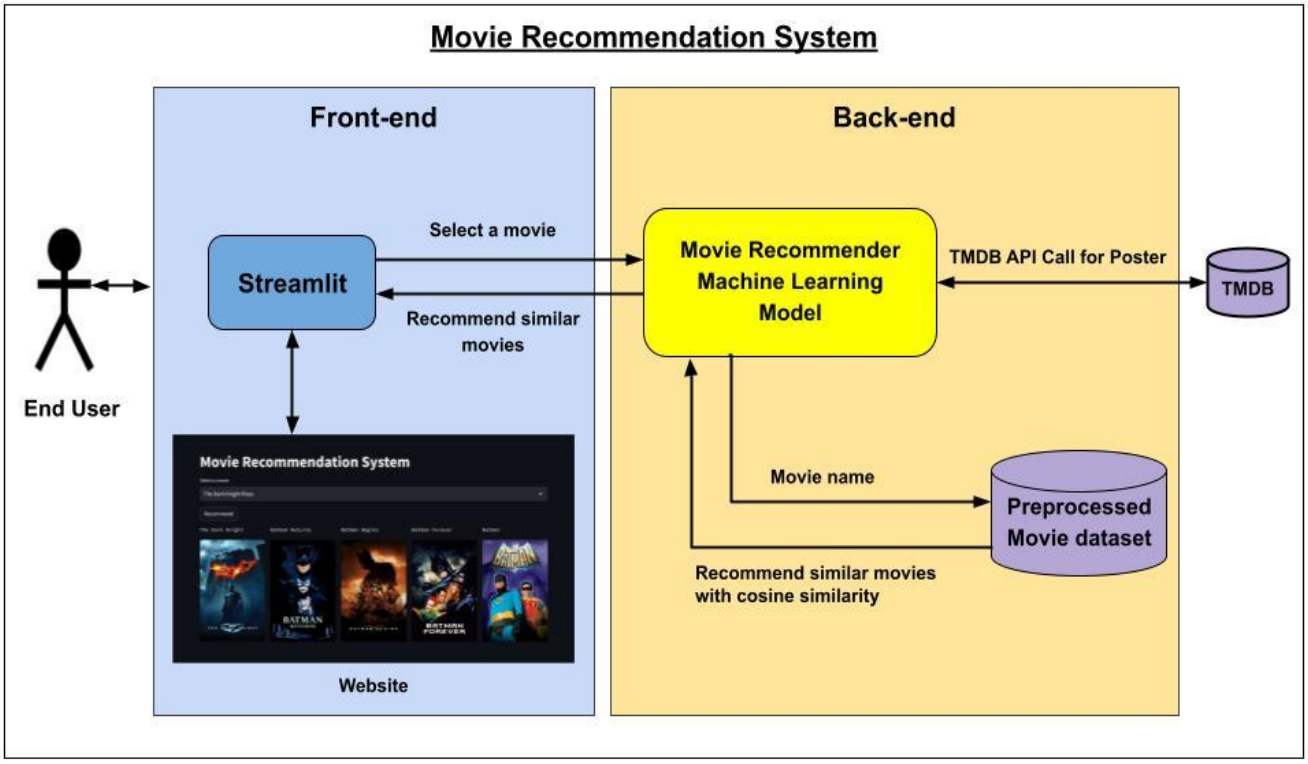

Figure 1.1: Existing System

3.2 PROPOSED SYSTEM

Collaborative filtering (CF) is one of the most widely adopted and successful recommendation approaches. Unlike many content-based approaches which utilize the attributes of users and items, CF approaches make predictions by using only the user-item interaction information. These methods can capture the hidden connections between users and items and have the ability to provide serendipitous items which are helpful to improve the diversity of recommendation.

These systems apply knowledge discovery techniques to make personalized recommendations that can help people sift through huge amount of available articles, movies, music, web pages, etc. Popular examples of such systems include product recommendation in Amazon, music recommendation in Last.fm, and movie recommendation in Movie lens..

3.3 BLOCK DIAGRAM OF PROPOSED SYSTEM

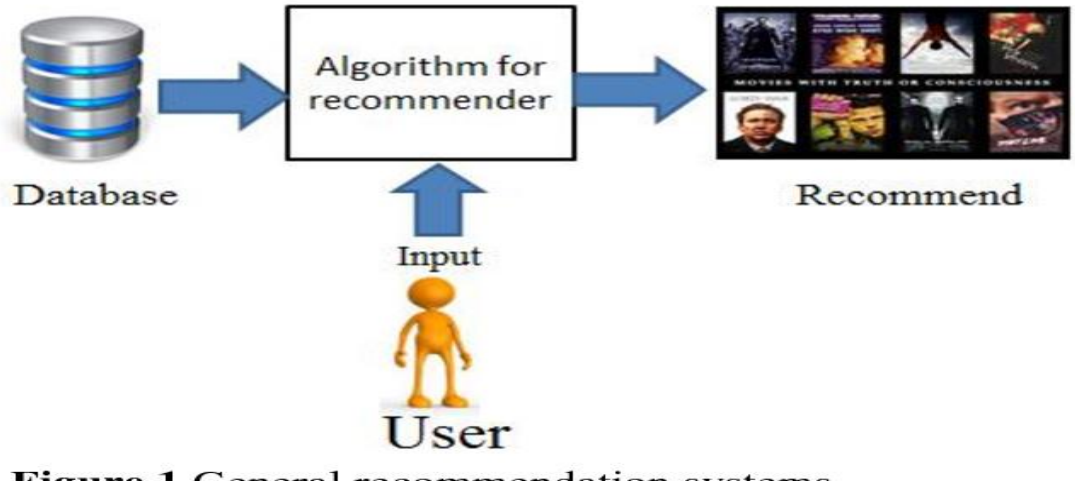

Figure 1.2: Block Diagram

3.4 FLOWCHART

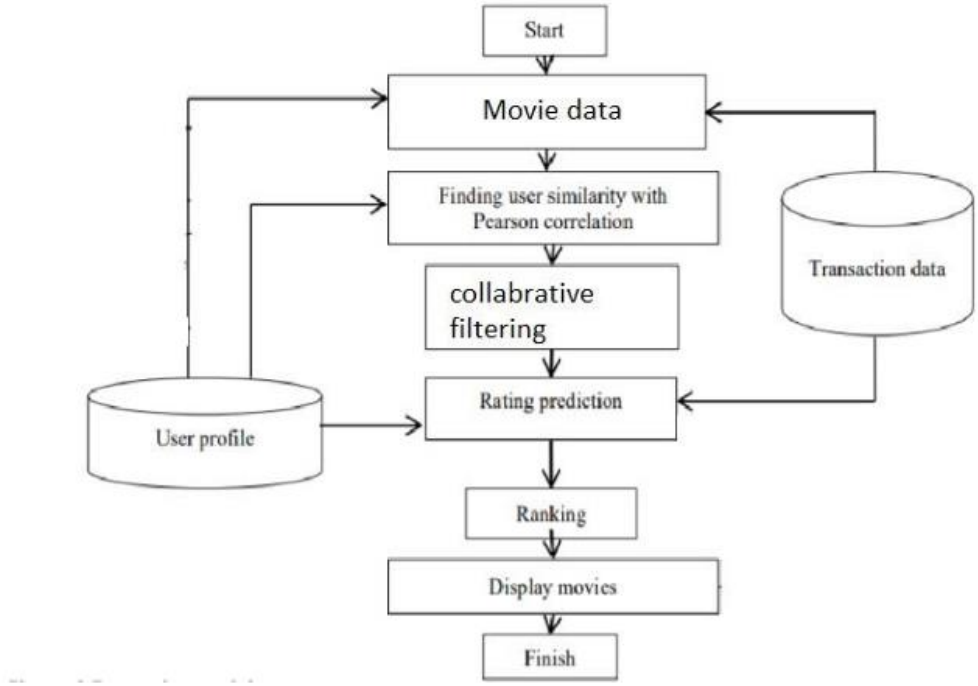

Figure 1.3 : Flow of Control

3.6 ACTIVITY DIAGRAM

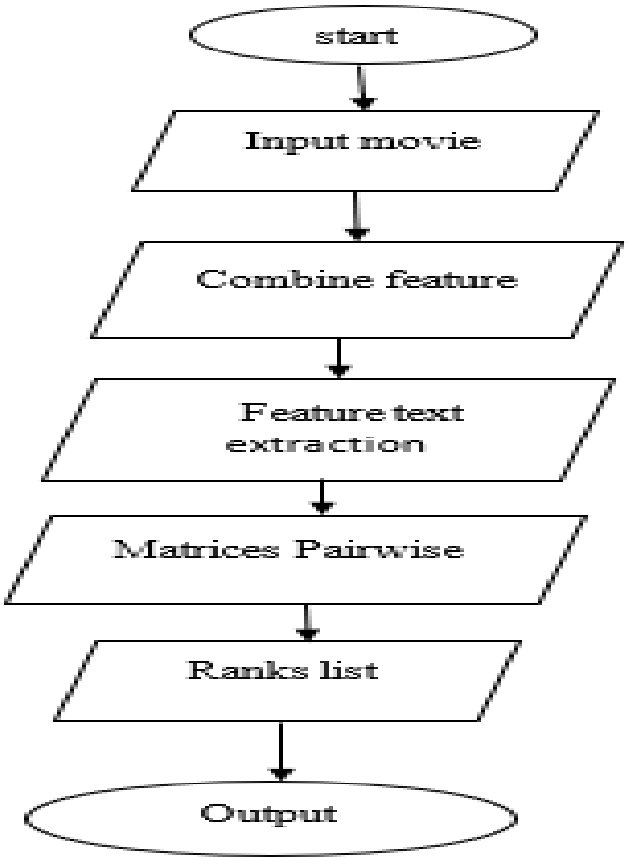

Figure 1.4: Activity Diagram

CHAPTER 4

MODULES

4.1 SYSTEM STUDY

A recommendation engine is a system that suggests products, services, information to users based on analysis of data. Notwithstanding, the recommendation can derive from a variety of factors such as the history of the user and the behaviour of similar users.

Recommendation systems are quickly becoming the primary way for users to expose to the whole digital world through the lens of their experiences, behaviours, preferences and interests. And in a world of information density and product overload, a recommendation engine provides an efficient way for companies to provide consumers with personalised information and solutions.

4.1.1 BENEFITS

A recommendation engine can significantly boost revenues, Click-Through Rates (CTRs), conversions, and other essential metrics. It can have positive effects on the user experience, thus translating to higher customer satisfaction and retention.

Let’s take Netflix as an example. Instead of having to browse through thousands of box sets and movie titles, Netflix presents you with a much narrower selection of items that you are likely to enjoy. This capability saves you time and delivers a better user experience. With this function, Netflix achieved lower cancellation rates, saving the company around a billion dollars a year.

Although recommendation systems have been used for almost 20 years by companies like Amazon, it has been proliferated to other industries such as finance and travel during the last few years.

4.1.2 DIFFERENT TYPES

The most common types of recommendation systems are CONTENT-BASED and COLLABORATIVE FILTERING recommendation systems. In collaborative filtering, the behaviour of a group of users is used to make recommendations to other users. The recommendation is based on the preference of other users. A simple example would be recommending a movie to a user based on the fact that their friend liked the movie. There are two types of collaborative models MEMORY-BASED methods and MODEL-BASED methods. The advantage of memory-based techniques is that they are simple to implement and the resulting recommendations are often easy to explain. They are divided into two: 1

- **User-based collaborative filtering**: In this model, products are recommended to a user based on the fact that the products have been liked by users similar to the user. For example, if Derrick and Dennis like the same movies and a new movie come out that Derick like, then we can recommend that movie to Dennis because Derrick and Dennis seem to like the same movies.

- **Item-based collaborative filtering**: These systems identify similar items based on users' previous ratings. if users A, B and C gave a 5-star rating to books X and Y then when a user D buys book Y they also get a recommendation to purchase book X because the system identifies book X and Y as similar based on the ratings of users A, B and C.

Model-based methods are based on Matrix Factorization and are better at dealing with sparsity. They are developed using data mining, machine learning algorithms to predict users' rating of unrated items. In this approach techniques such as dimensionality reduction are used to improve accuracy. Examples of such model-based methods include Decision trees, Rule-based Model, Bayesian Model, and latent factor models.

- **Content-based systems** use metadata such as genre, producer, actor, musician to recommend items say movies or music. Such a recommendation would be for instance recommending Infinity War that featured Vin Diesel because someone watched and liked The Fate of the Furious. Similarly, you can get music recommendations from certain artists because you liked their music. Content-based systems are based on the idea that if you liked a certain item you are most likely to like something that is similar to it.

4.1.3 CHALLENGES A RECOMMENDATION SYSTEM FACE

1. Sparsity of data. Data sets filled with rows and rows of values that contain blanks or zero values. So finding ways to use denser parts of the data set and those with information is critical.
2. Latent association. Labelling is imperfect. Same products with different labelling can be ignored or incorrectly consumed, meaning that the information does not get incorporated correctly.
3. Scalability. The traditional approach has become overwhelmed by the multiplicity of products and clients. This becomes a challenge as data sets widen and can lead to performance reduction.

4.2 DATA PRE-PROCESSING

For k-NN-based model, the underlying dataset ml-100k from the Surprise Python sci-unit was used. Shock may be a tight call in any case, to search out out regarding recommendation frameworks. It's acceptable for building and examining recommendation frameworks that manage unequivocal rating data.

4.3 MODEL BUILDING

Information is an element into a seventy fifth train take a look at and twenty fifth holdout take a look at. Grid Search CV completed over five - overlap, is employed to find the most effective arrangement of closeness live setup (sim_options) for the forecast calculation. It utilizes the truth measurements because the premise to get completely different mixes of sim options, over a cross-approval system.

4.4 DATA SET USED

we are using the Movie Lens Data Set. This dataset was put together by the Group lens research group at the University of Minnesota. It contains 1, 10, and 20 million ratings. Movie lens also has a website where you can sign up, contribute reviews and get movie recommendations

4.5 RECOMMENDATION VISUALIZATION:

Visualization of movie recommendation systems helps users and developers understand how recommendations are generated and how they relate to user preferences and behaviors. Effective visualizations can improve the usability and transparency of the system, enhancing user trust and engagement. Below are some common visualization techniques used in movie recommendation systems**:**

1. User-Item Interaction Matrix: One of the most fundamental visualizations is the user-item interaction matrix. This matrix displays the relationships between users and movies, with rows representing users, columns representing movies, and values indicating interactions (like ratings or views). A sparse matrix is often used to show that most users have interacted with only a small fraction of the available movies. Visual tools like heatmaps are frequently used to highlight these interactions, making it easier to identify patterns and clusters of similar user preferences

2. Recommendation Lists with Relevance Scores: A common method for presenting recommendations is through a ranked list of movies, which is accompanied by relevance scores that show how well each recommendation matches the user's preferences. These lists can be visually enhanced with tags, thumbnails, or genres to provide a more intuitive interface. The top recommendations may be displayed at the top, with filtering options like genres, ratings, or release dates available for further customization

3. Collaborative Filtering Graphs: Collaborative filtering, especially in its user-based or item-based form, can be visualized using graphs or networks. Users and items are represented as nodes, and edges between them represent interactions or similarities. Graph-based models, such as Graph Neural Networks (GNNs), often utilize these networks to show how movies (nodes) are connected to each other through shared attributes or user interactions. Interactive network visualizations can allow users to explore the relationships between different movies or between users with similar viewing habits

4. Feature-based Clustering: Another visualization technique is the clustering of movies based on shared features. Using algorithms like k-means clustering or dimensionality reduction

techniques (e.g., PCA or t-SNE), movies can be grouped into clusters based on similarities in genre, director, or viewer ratings. These clusters can be visualized as scatter plots, allowing users to see the relationships between different movies. For example, if a user enjoys action movies, they could visually explore a cluster of similar action titles

5. Personalized Movie Recommendation Dashboard: For a more dynamic user experience, platforms like Netflix or Amazon Prime Video often use personalized recommendation dashboards. These dashboards provide personalized rows of movie recommendations, often organized by categories such as "Trending for You," "Because You Watched," or "Recommended for You." These rows use visually appealing elements like movie thumbnails, rating information, and progress indicators to make it easy for users to find content suited to their preferences

6. Recommendation Confidence and Explanations: Some advanced systems go a step further by visualizing the confidence of the recommendations. For example, a visualization may include a confidence bar or score alongside each recommended movie, indicating how confident the system is that the user will like it. Additionally, systems that provide explanations for recommendations often include reasons such as "You watched similar movies," "This movie is highly rated by people with similar tastes," or "Your friends watched this." This These visualization techniques help make the underlying algorithms of recommendation systems more understandable and interactive for users, leading to a better overall experience. Effective visualization not only aids in system transparency but also plays a key role in improving user engagement and satisfaction.

CHAPTER 5

SYSTEM SPECIFICATION

5.1 SOFTWARE REQUIREMENTS

- Operating System: Windows 10/11, Linux, or macOS.
- Programming Language: Python 3.x or later.
- Libraries: Wheel ,Django pandas,gunicorn, pyarrow.
- Database: MySQL/PostgreSQL/SQLite (for storing user and movie data).
- IDE: PyCharm, Jupyter Notebook, or Visual Studio Code.

5.2 HARDWARE REQUIREMENTS

- Processor: Intel i5 or higher.
- RAM: 8 GB or more- Storage: Minimum 500 GB HDD/SSD.
- Graphics Card: Not mandatory.
- Internet Connection: Stable high-speed connection for real-time data fetching and updates .

5.1. 1. Wheel

Wheel is a packaging format for Python that helps with faster and more efficient installation of libraries and dependencies. Unlike traditional source distribution formats, wheel allows Python packages to be installed without needing to compile the code. This is especially useful in production environments where you want to ensure that the necessary libraries for your movie recommendation system are installed quickly and reliably. When developing a movie recommendation system, you may be using several third-party libraries for machine learning, web frameworks, or data processing. With wheel, you avoid potential issues with compilation, making the setup of your development and production environments smoother. It ensures that your system, whether for data collection, user interactions, or movie recommendations, is easily deployable on various platforms.

5.1.2 . Django

Django is a high-level Python web framework that facilitates the rapid development of secure and maintainable web applications. For a movie recommendation system, Django can be used to build the backend of the application where users interact with the system. It provides an easy-to-use ORM (Object-Relational Mapping) to interact with the database, allowing you to store and retrieve data such as user preferences, movie ratings, and metadata (e.g., genres, release dates). Django also provides built-in tools for user authentication and session management, which are important for personalizing movie recommendations based on users' historical data. Additionally, Django allows the creation of RESTful APIs to deliver recommendations to users, providing a seamless experience for both the web and mobile applications.

5.1.3 PANDAS

Pandas is a powerful data manipulation and analysis library in Python that is widely used for handling structured data. In the context of a movie recommendation system, Pandas is essential for data preprocessing, cleaning, and analysis. Movie recommendation systems rely on large datasets, such as user ratings, reviews, and movie metadata. Pandas helps process these datasets by providing flexible data structures like DataFrames, which are ideal for handling and transforming data. For example, you can use Pandas to clean missing or invalid data, normalize ratings, and merge different sources of data (e.g., movie genres and user preferences).

5.1.4. Gunicorn:

Gunicorn (Green Unicorn) is a Python WSGI HTTP server used to serve Python web applications in production. When developing a movie recommendation system using Django, Gunicorn is a crucial tool for deployment. It acts as the interface between the web server (e.g., Nginx or Apache) and the Django application, handling HTTP requests from users. Gunicorn is designed to handle multiple requests simultaneously by forking multiple worker processes, ensuring the application can scale effectively under heavy traffic. In a movie recommendation system, where users might interact with the system to get personalized movie suggestions, Gunicorn ensures that requests for movie data or recommendations are processed swiftly, even when the system experiences high user demand. It is lightweight, fast, and robust, making it an excellent choice for production environments.

5.1.5 pyarrow:

pyarrow is a Python library that provides tools for high-performance data processing and interoperability with Apache Arrow, a cross-language development platform for in-memory data. pyarrow enables fast reading and writing of data in various formats, such as Parquet and Feather, and supports efficient data transfers between Python and other systems. In a movie recommendation system, pyarrow can be particularly useful for handling large datasets that need to be processed and analyzed quickly. Additionally, its support for Apache Arrow’s memory format allows for fast, zero-copy data transfers, which is ideal for optimizing performance in recommendation algorithms that rely on large-scale data processing..

CHAPTER 6

METHODOLOGY

6.1AIM OF THE PROJECT

To implement a recommendation for movies, based on the content of providing the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly.

6.2 SYSTEM REQUIREMENTS

6.2.1 SOFTWARE REQUIREMENTS

Operating system : Windows 7 and above (64-bit).

Python , wheel ,Django pandas,gunicorn,whitenoise,fastparquet,pyarrow.

6.2.2HARDWARE REQUIREMENTS

Hard disk : 80GB or more

Ram : 70Mb or more

Processor : Intel Core Duo 2.0 GHz or more

6.3 OVERVIEW OF THE PLATFORM

6.3.1 Python :

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.
- Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End of Life date of 9 January 1, 2020 has been established for Python 2, after which time it will no longer be maintained.
- Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.
- It is possible to write Python in an Integrated Development Environment,such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses. Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose. Python is Interpreted Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.
- This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.
- One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some 10 applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
- In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.
- For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.
- Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.
- Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

6.3.2 Collaborative Filtering

Collaborative filtering is a technique used by recommendation system. Collaborative filtering has two senses, a narrow one and a more general one.

In the newer, narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly 8 chosen person.

For example, a collaborative filtering recommendation system for television tastes could make predictions about which television show a user should like given a partial list of that user's tastes (likes or dislikes). Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the simpler approach of giving an average (non-specific) score for each item of interest, for example based on its number of votes.

In the more general sense, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets. Collaborative filtering methods have been applied to many different kinds of data including: sensing and monitoring data, such as in mineral exploration, environmental sensing over large areas or multiple sensors; financial data, such as financial service institutions that integrate many financial sources; or in electronic commerce and web applications where the focus is on user data, etc. The remainder of this discussion focuses on collaborative filtering for user data, although some of the methods and approaches may apply to the other major applications as well.

The growth of the internet has made it much more difficult to effectively extract useful information from all the available online information. The overwhelming amount of data necessitates mechanisms for efficient information filtering Collaborative filtering is one of the techniques used for dealing with this problem.

The motivation for collaborative filtering comes from the idea that people often get the best recommendations from someone with tastes similar to themselves. Collaborative filtering encompasses techniques for matching people with similar interests and making recommendations on this basis.

Collaborative filtering algorithms often require (1) users' active participation, (2) an easy way to represent users' interests, and (3) algorithms that are able to match people with similar interests.

Typically, the workflow of a collaborative filtering system is:

1. A user expresses his or her preferences by rating items (e.g. books, movies or CDs) of the system. These ratings can be viewed as an approximate representation of the user's interest in the corresponding domain.
2. The system matches this user's ratings against other users' and finds the people with most "similar" tastes.
3.With similar users, the system recommends items that the similar users have rated highly but not yet being rated by this user (presumably the absence of rating is often considered as the unfamiliarity of an item)

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

In the last few decades, recommendation systems have been used, among the many available solutions, in order to mitigate information and cognitive overload problem by suggesting related and relevant items to the users. In this regards, numerous advances have been made to get a high-quality and fine-tuned recommendation system. Nevertheless, designers face several prominent issues and challenges. Although, researchers have been working to cope with these issues and have devised solutions that somehow and up to some extent try to resolve these issues, however we need much to do in order to get to the desired goal. In this research article, we focused on these prominent issues and challenges, discussed what has been done to mitigate these issues, and what needs to be done in the form of different research opportunities and guidelines that can be followed in coping with at least problems like latency, sparsity, context-awareness, grey sheep and cold-start problem.

Overall flaws can be removed by the hybrid based collaborative filtering with two or more examination techniques are combined to gain the better performance with the less possibilities of drawback of this system. In general in case of hybrid filtering techniques the collaborative filtering technique is combined with some other type of filtering technique to avoid the ramp up problem and thus it outperforms the the major drawbacks of the system in case if we prefer to use single content based or collaborative filtering technique. While we can say that collaborative filtering technique stands good only in terms of the quality perspective but when it comes to both qualitative and quantitative achievement of the result will prefer hybrid filtering technique where the all flaws. In the end hybrid system stands alone the better performer for Recommending movies to the users of different taste, choices or similarities.

7.2 FUTURE ENHANCEMENT

1.) In case of content based filtering method we can look up on the cast and crew also where we have only considered the genre and also we can see at the movies are compatible or not.

2.) Comparison of collaborative filtering based approaches and different kind of similarity measurements would be a good one for the recommender system

3.) We can use matrix factorization for calculating the number of factors involved.

4.) We can also apply deep learning techniques to for the the enhance the recommender system and optimising the efficiency of the system.

5.) We can work on different areas such as video some books aur even recommending some songs to the users of the mobile phones based on the platforms of the different apps available on the Play Store

6.) Various techniques such as clustering classification can be used to get the better version of our recommender system which for the the enhance the accuracy of the overall model.

APPENDIX – 1

SOURCE CODE

view.py

```
from django.shortcuts import render
import pandas as pd
import pyarrow as pa

movies_data = pd.read_parquet("static/top_2k_movie_data.parquet")
titles = movies_data['title']
titles_list = titles.to_list()
def get_recommendations(movie_id_from_db,movie_db):
            try:
sim_scores = list(enumerate(movie_db[movie_id_from_db]))
sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
sim_scores = sim_scores[1:15] ## get top 15 Recommendations
             movie_indices = [i[0] for i in sim_scores]
             output = movies_data.iloc[movie_indices]
           output.reset_index(inplace=True, drop=True)
                response = []
                 for i in range(len(output)):
                            response.append({
                                        'movie_title':output['title'].iloc[i],
                              'movie_release_date':output['release_date'].iloc[i],
                                 'movie_director':output['main_director'].iloc[i],
                                  'google_link':"https://www.google.com/search?q=" +
                             '+'.join(output['title'].iloc[i].strip().split()) + " (" +
                                 output['release_date'].iloc[i].split("")[0]+")"
      })
     return response
    except Exception as e:
        print("error: ",e)
        return []
def main(request):
```

```
global titles_list, model
if request.method == 'GET':
        return render(
                request,
                  'recommender/index.html',
                  {
                   'all_movie_names':titles_list,
                   'input_provided':'',
                   'movie_found':'',
                   'recomendation_found':'',
                      'recommended_movies':[],
                  'input_movie_name':''
                   }
    )
if request.method == 'POST':
        data = request.POST
       movie_name = data.get('movie_name') ## get movie name from the frontend input
                                                field


            if movie_name in titles_list:
                    idx = titles_list.index(movie_name)
            else:
                    return render(
                            request,
                              'recommender/index.html',
                            {
                               'all_movie_names':titles_list,
                                    'input_provided':'yes',
                                       'movie_found':'',
                                   'recomendation_found':'',
                                         'recommended_movies':[],
                                 'input_movie_name':movie_name
                            }
            )
```

```
            model = pa.parquet.read_table('static/demo_model.parquet').to_pandas()
            final_recommendations = get_recommendations(idx,model)
if final_recommendations:
            return render(
                        request,
                          'recommender/result.html',
                                {
                                            'all_movie_names':titles_list,
                                                'input_provided':'yes',
                                                 'movie_found':'yes',
                                             'recomendation_found':'yes',
                                    'recommended_movies':final_recommendations,
                                          'input_movie_name':movie_name
                                        }
   )
else:
       return render(
                 request,
              'recommender/index.html',
                      {
                                        'all_movie_names':titles_list,
                                            'input_provided':'yes',
                                                'movie_found':'',
                                           'recomendation_found':'',
                                          'recommended_movies':[],
                                       'input_movie_name':movie_name
                                         }
   )
```

APPENDIX – 2

SCREENSHOTS

Sample Output

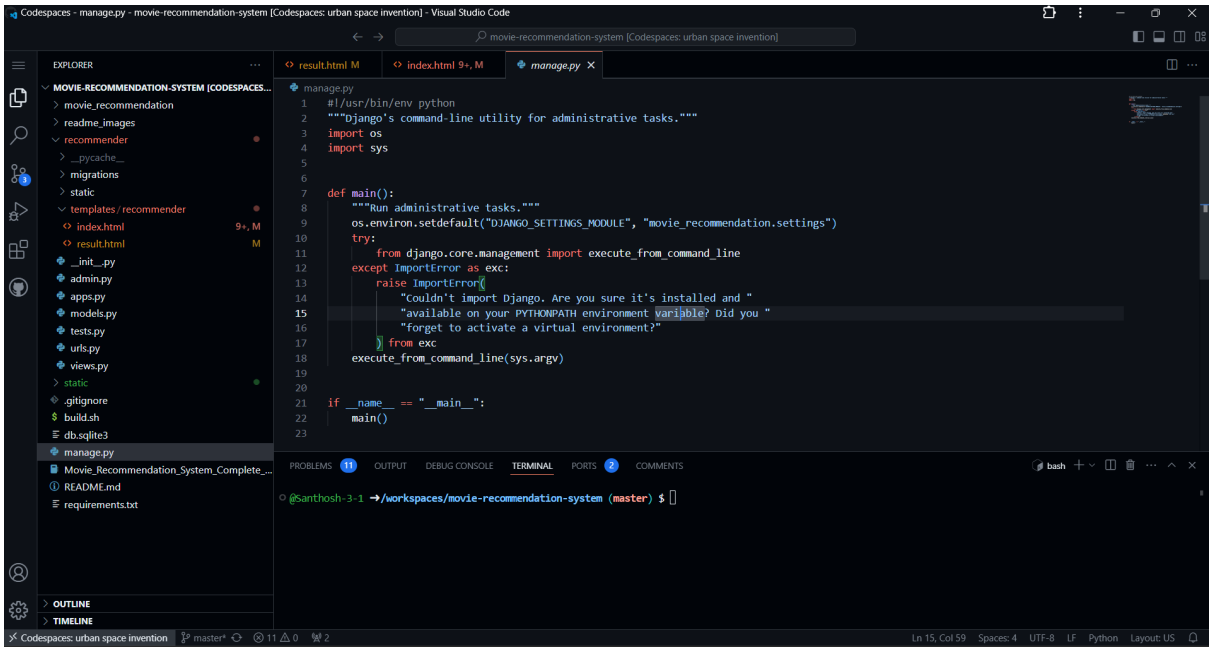

Figure 2.1: manage.py

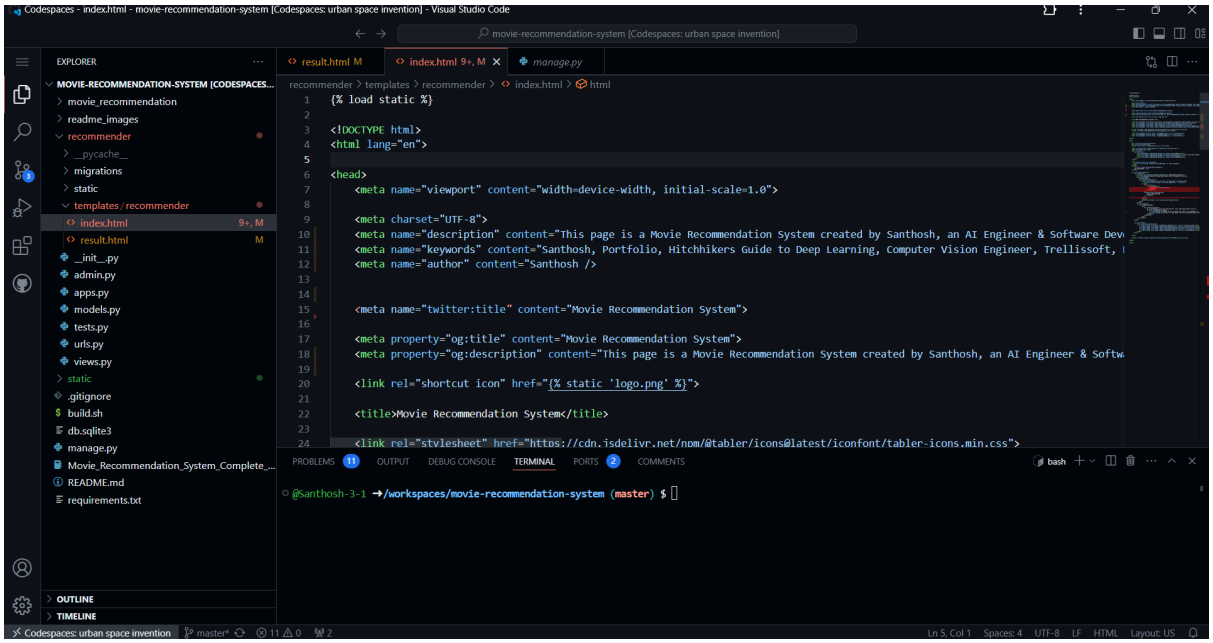

Figure 2.2: index.html

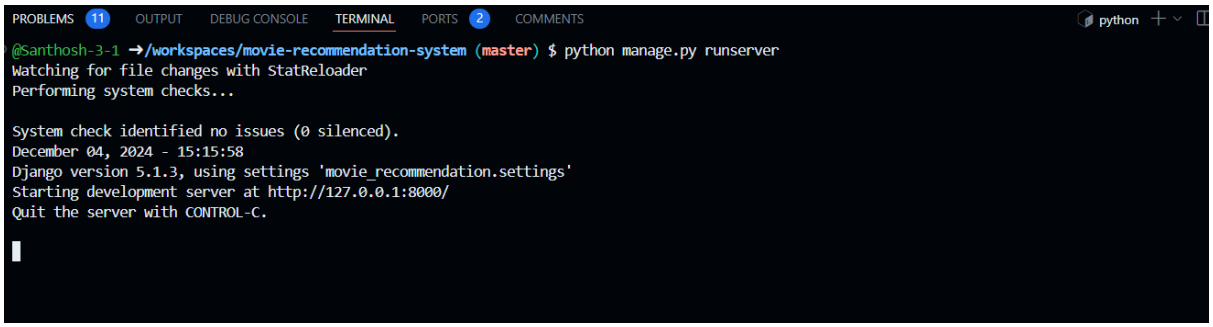

Figure 2.3: Execution of Program

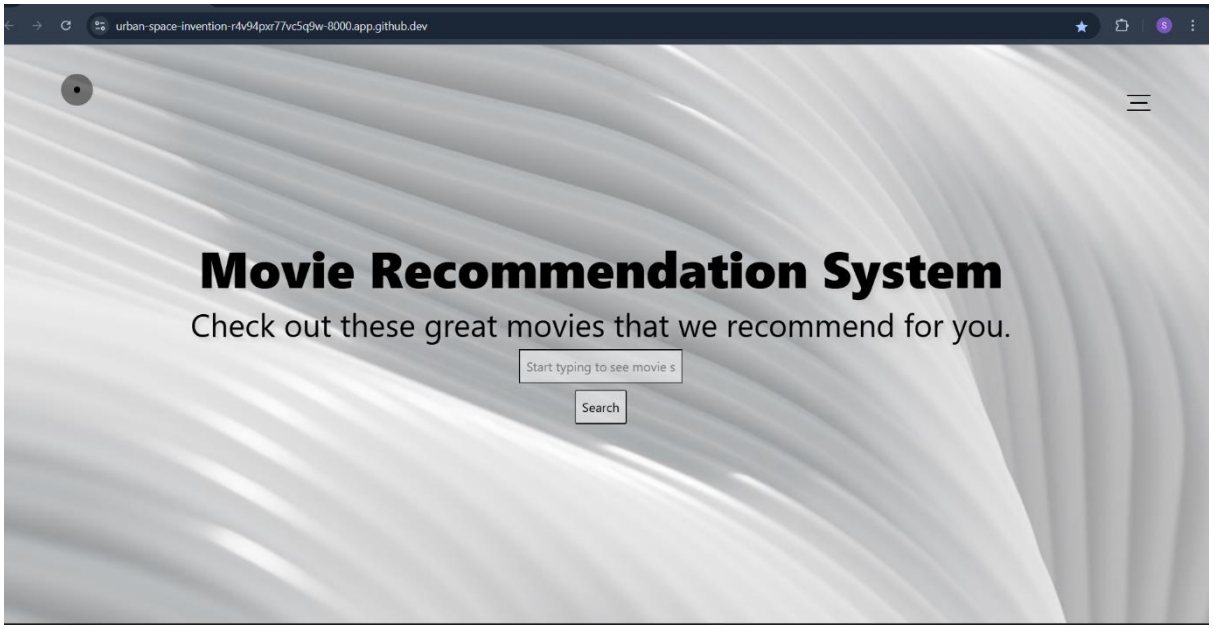

Figure 2.4.home page

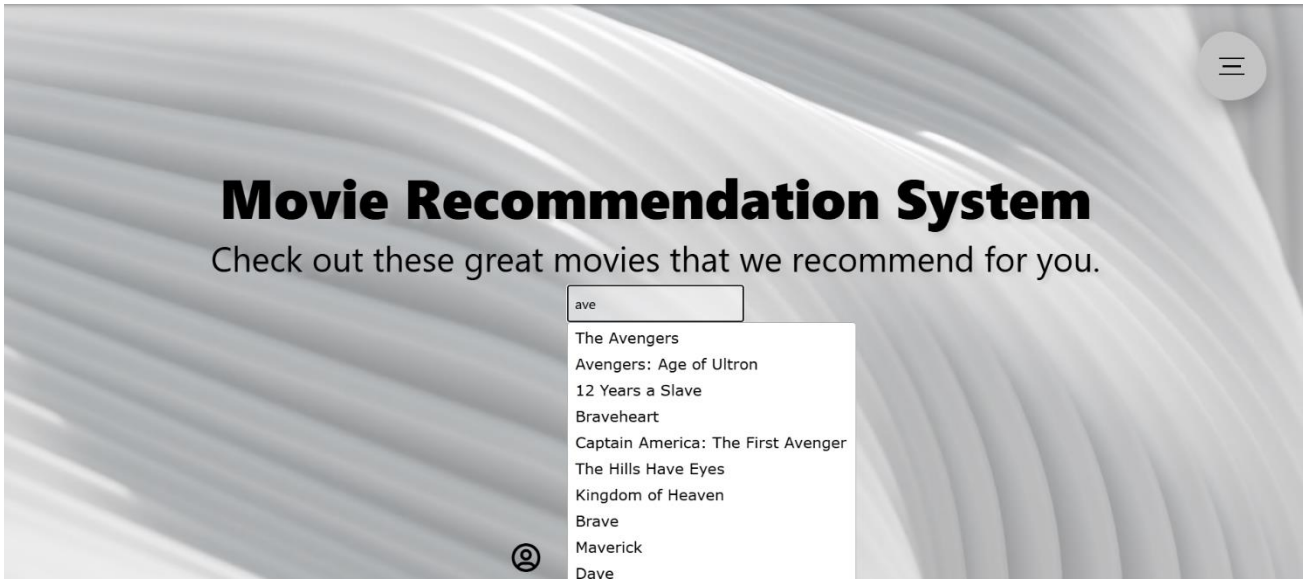

Figure 2.5.input search

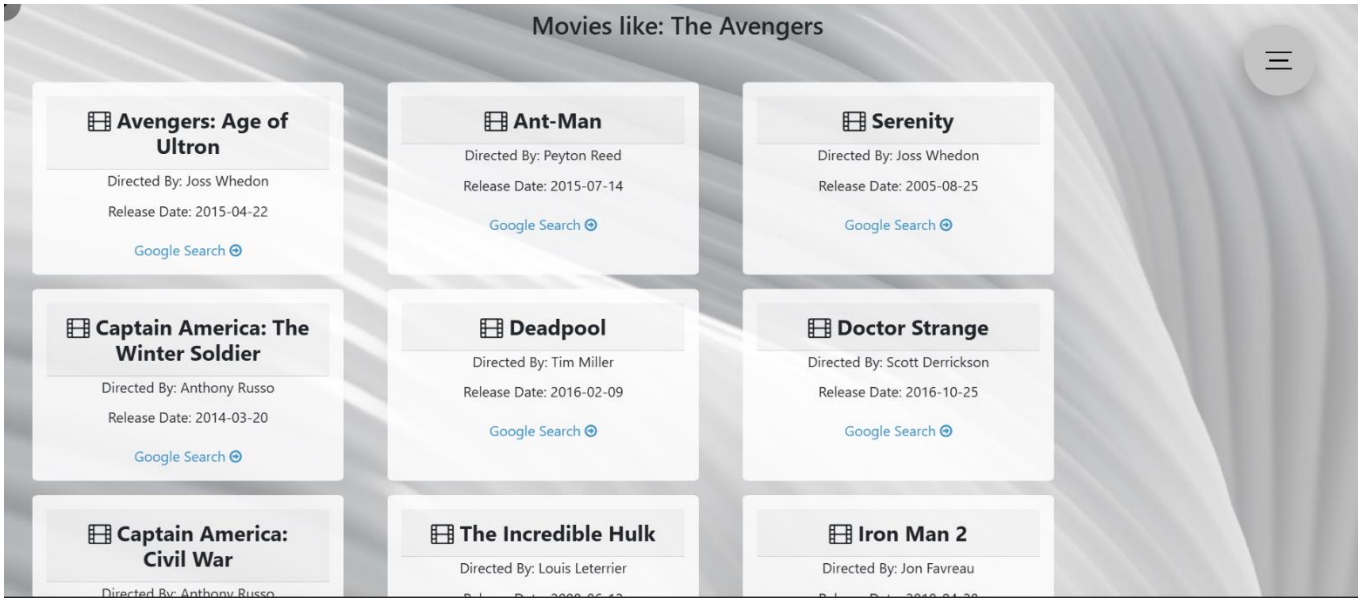

Figure 2.6.movie recommendation

REFERENCES

1. Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) Recommender systems survey. Knowl Based Syst 46:109–132. https://doi.org/10.1016/j.knosys.2013.03.012

2. Cui, Bei-Bei. (2017). Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm. ITM Web of Conferences. 12. 04008. 10.1051/itmconf/20171204008.

3. Fadhel Aljunid, Mohammed & D H, Manjaiah. (2018). Movie Recommender System Based on Collaborative Filtering Using Apache Spark.

4. Miryala, Goutham & Gomes, Rahul & Dayananda, Karanam. (2017). COMPARATIVE ANALYSIS OF MOVIE RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING IN SPARK ENGINE. Journal of Global Research in Computer Science. 8. 10-14.

5. Banerjee, Anurag & Basu, Tanmay. (2018). Yet Another Weighting Scheme for Collaborative Filtering Towards Effective Movie Recommendation.

6. Zhao, Zhi-Dan & Shang, Ming Sheng. (2010). UserBased Collaborative-Filtering Recommendation Algorithms on Hadoop. 3rd International Conference on Knowledge Discovery and Data Mining, WKDD 2010. 478- 481.

7. A. V. Dev and A. Mohan, "Recommendation system for big data applications based on the set similarity of user preferences," 2016 International Conference on Next Generation Intelligent Systems (ICNGIS), Kottayam, 2016, pp. 1-6. DOI: 10.1109/ICNGIS.2016.7854058

8. Subramaniyaswamy, V., Logesh, R., Chandrashekhar, M., Challa, A. and Vijayakumar, V. (2017) ‘A personalized movie recommendation system based on collaborative filtering,’ Int. J. HighPerformance Computing and Networking, Vol. 10, Nos. 1/2, pp.54–63. 3

9 .Thakkar, Priyank & Varma (Thakkar), Krunal & Ukani, Vijay & Mankad, Sapan & Tanwar, Sudeep. (2019). Combining UserBased and Item-Based Collaborative Filtering Using Machine Learning: Proceedings of ICTIS 2018, Volume 2. 10.1007/978-981-13-1747- 7_17

10.GroupLens, Movielens Data, 2019 , http://grouplens.org/datasets/movielens/