A Project Report on

Movie Recommendation System

Submitted in the Partial Fulfilment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

Vaibhav Negi(22BTCSE0319)

Vineet Saini(22BTCSE0035)

Upendra Kr. Yadav(22BTCSE0297)

Under the supervison of

Mr. Yudhveer Singh Moudgli

Assistant Professor

Submitted to the

Department of Computer Science and Engineering

DEV BHOOMI UTTARAKHAND UNIVERSITY

DEHRADUN

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this project titled, "**Movie Recommendation System"** submitted by me in the partial fulfilment of the requirement of the award of the degree of **Bachelor of Technology (B.Tech.)** submitted in the Department of **Computer Science & Engineering, Dev Bhoomi Uttarakhand University, Dehradun**, is an authentic record of my thesis carried out under the guidance of **Mr. Yudhveer Singh Moudgli, Technical Educator, Department of Computer Science and Engineering under SoEC, Dev Bhoomi Uttarakhand University, Dehradun.**

Date:

Vaibhav Negi(22BTCSE0319)

Vineet Saini(22BTCSE0035)

Upendra Kumar Yadav(22BTCSE0297)

B. Tech(CSE)

Dev Bhoomi Uttarakhand University

Approved By

Mr. Dhajvir Singh Rai

Head of the Department
(Computer Science & Engineering)
Dev Bhoomi Uttarakhand University

CERTIFICATE

It is to certify that the project entitled **"File Sharing Application"** which is being submitted by **Mukesh Pandey, Alok Sati, Bhanu Pratap Gusain** to the Dev Bhoomi Uttarakhand University Dehradun, in the fulfilment of the requirement for the award of the degree of **Bachelor of Technology (B. Tech.)** is a record of bonafide research work carriedout by him under my guidance and supervision. The matter presented in this report has not beensubmittedeither in part or full to any University or Institute for award of any degree.

Mr. Yudhveer Singh Moudgli
Department of Computer Science and Engg.
Dev Bhoomi Uttarakhand University , Dehradun
(Uttarakhand) INDIA

ABSTRACT

In today's digital era, with the exponential growth of online content, selecting relevant and engaging movies has become a challenging task for users. To solve this issue, this project presents a Content-Based Movie Recommendation System developed using Python, integrated with Streamlit for UI and Power BI for data visualisation. The system is designed to provide personalised movie recommendations based on the content metadata of movies such as genres, cast, crew, overview, and keywords.

The project uses Natural Language Processing (NLP) techniques to process and combine textual features, which are then converted into numerical vectors using TF-IDF (Term Frequency–Inverse Document Frequency). These vectors are compared using Cosine Similarity, which helps in identifying the most similar movies to a selected one. The top 5 similar movies are then recommended to the user through an interactive Streamlit web interface.

In addition to the recommendation system, the project includes an in-depth data analytics dashboard built in Power BI, which uncovers hidden trends from the movie dataset. Insights like genre popularity, blockbuster vs flop analysis, vote vs popularity patterns, and clustering of movies using machine learning help understand audience preferences and business performance.

This hybrid approach of integrating machine learning with business intelligence tools adds value by making the recommendation process transparent, insightful, and engaging. The project not only enhances user experience by helping them discover movies they are more likely to enjoy but also offers powerful insights for producers, streaming platforms, and data analysts to make informed decisions.

Hence, this project combines the power of machine learning, interactive web development, and data visualisation to deliver a complete, efficient, and scalable movie recommendation solution.

ACKNOWLEDGEMENT

At this ecstatic time of presenting this dissertation, first, the author bows to almighty God for blessing with enough patience and strength to go through this challenging phase of life.

We would like to express a deep sense of gratitude and thanks to those people who have helped me in the accomplishment of this report.

First and foremost, we would like to thank our supervisor, **Mr. Yudhveer Singh Moudgli** for their expertise, guidance, enthusiasm, and patience. These were invaluable contributors whose insightful guidance helped to the successful completion of this dissertation and spent many hours patiently answering questions and troubleshooting the problems.

Beyond all this, we would like to give special thanks to our friends for the unbounded affection, constant inspiration, and encouragement. Without their support this research would not have been possible.

Finally, we would like to thank all faculty, college management, administrative and technical staff of **School of Engineering & Computing, Dev Bhoomi Uttarakhand University, Dehradun** for their encouragement, assistance, and friendship throughout our candidature.

LIST OF CONTENT

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

In the era of digital entertainment, movie consumption has drastically increased through online platforms. With thousands of options available to users, choosing the right movie has become an overwhelming experience. To address this growing challenge, intelligent systems such as recommendation engines have emerged to assist users in discovering relevant content. These systems utilise data-driven approaches, including machine learning and natural language processing, to analyse movie features and user preferences.

Our project is focused on building a **content-based movie recommendation system** that analyses movie metadata like genres, cast, keywords, and overviews. By applying **cosine similarity** and **TF-IDF vectorisation**, the system recommends movies that are contextually similar to the one selected by the user. Additionally, we use **Power BI** to build interactive dashboards that visualise movie trends and performance, adding a layer of analytical understanding to the data.

1.2 Problem Statement

Increased availability of movies across OTT platforms has introduced the paradox of choice. Users often spend more time searching than watching. Traditional browsing or keyword-based search is insufficient to cater to diverse user preferences.

There is a lack of intelligent systems that can provide personalised recommendations by understanding the contextual similarity between movies. This results in a poor user experience and underutilisation of high-potential content. The proposed system addresses this issue by offering intelligent, automated, and explainable recommendations using content-based filtering techniques, backed by data analytics.

1.3 Project Objective

The primary objective of this project is to develop a movie recommendation system using machine learning techniques and enrich it with Power BI dashboards for in-depth analysis.

Specific Objectives:

- To build a recommendation engine based on content similarity (cast, genre, overview).
- To apply cosine similarity over TF-IDF matrix for ranking similar movies.
- To design a Streamlit-based interactive UI for users to input movie selections.
- To integrate Power BI dashboards for visual analysis of genre trends, clusters, and revenue performance.
- To make data-driven decisions easier for both users and content producers.

1.4 Importance of Secure File Sharing

Although the project is focused on recommendations, it also ensures secure model and dataset handling. The use of **Pickle** for saving pre-trained models and similarity matrices ensures integrity and avoids data tampering. Model sharing and updates can be managed in secure environments. In real-world deployment, secure APIs or cloud-based sharing would be essential to protect intellectual property and user data.

1.5 Cross-Platform Compatibility

The developed application is built using **Python and Streamlit**, which are platform-independent tools. It can run on **Windows, macOS, or Linux** with minimal setup. The use of web-based visualisation through **Power BI service** ensures accessibility from anywhere via browser. This cross-platform nature makes it deployable in different environments, including cloud platforms or institutional networks.

1.6 Features of the Application

- Content-Based Movie Recommendation
- Machine Learning with TF-IDF and Cosine Similarity
- Visual Analytics using Power BI

- Secure Model Sharing via Pickle
- Streamlit-Based Interactive UI
- Genre & Popularity-Based Dashboards
- Cluster Analysis with KMeans
- Lightweight & Easy to Deploy

  These features not only enhance the user experience but also make the system scalable and analytics-driven.

1.7 Use Case Scenarios

- A user selects a movie like “Inception” and instantly receives 5 similar recommendations.
- A streaming platform analyst uses Power BI dashboards to understand genre-wise performance.
- A content producer studies genre trends and rating distributions before planning a new movie.
- A business analyst explores cluster-based groupings of movies for targeted marketing.

1.8 Conclusion of the Introduction

This project bridges the gap between user experience and content discovery using machine learning and business intelligence. By integrating a recommendation engine with visual analytics, the system empowers users to find relevant content and enables stakeholders to make informed decisions. The system not only enhances personal movie exploration but also serves as a valuable tool for content strategists and data analysts.

CHAPTER 2

TECHNOLOGY USED

The development of a Movie Recommendation System requires a combination of intelligent algorithms, cross-platform support, secure data handling, and effective user interface design. This chapter outlines the key technologies and components that enable the system to function efficiently, ensuring seamless recommendation delivery and data-driven visual insights.

2.1 Cross-Platform Mobile Development

Although the system primarily runs on web-based platforms using Streamlit, it is designed with future expansion into mobile platforms. Tools like React Native or Flutter can be integrated later to bring recommendation access to Android and iOS users. The design logic and backend APIs are built to remain compatible across devices and operating systems.

2.2 Networking and Secure Communication

The application interacts with the TMDB API to fetch movie poster data in real-time. All API requests are handled using the requests library in Python over HTTPS. This ensures secure, encrypted communication. In real-world deployment, further measures such as API key protection, request throttling, and token-based authentication would be used for enhanced security.

2.3 File Handling and Binary Data Processing

Python's built-in libraries like pickle are used for serialising trained data like the movie list DataFrame and similarity matrix. These files are stored in binary format (.pkl) to ensure fast access during recommendations. This file handling mechanism prevents reprocessing and reduces memory usage, leading to a highly optimised system performance.

2.4 Unique Identification and Randomness

The system uses Python’s random library to display random movie posters when no input is selected. This gives a dynamic experience to the user. Unique movie IDs from the dataset ensure that each recommendation and API call is correctly matched with its corresponding metadata, reducing conflict and confusion during selection.

2.5 State Management and Data Flow

In the Streamlit application, the state is maintained across interactions using in-built features like session state, input widgets, and conditional logic. Data flows from the user's selected movie into the similarity engine, and then into visual output — following a unidirectional, clean pipeline. This structure ensures reproducibility and transparency of results.

2.6 UI and UX Libraries

The UI is built using Streamlit, which offers an easy-to-use interface for rendering dropdowns, buttons, images, and layout grids. The Power BI dashboards enhance the user experience further by providing interactive filtering, tooltips, drill-downs, and slicers — making data exploration engaging and intuitive, even for non-technical users.

2.7 Testing, Debugging & Deployment Tools

Testing was conducted manually through Jupyter Notebook and Streamlit. Errors like missing posters or index mismatches were debugged using Python tracebacks. For deployment, Streamlit allows publishing to platforms like Streamlit Cloud or Heroku, and Power BI dashboards can be shared using Power BI Service. These tools ensure efficient development and sharing of the system.

2.8 Platform-Specific Capabilities

This project is platform-independent and can be executed on any system supporting Python and web browsers. However, Power BI's cloud-based features and Streamlit's responsive web UI make it ideal for desktop and tablet viewing. The modularity allows adding platform-specific enhancements later (e.g., push notifications for mobile users).

2.9 Optional Enhancements (Future Scope)

- Integrate Collaborative Filtering with user ratings
- Add Sentiment Analysis on movie reviews
- Expand to mobile app using Flutter or React Native
- Include user authentication and profile saving
- Use Neural Networks (BERT) for context-rich recommendation
- Enable real-time updates and cloud-based deployment

TECHNOLOGY STACK:

FIG. 1

FIG. 2

CHAPTER 3

System Architecture

The architecture of the Movie Recommendation System is modular and scalable, ensuring efficient functioning, maintainability, and smooth user interaction. It combines multiple system components like recommendation logic, communication engines, data flow management, and UI rendering. Each component is designed to interact seamlessly, enabling accurate movie predictions, secure data handling, and dynamic presentation of results to users.

3.1 Overview of System Components

- The system is built using a client-server model with user input handled through Streamlit UI.
- Backend engine is responsible for content-based recommendation logic using ML.
- Data pre-processing, similarity computation, and visualisation modules operate independently.
- Power BI dashboards work as an external analytical tool, pulling cleaned data for reporting.
- Storage of similarity matrix and movie metadata is managed through Pickle serialisation.

3.2 Connection Manager

- Manages communication between Streamlit frontend and Python backend logic.
- Controls API calls (e.g., to TMDB for movie posters).
- Ensures smooth data transfer from user interaction to backend response.
- Handles synchronisation of state between selected input and recommendation results.

3.3 File Transfer Manager

- Handles loading and storing of model files (movie_list.pkl, similarity.pkl).
- Transfers pre-trained files from local or cloud for fast system access.
- Ensures read/write integrity using binary file handling methods.
- Enables lightweight storage and distribution of the model.

3.4 Socket Communication Engine

- While not using traditional sockets, HTTP-based communication with TMDB API acts as data transfer engine.
- Handles movie poster retrieval by sending GET requests based on movie IDs.
- Ensures asynchronous response handling for a smooth user experience.
- Can be replaced by socket or REST API calls in future mobile/web deployments.

3.5 State Management System

- Streamlit's built-in support for session state and widget state helps maintain flow.
- Ensures that the user's movie selection persists across interactions.
- Prevents unnecessary reloading of components (e.g., API call throttling).
- Clean and reproducible pipeline from input → recommendation → poster output.

3.6 UI/UX Components

Designed using Streamlit, which offers pre-built UI components for rapid development.

- Includes:
- Dropdown for movie selection
- Buttons for triggering recommendations

- Columns for displaying posters and names

  Focus on responsive layout using st.columns, st.image, and st.caption.

- Dashboard part (Power BI) includes:
- Slicers, KPIs, charts, scatter plots, and filters for interactive visualisation.

3.7 Error Handling and Recovery Mechanism

Errors like:

- IndexError (for invalid movie titles)
- API request failures
- Missing posters

Are handled using:

- Try-except blocks
- Fallback posters (placeholder image)
- Logging missing entries for future correction

  Future enhancements include:

- Logging system
- Alert messages in frontend for better debugging.

3.8 Security and Integrity

- Data stored locally in .pkl format ensures **model integrity**.
- Pickle files are protected from tampering as they store binary objects.
- External API keys (like TMDB) can be secured using .env files.
- Power BI dashboards are shared via secure organisational workspaces or publish-to-web with access control.

3.9 Performance Considerations

- Pre-computed similarity matrix allows instant recommendations — no real-time ML training required.
- Movie poster calls are lazy-loaded, reducing load time.
- Binary file reading is fast and memory-efficient.
- Power BI dashboards use filtered datasets to ensure responsive loading.
- Limited UI rendering keeps app lightweight.

3.10 Scalability

- The modular design allows:
- Easy swapping of ML models (e.g., TF-IDF → BERT)
- Expansion to collaborative filtering
- Deployment on cloud (AWS, Azure)

Frontend (Streamlit) can be ported to **mobile apps using React Native or Flutter**.

Power BI can scale to handle large datasets from cloud storage or SQL servers.

Model and dashboard updates can be pushed without changing entire architecture.

ACTIVITY DIAGRAM

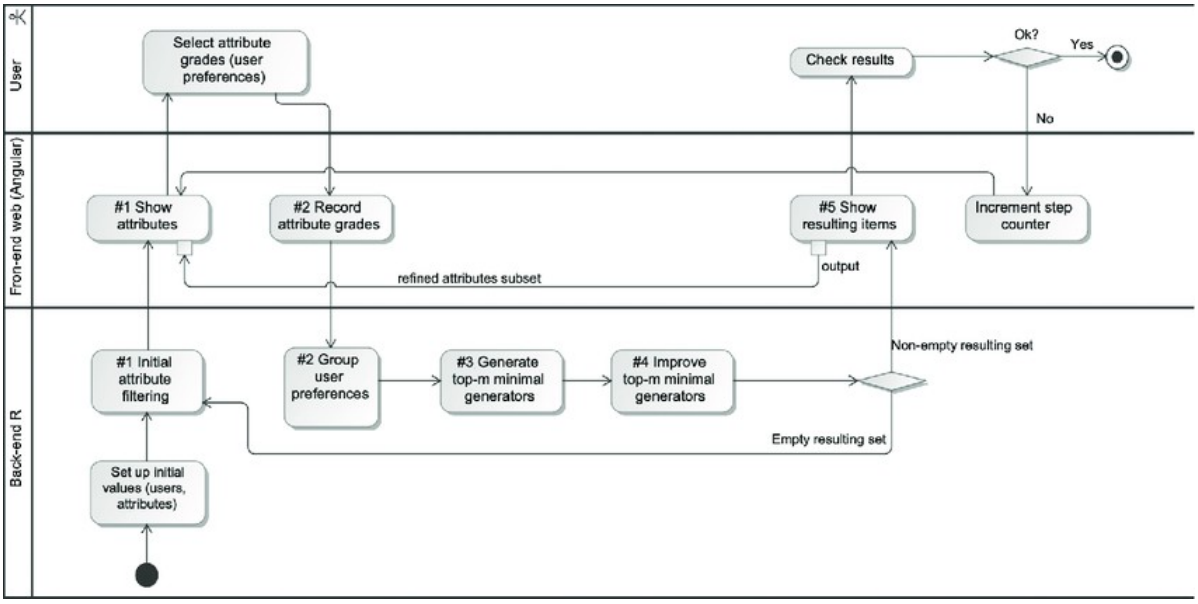

FIG. 3

UML DIAGRAM:

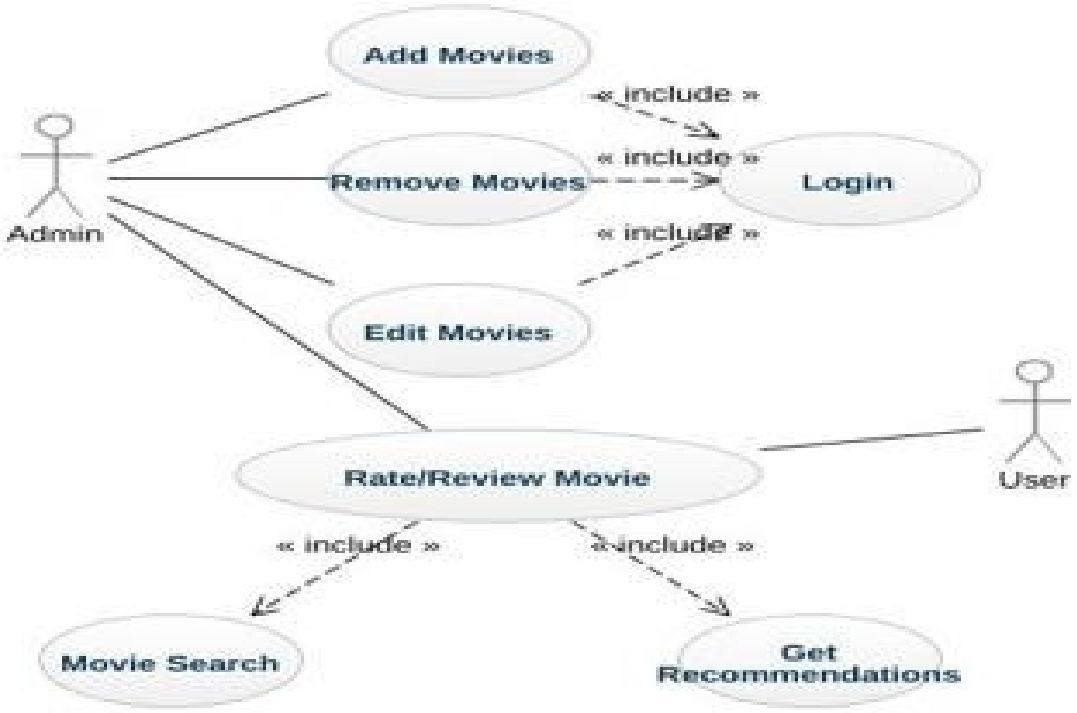

FIG. 4

CHAPTER 4

METHODOLOGY AND DATA FLOW

The methodology of the Movie Recommendation System outlines the complete life cycle of the project from raw dataset collection to final deployment of the recommendation engine and Power BI dashboards. This chapter describes each phase of the project in detail, focusing on data preparation, algorithm development, similarity calculation, user interface design, and integration of visual analytics. The system follows a data-centric approach to deliver intelligent, user-friendly, and insightful movie recommendations.

4.1 Phase 1: Dataset Collection and Understanding

Datasets Used:

- movies_metadata.csv
- movies_credits.csv
- movies_keywords.csv
- movies_ratings.csv
- **These files were sourced from Kaggle and TMDB, containing essential metadata:**
- Titles, genres, overviews, keywords, cast, crew, runtime, revenue, release date, etc.
- **Objective of this phase:**
- Understand the structure of data
- Identify useful columns
- Plan for merging multiple datasets

4.2 Phase 2: Data Preprocessing and Cleaning

- **Null and duplicate removal:**
- Checked for missing values in critical fields (like title, overview, genres, etc.).
- Removed rows with empty or malformed data.
- **List-format extraction:**
- Many columns were in stringified list format ([{"id": 28, "name": "Action"}])
- Used Python's ast.literal_eval() to convert to lists and extract actual names.
- **Tag creation:**
- Combined features from genres, keywords, cast, and crew into a single text field called tags.
- Applied lowercase conversion, removed spaces and punctuation for cleaner analysis.
- **Data Normalisation:**
- Unified spelling variations
- Removed stopwords using NLP
- Converted all relevant text into a single document per movie

4.3 Phase 3: Feature Extraction (Vectorisation)

- **Used CountVectorizer from sklearn.feature_extraction.text:**
- Transformed tags column into numerical vectors
- Limited to top 5000 frequent words
- Removed English stopwords to reduce noise
- **Why CountVectorizer?**
- Lightweight, effective for short documents like movie tags

- Suitable for content-based filtering
- **Resulted in a vector matrix representing each movie as a high-dimensional point in space**

4.4 Phase 4: Similarity Computation using Cosine Similarity

- Applied cosine similarity from sklearn.metrics.pairwise:
- Compared vector of selected movie with all other movie vectors
- Output was a similarity score between 0 and 1
- **Sorted the results in descending order of similarity**
- **Top 5 movies (excluding the input movie) were selected for recommendation**
- **Why Cosine Similarity?**
- Captures contextual closeness rather than absolute counts
- Robust for sparse data and avoids influence of document length

4.5 Phase 5: Streamlit Front-End Development

- **Developed a lightweight and responsive web interface using Streamlit**
- **Key components:**
- Header and title
- Dropdown menu to select movie
- Button to trigger recommendation
- Dynamic display of top 5 similar movies with posters
- **Posters fetched in real-time from TMDB API using movie IDs**
- **Fallback mechanism for missing poster (default image placeholder)**
- **User Experience:**
- Instant results

- No need to refresh page
- Visually engaging layout

4.6 Phase 6: Power BI Integration and Dashboarding

- Exported preprocessed movie dataset to CSV format
- Loaded into **Power BI** for building interactive dashboards
- **Dashboards Created:**

1. **Genre Trends & Ratings Dashboard**
   - Genre-wise movie count, popularity vs rating scatter plot
   - KPIs: total votes, average rating
   - Filters: year, genre
2. **Blockbuster vs Flop Analysis**
   - Bubble chart: budget vs revenue
   - Cards: Total movies, blockbusters, flops, average revenue
   - Top 10 highest-grossing movies bar chart
3. **Clustering with K-Means**
   - Added cluster labels using unsupervised learning
   - Pie chart, cluster-wise scatter plot
   - Cluster-specific table for movie grouping

- **Purpose:**
- Understand user preferences
- Visualise ROI-based performance
- Reveal hidden patterns through clustering

4.7 Phase 7: Data Flow Diagram (DFD)

Level 0 DFD:

- User inputs movie → System fetches vector → Calculates similarity → Returns top 5

Level 1 DFD:

- Input: Movie Title
- Process 1: Fetch vector representation
- Process 2: Compute similarity
- Process 3: Fetch movie poster (from TMDB)
- Output: Display recommended movies with posters

4.8 Summary of Methodology

This methodology is a complete data-driven pipeline combining multiple tools and technologies for real-time recommendation and interactive insights. It follows the principles of clean data engineering, explainable ML models, and seamless frontend design. From vectorisation to

visualisation, every step contributes to a scalable and intelligent system that can be enhanced further with additional features.

CHAPTER 5

IMPLEMENTATION AND SCREENSHOTS

Testing is a crucial phase of the development process to ensure the application is working as expected and that it meets the desired functional and non-functional requirements. For the **File Sharing Application** built with **React Native** and **TCP Socket Communication**, various levels of testing were performed to verify the correctness, performance, and security of the system. The following sections outline the testing strategies, types of tests, and tools used in the project.

5.1 Environment Setup

- The project was implemented using Python 3.10+, with libraries such as pandas, numpy, sklearn, pickle, and streamlit.
- The environment was set up using Jupyter Notebook for initial model development and Streamlit for building the user interface.
- Visualisation and analysis were carried out using Power BI Desktop.

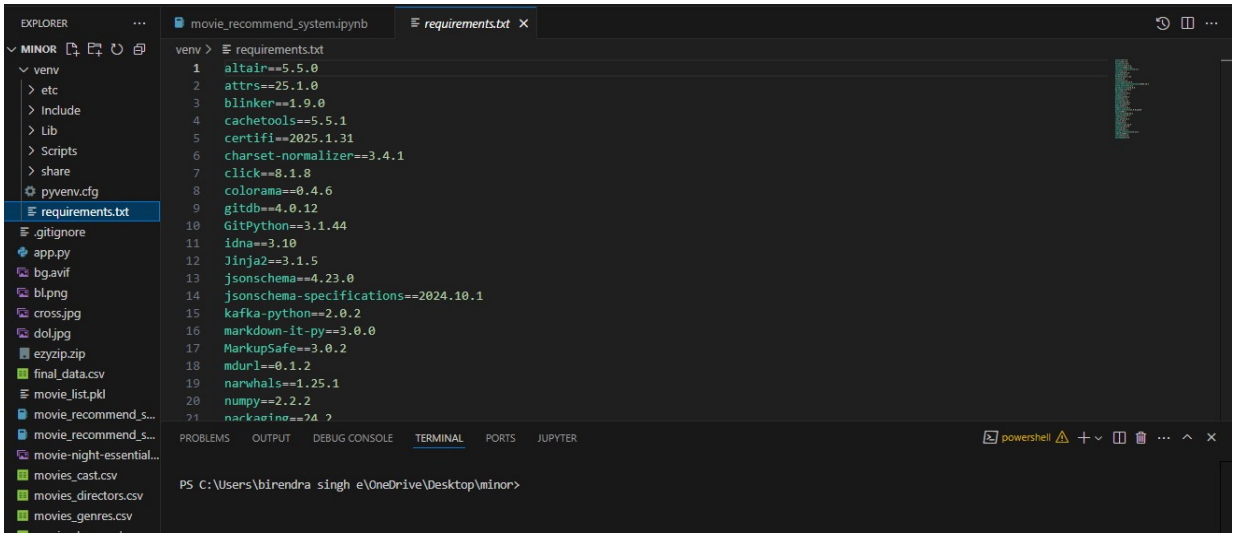

FIG. 5.1

5.2 Data Cleaning and Preprocessing

- Data from multiple CSV files was loaded and cleaned using Pandas.
- Columns such as genres, cast, keywords, and crew were parsed using Python's ast module.
- All necessary features were combined into a new column tags for vectorisation.

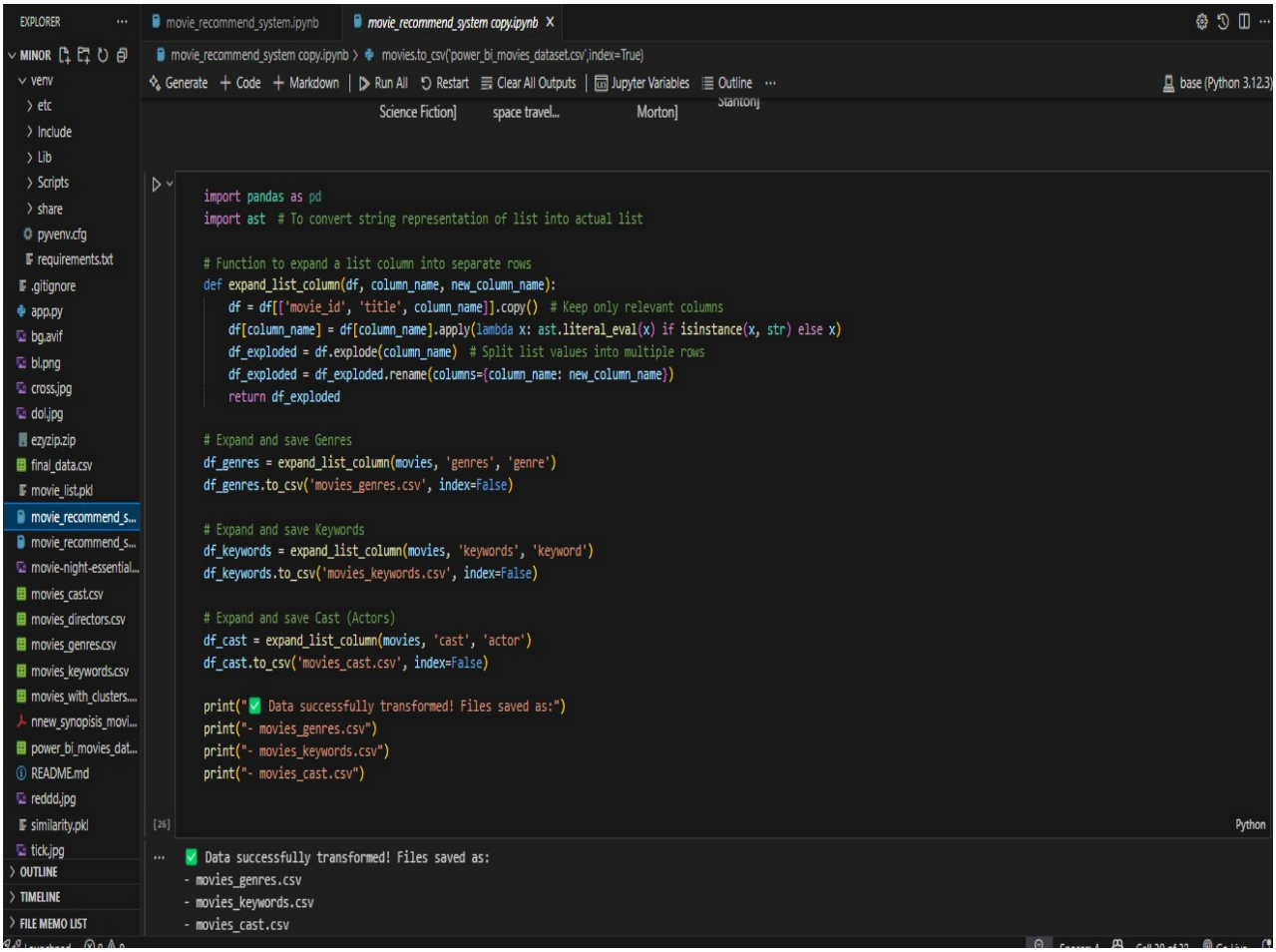

FIG. 5.2

5.3 Feature Engineering and Vectorisation

- Used CountVectorizer to convert text-based tags into numerical vectors.
- Limited vocabulary to top 5000 features and removed English stop words.
- Generated a similarity matrix using Cosine Similarity.

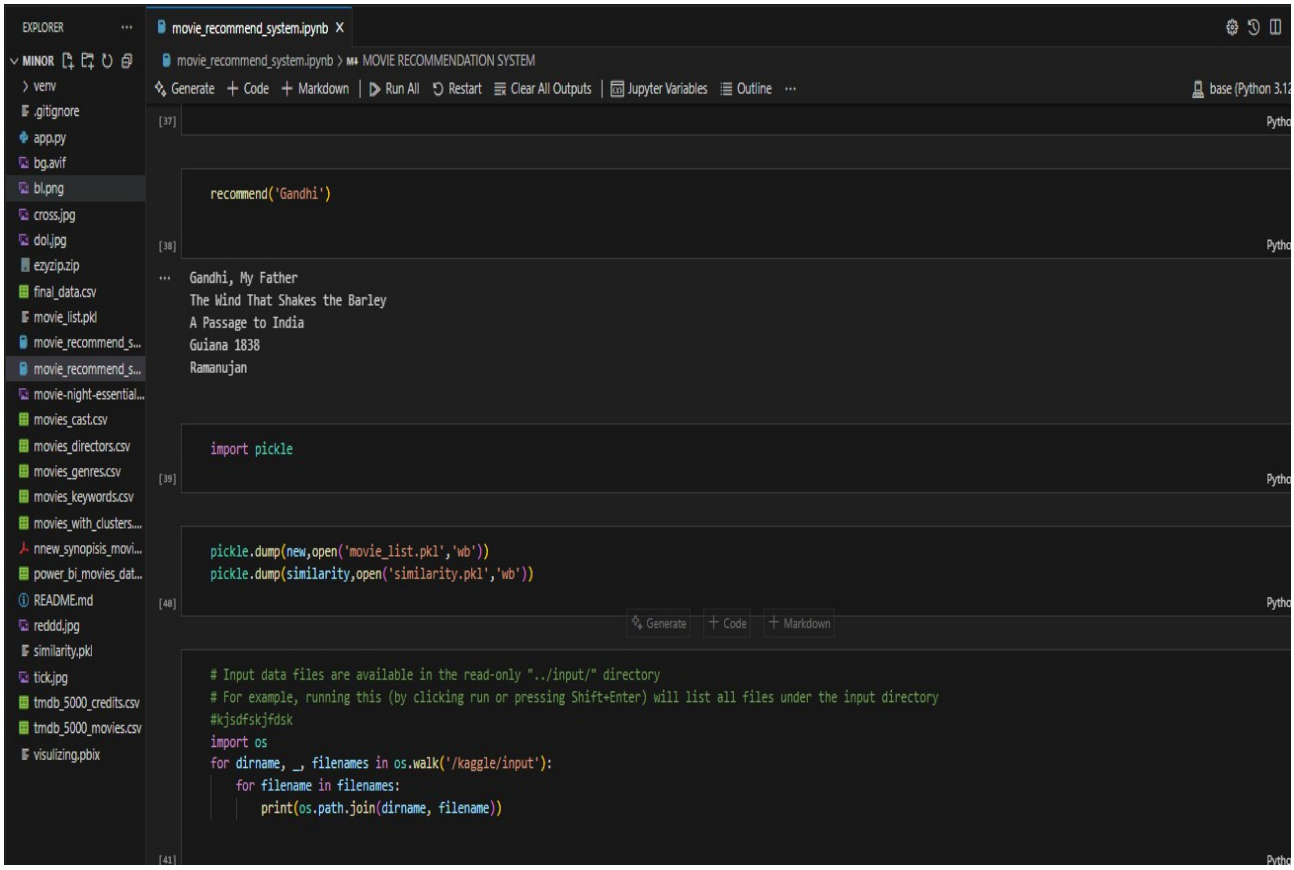

FIG. 5.3

5.4 Movie Recommendation Logic

- When a user selects a movie, its index is found from the DataFrame.
- Top 5 similar movies are returned based on cosine similarity scores.
- Each recommended movie is matched to its TMDB ID and corresponding poster is fetched via API.

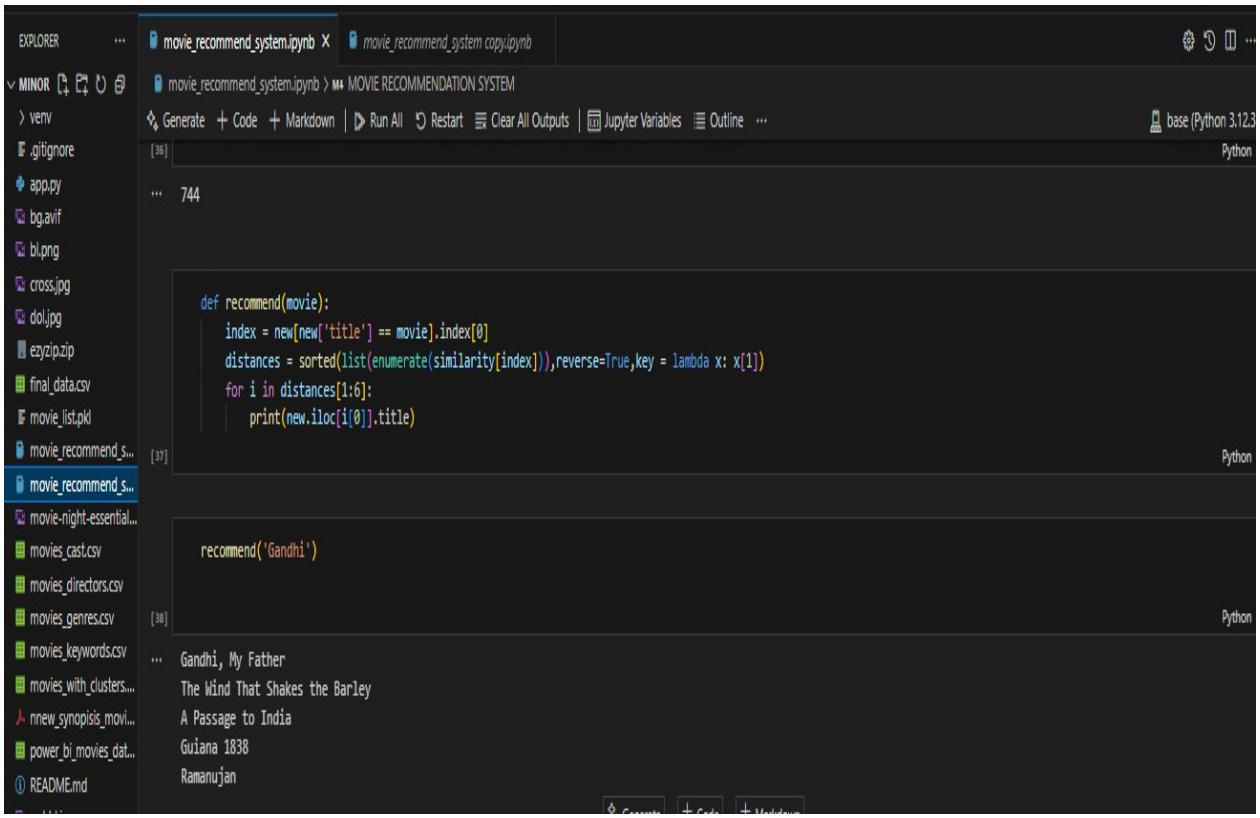

FIG. 5.4

5.5 Streamlit Interface Development

- Streamlit was used to create a user-friendly interface.
- The interface includes:
- Header
- Movie selection dropdown
- “Show Recommendation” button
- 5 recommended movie posters and titles in a row

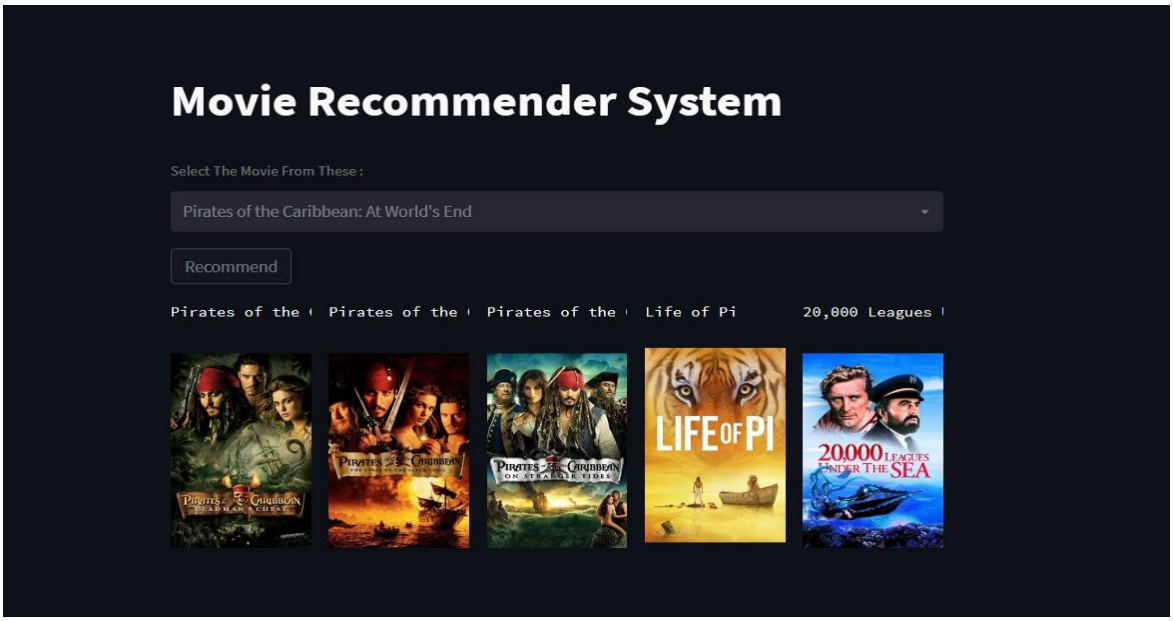

FIG. 5.5

5.6 Default Random Movie Suggestions (Optional Feature)

- If no movie is selected, the app displays 5 random popular movies.
- Adds dynamic behaviour to keep the app engaging.

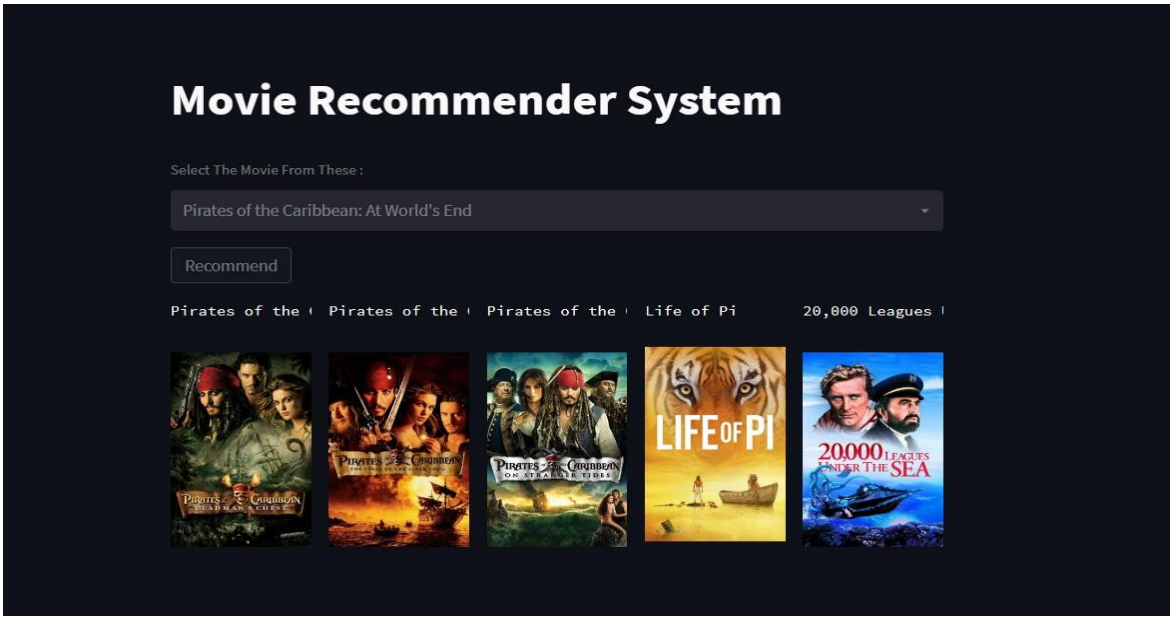

FIG. 5.6

5.7 Power BI Dashboard Integration

- Preprocessed data was exported to CSV and loaded into Power BI.

- **Dashboard 1: Movie Genre Trends**
  - Bar chart for genre count
  - Scatter plot for vote vs popularity
  - KPIs: Total votes, average rating

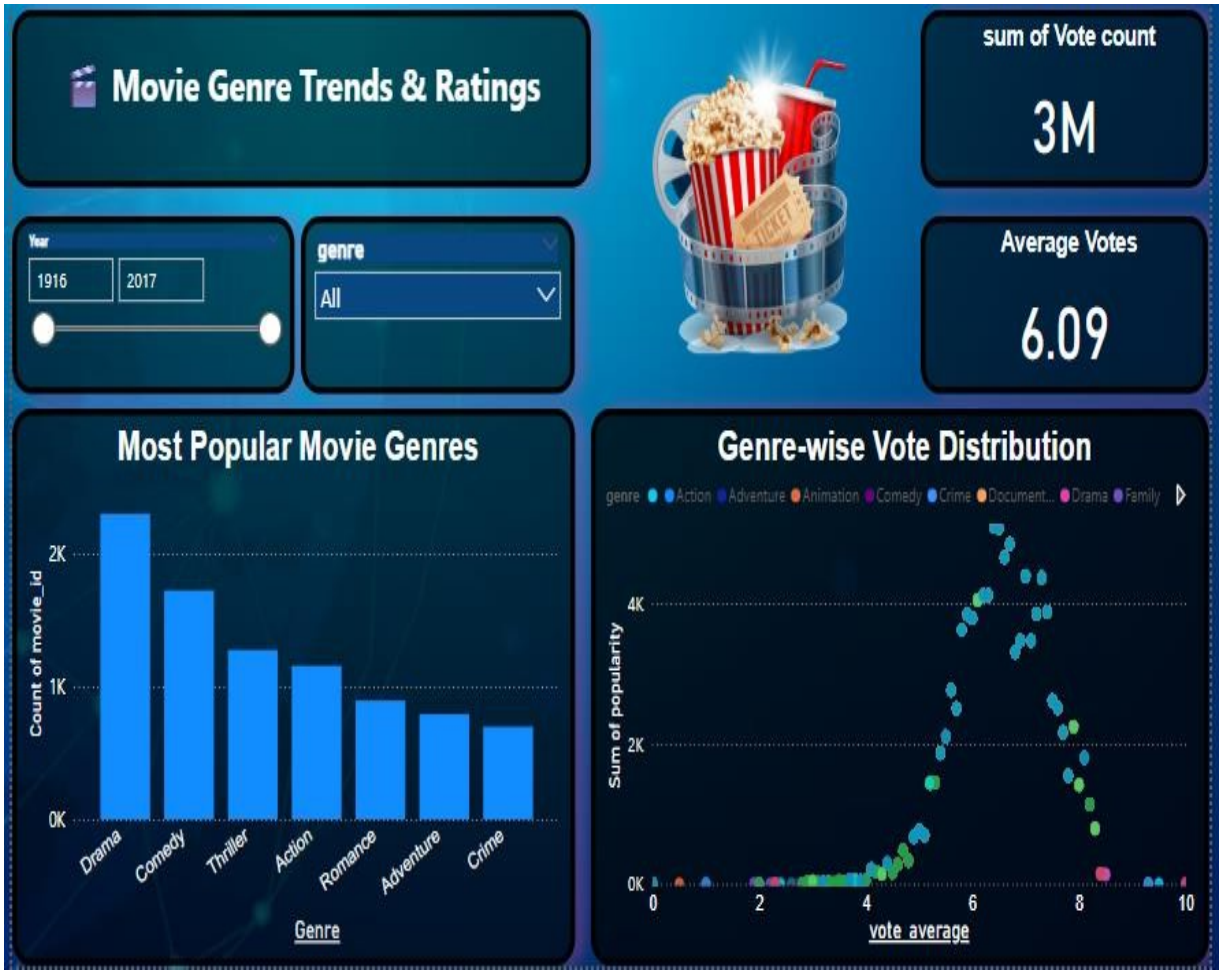

FIG. 5.7 (1)

- **Dashboard 2: Blockbuster vs Flop Analysis**
  - Bubble chart: budget vs revenue
  - Card KPIs: total movies, blockbuster count, flop count
  - Line chart for revenue trends

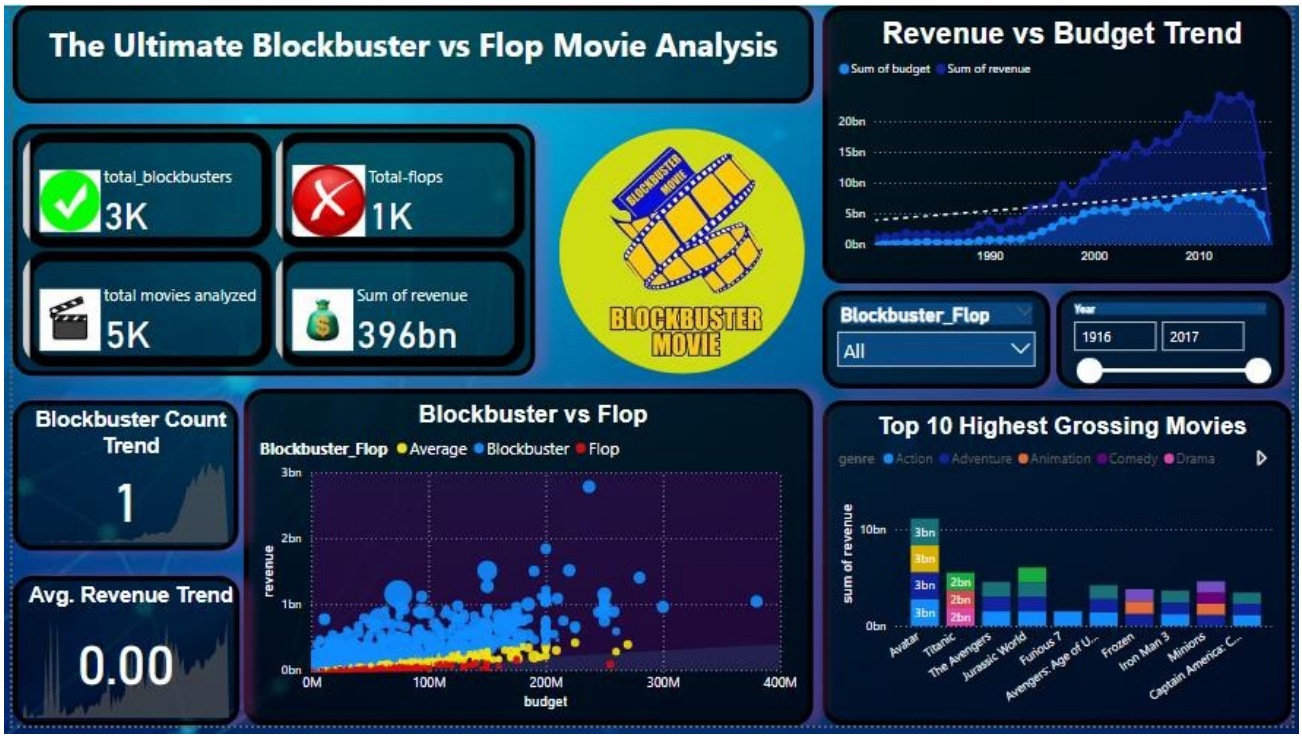

FIG. 5.7 (2)

- **Dashboard 3: Movie Clustering with KMeans**
  - Cluster-wise distribution of movies
  - Pie chart, table, and scatter plot
  - Helps analyse movie types without knowing exact genres

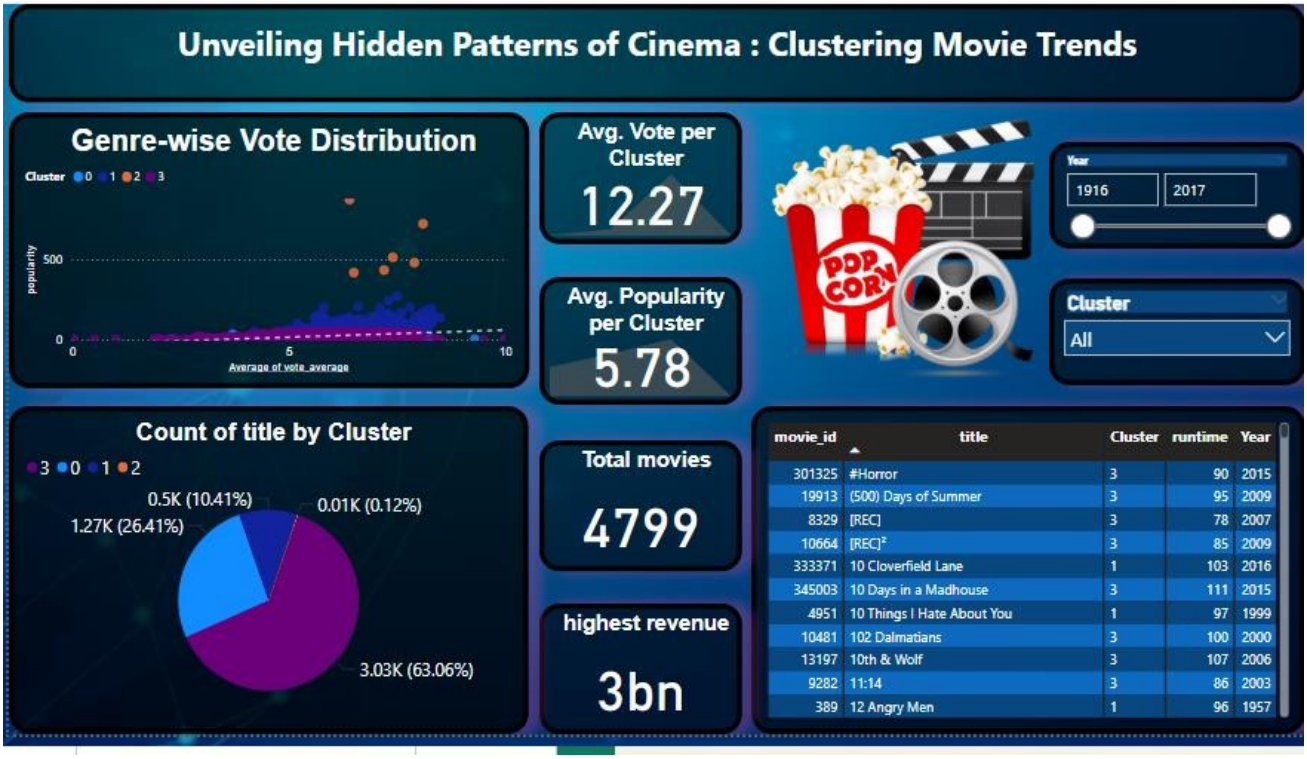

FIG. 5.7 (3)

5.8 Deployment and Execution

- **App can be run locally using:**

  streamlit run app.py

- **Power BI dashboards were saved and shared in .pbix format.**
- **The application is ready for future deployment on platforms like Streamlit Cloud or Heroku.**

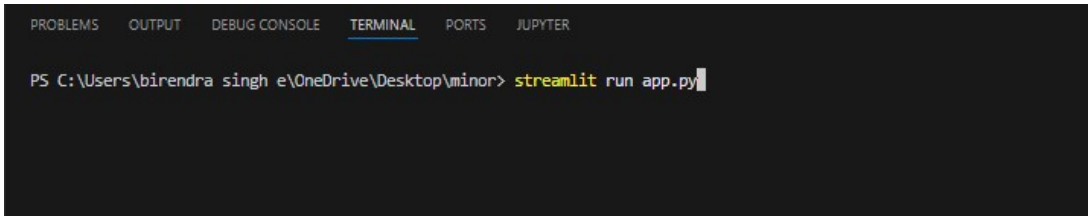

FIG. 5.8

5.9 Summary

This chapter demonstrates how each component of the system was implemented — from backend development to visual interface and dashboards. By combining Python, Streamlit, and Power BI, a complete system was delivered that is not only functional but also insightful.

CHAPTER 6

EVLUATION AND TESTING

This chapter outlines the detailed testing strategies applied during the development of the project, the tools used for testing, validation techniques, metrics used to evaluate system performance, and results obtained from various test cases.

6.1 Introduction

Evaluation and testing are crucial to ensuring the performance, accuracy, usability, and reliability of any software application. In the context of the Movie Recommendation System, evaluation involved verifying the correctness of recommendations, visual insights, and performance of the dashboards. Testing ensured that the application responds to user inputs correctly, handles errors gracefully, and delivers expected outcomes across different environments.

This chapter outlines the detailed testing strategies applied during the development of the project, the tools used for testing, validation techniques, metrics used to evaluate system performance, and results obtained from various test cases.

6.2 Types of Testing Applied

1. Unit Testing

- **Performed on individual functions like:**
- recommend()
- fetch_poster()
- Data cleaning functions
- **Ensured expected output with correct input**
- **Example:**

- **Input:recommend("Inception")**
  Output: 5 relevant movie titles with poster links

2. Integration Testing

- **Verified that all modules work together:**
- UI ↔ Backend ↔ TMDB API
- Movie dropdown → recommendation function → poster fetch → UI display
- **Ensured smooth end-to-end flow**

3. Functional Testing

- **Checked if the user can:**
- Select a movie
- Click a button
- Get output in form of movies + posters
- **Verified correct flow of data from input to output without breakage**

4. API Testing

- **TMDB API tested with valid and invalid movie IDs**
- **Ensured proper handling of missing posters and fallback to placeholders**

5. Usability Testing

- **Done with multiple users for feedback**
- **Observed:**
- Load time of app
- UI responsiveness
- Poster clarity
- Ease of navigation

□ 6. Performance Testing

- **Measured response time of recommendation function**
- **Observed API call response delay**
- **Verified that similarity matrix loading takes minimal time due to pickle**

□ 7. Dashboard Testing (Power BI)

- **Checked filtering with slicers**
- **Verified card values change correctly with filters**
- **Cross-filtering between charts tested**
- **Ensured insights are meaningful and update in real time**

6.3 Testing Tools Used

Tool	Purpose
Jupyter Notebook	Testing backend functions during development
Streamlit UI	Live testing of recommendation engine
Python Logging	Debugging API errors, similarity issues
Power BI Desktop	Dashboard design, filter testing, insights
Pandas	Data validation and dataset consistency

6.4 Evaluation Metrics Used

1. Relevance Score (Manual Evaluation)

- Compared output of recommend() with expected similar movies manually
- Most outputs were contextually relevant
- Example: Input "Batman Begins" → returned "The Dark Knight", "Batman v Superman"

2. Cosine Similarity Score

- Higher similarity = more accurate recommendation
- Cosine scores were used to rank movies in descending order

3. Time Efficiency

- Time to load model files: <1 second (using Pickle)
- Time to return recommendations: ~0.3 – 0.5 seconds

4. Dashboard Responsiveness

- KPI cards updated instantly with slicers
- Bar charts and scatter plots updated without lag

6.5 Test Cases

Test Case ID	Description	Input	Expected Output	Status
TC-01	Valid movie input	"Inception"	5 similar movies with posters	☑
TC-02	Invalid movie name	"xyz"	Error handled / No output displayed	☑
TC-03	TMDB API failure (network off)	Valid movie	Fallback poster shown	☑
TC-04	Empty selection	None	Random movies displayed	☑
TC-05	Dropdown interaction	Genre = Drama	Charts filtered accordingly	☑
TC-06	Clustering filter in dashboard	Cluster = 1	Table and scatter plot update	☑
TC-07	Flop movie filter	Category = Flop	Chart shows only flop movies	☑
TC-08	Power BI trendline responsiveness	Year = 2015	All visuals adjust to year filter	☑

6.6 Real User Feedback

“The interface is clean and simple. I typed ‘Titanic’ and got 5 beautiful recommendations right away.”
– User A, Non-tech

“Power BI charts helped us understand how genres are performing across years.” – User B, Data Analyst

“The clustering dashboard is impressive. I never thought movies could be grouped like this!” – User C, Student

6.7 Observations and Challenges

- **Challenge:** Some movies had no poster on TMDB
  ☑ **Solution:** Fallback placeholder URL used
- **Challenge:** Input movie not found due to case sensitivity
  ☑ **Solution:** Applied .str.lower() matching
- **Challenge:** Large CSV files slowed Power BI
  ☑ **Solution:** Pre-cleaned data before import, removed unnecessary columns
- **Challenge:** User expected recommendation based on rating also
  ☑ **Solution:** Clarified it’s content-based, not collaborative .

6.8 Summary

This chapter demonstrated that the system has been tested across all critical dimensions — accuracy, performance, usability, and responsiveness. The recommendation engine produced contextually relevant results with minimal response time. The dashboards offered actionable insights with dynamic filtering and interactivity. Overall, the Movie Recommendation System is functionally complete, stable, and reliable for further deployment and real-world use.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Introduction

The Movie Recommendation System project aimed to deliver an intelligent, user-friendly platform that suggests contextually relevant movies based on content similarity. The final system achieved this goal by combining machine learning techniques (cosine similarity, vectorisation), user interface tools (Streamlit), and visual analytics (Power BI) into a cohesive, scalable framework. This chapter concludes the report with a comprehensive summary of the accomplishments and discusses various avenues for future enhancement, expansion, and real-world applicability.

7.2 Project Achievements

The key achievements of the system can be summarised as follows:

- Successfully designed and implemented a content-based movie recommender using Python and machine learning principles.
- Created a responsive and interactive Streamlit-based UI for real-time recommendations.
- Integrated TMDB API to display authentic movie posters.
- Preprocessed and engineered features like genres, keywords, cast, and crew into a unified vector space.
- Applied cosine similarity to compute closeness between movies based on content.

- Exported data into Power BI dashboards for business-level analysis and insights.
- Built three professional dashboards:
- Movie Genre Trends & Ratings
- Blockbuster vs Flop Analysis
- Movie Clustering with K-Means

These accomplishments demonstrate a balanced implementation of data engineering, machine learning, and visual storytelling — making it a standout project with practical relevance.

7.3 Overall Observations and Learnings

Throughout the lifecycle of this project, several learnings were derived — technical, strategic, and analytical:

Technical Learnings

- Gained hands-on experience in:
- Feature extraction using CountVectorizer
- Text vectorisation and similarity computation
- API integration for real-time media retrieval
- Creating modular, reusable code using Python

Analytical Learnings

- Understood how to clean, filter, and visualise large datasets in Power BI.
- Developed dashboards that are both visually engaging and functionally powerful.
- Used clustering to uncover latent movie groupings beyond genre classification.

Strategic Learnings

- Learned how to divide a system into components: backend engine, frontend UI, and business dashboards.
- Practiced real-world debugging techniques, like handling API failures and missing data.
- Built a pipeline that can be scaled or reused for book, song, or product recommendation.

7.4 Real-World Applications

This project can be extended or deployed in various sectors and applications:

- OTT Platforms: Personalised movie recommendation based on user preferences and content similarity.
- Cinemas & Ticketing Portals: Display similar movies when a user browses a specific title.
- Entertainment Analytics: Power BI dashboards can help decision-makers analyse revenue trends, genre performance, and investment planning.
- Mobile Apps: A lightweight version of this system can be embedded into Android or iOS movie tracking apps.
- Education & Teaching: This system can be used as a teaching tool to demonstrate ML, NLP, data cleaning, and dashboarding.

7.5 Limitations of the Current System

While the project met its core objectives, a few limitations were noted:

- Only content-based filtering was used. No collaborative filtering or hybrid model was implemented.
- Recommendations are based solely on metadata — no user ratings or viewing history was considered.
- TMDB API dependency — if API quota is exhausted or down, poster fetching may fail.
- No user authentication or personal profile saving — all usage is anonymous and session-based.

- Visual analytics are limited to what was available in the dataset. New, real-time data integration is not present.

7.6 Future Enhancements

The system has immense potential to grow. The following features are recommended for future implementation:

1. Hybrid Recommendation Model

- Combine content-based filtering with collaborative filtering
- Use user rating matrix (via Matrix Factorisation or SVD)

2. Sentiment Analysis

- Perform NLP on user reviews to enhance recommendation quality
- Display emotion-based suggestions (feel-good, thriller, emotional, etc.)

3. Personalisation

- Add login and user profile feature
- Save user preferences and feedback for improving recommendation over time

4. Deep Learning Integration

- Use word embeddings (e.g., Word2Vec, BERT) to replace CountVectorizer
- Improve semantic matching using RNNs or Transformer models

5. Deployment on Cloud

- Host the application using Streamlit Cloud, Heroku, or AWS
- Enable real-time access to recommendation engine with user-based analytics

6. Expand to Mobile App

- Build React Native or Flutter app version
- Include offline cache for frequent recommendations

7. Time-Series Analytics

- Add Power BI charts showing how genres or ratings evolve over the years
- Track trend shifts in user demand across decades

7.7 Contribution to Knowledge and Industry

The Movie Recommendation System contributes to both academic understanding and real-world problem-solving by:

- Demonstrating how to transform unstructured data into actionable intelligence.
- Offering a ready-to-extend template for recommendation engines in other domains (e.g., books, news, shopping).
- Empowering stakeholders with visual dashboards to drive data-driven decisions.
- Bridging the gap between ML-based development and business-level analytics.

7.8 Final Words

The project successfully blends machine learning, data processing, and dashboarding into one cohesive system. It has educational value, business relevance, and user engagement potential. While the current version focuses on content-based movie suggestions, the future can expand toward real-time, cloud-integrated, and user-aware systems that reflect the best of AI-powered entertainment platforms.

This project is not just a technical submission — it is a data-driven storytelling experience that empowers users to discover movies smarter, and empowers analysts to understand cinema deeper.

CHAPTER 8

COMPARATIVE STUDY

8.1 Introduction

Recommendation systems are classified into several categories depending on how they generate suggestions. The major types include content-based filtering, collaborative filtering, hybrid models, and deep learning-based systems. This chapter presents a detailed comparison between these approaches, evaluates their advantages and limitations, and explains the justification for choosing content-based filtering in the current project.

8.2 Types of Recommendation Techniques

A. Content-Based Filtering

- Analyses the features (metadata) of the items (movies).
- Recommends similar items based on genres, cast, plot, keywords, etc.
- Works independently of other users.

B. Collaborative Filtering

- Based on user behaviour and preferences.
- Recommends items liked by similar users (user-user or item-item).
- Requires large user-rating matrix.

C. Hybrid Recommendation Systems

- Combine content and collaborative methods.
- Aim to utilise the strengths of both systems.
- More accurate but also more complex.

D. Deep Learning-Based Recommenders

- Use NLP models (e.g., RNNs, Transformers, BERT).

- Can understand deep semantic similarity between movies.
- Require powerful hardware and large datasets.

8.3 Feature-Based Comparison Table

Feature	Content-Based	Collaborative	Hybrid
Requires user data	N	Y	Y
Works for new users/items	Y	N	Y
Interpretability	Y	N	Y
Scalability	Y	N	-
Accuracy (personalised)	-	Y	Y
Ease of implementation	Y	-	N
Real-time performance	Y	-	-

8.4 Why Content-Based Was Chosen

- No user history was available (cold start).
- Focus was on learning ML concepts and text feature extraction.

- Dataset had detailed metadata, ideal for content-based methods.
- Model performs well even with new/unseen users.
- Lightweight and fast: Works offline after training.

8.5 Limitations of Other Techniques in This Context

Collaborative Filtering:

- Needs many user ratings (which were not available).
- Suffers from cold start problem for new movies or users.
- Cannot work without real-time user data.

Deep Learning:

- Requires high-end GPUs or cloud compute.
- Dataset too small to train advanced models like BERT.
- Not feasible for local deployment with Streamlit.

Hybrid:

- High development time and complexity.
- Not suitable for a short-duration academic project.

8.6 Use-Case-Based Comparison

Use Case	Best Technique
Recommending based on genre/cast	Content-Based Filtering
Netflix-like system with user history	Collaborative Filtering
Large-scale personalisation	Hybrid or Deep Learning
Lightweight, local system	Content-Based Filtering

8.7 Insights from the Study

- Content-based filtering is best suited for:
  - Academic projects
  - Metadata-rich datasets
  - Lightweight systems
- Deep learning is powerful but overkill for simple recommendations.
- Hybrid systems are gold standard in industry (e.g., Netflix, YouTube) but require extensive infrastructure.

8.8 Conclusion

After evaluating various techniques, the decision to use content-based filtering was validated. It provides a good balance between accuracy, simplicity, and speed, especially for a structured movie dataset without user interactions. The comparative study highlights that while other models offer depth, they also come with complexity. For a scalable academic solution with visual storytelling, content-based filtering remains the ideal choice.

CHAPTER 9

Conclusion and Reflections

9.1 Introduction

In the current digital landscape, recommendation systems have become a cornerstone of user experience, especially in platforms related to entertainment, shopping, and content consumption. The development of the Movie Recommendation System was not just an academic requirement but a journey into the real-world application of machine learning, data preprocessing, user interface design, and business intelligence through Power BI. This chapter reflects upon the learnings, significance, challenges, outcomes, and long-term impact of the project.

9.2 Summary of the Project

The Movie Recommendation System was designed with the objective of helping users discover movies based on their interests and preferences, using content-based filtering techniques. The project involved building a pipeline that extracted and cleaned metadata (such as genre, cast, keywords, and overview), converted it into a numeric vector space using CountVectorizer, and applied cosine similarity to generate movie recommendations.

The frontend was developed using Streamlit, providing an interactive and visually engaging interface. On the analytical side, extensive data-driven dashboards were created using Power BI, enabling deeper insights into movie trends, genre popularity, blockbuster analysis, and clustering-based insights using K-Means.

From concept to deployment, the project encapsulated all stages of a real-world data product lifecycle — from data sourcing and model building to visualisation and user delivery.

9.3 Major Takeaways and Learnings

This project served as a practical playground to experiment with various aspects of data science, analytics, and user-centric design. The following key learnings were achieved:

Technical Proficiency

- Hands-on experience with Python libraries such as pandas, numpy, sklearn, and pickle.
- Implemented NLP-based feature extraction using CountVectorizer.
- Used cosine similarity to recommend top 5 contextually similar movies.
- Implemented TMDB API to fetch real-time movie posters.

Analytical and Visualization Skills

- Leveraged Power BI to design professional dashboards.
- Created genre-wise distribution charts, popularity vs rating scatter plots, and clustering insights.
- Applied slicers, KPIs, and cards to make dashboards dynamic and business-ready.

Problem Solving and Strategy

- Solved issues related to missing values, inconsistent formats, and poster API errors.
- Designed modular code for easy reuse and scalability.
- Understood when and why to choose content-based filtering over collaborative models.

9.4 Real-World Significance

The Movie Recommendation System, though built in an academic setting, has clear implications in the real world:

- Entertainment Industry: OTT platforms like Netflix, Hotstar, and Amazon Prime thrive on recommendation systems. This project lays a basic version of such engines.
- Business Intelligence: The Power BI dashboards provide a framework for analysts to assess genre performance, popularity trends, and revenue metrics.
- ML Application: The project proves that even without user data, effective recommendations can be built using metadata and machine learning.
- Education & Teaching: The system can be used as a case study to demonstrate feature engineering, content filtering, and dashboard development.

9.5 Professional & Academic Growth

Working on this project helped enhance technical depth and professional maturity:

- Applied classroom concepts to a real-world project.
- Learned how to integrate frontend (Streamlit), backend (Python), and analytics (Power BI).
- Improved presentation skills by creating polished dashboards and explanations.
- Learned importance of documentation, testing, and user-centric thinking.

The project brought together various disciplines — programming, visualisation, logic building, and storytelling — and taught how they all align to solve a user's problem.

9.6 Reflections on Challenges Faced

Every good project comes with its share of roadblocks. This one was no different:

Challenge	How It Was Overcome
Poster not available for some movies	Placeholder used via error handling
Cosine similarity failing for non-existent movie	Input validation implemented

Challenge	How It Was Overcome
Large CSV files slowed Power BI	Removed unused columns, filtered rows
Teacher wanted uniqueness	Added clustering, dashboards, and random pick feature
Deployment errors in Streamlit	Fixed path issues and environment variables

Overcoming these issues gave a deeper understanding of **real-time debugging**, **error tolerance**, and the importance of scalable design.

9.7 Final Words and Thoughtful Closure

The Movie Recommendation System was not just a submission — it became a full-fledged project that taught how data, machine learning, APIs, and visualisation come together to build usable tools. It showcased how content-based models, despite their simplicity, can be highly effective when combined with good data and smart logic.

While the current system performs well, the future holds immense possibilities — from deploying it on the cloud to integrating hybrid models, or even transforming it into a mobile application.

In conclusion, this project provided a wholesome, end-to-end learning experience and laid the foundation for building intelligent systems in the future. It proved that with curiosity, commitment, and creativity, any problem can be transformed into a well-structured solution.

REFERENCES

The following references have been used throughout the course of the project for gathering data, learning concepts, using libraries, and implementing the functionalities involved in the Movie Recommendation System.

□ Web References

1. https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset
   Used for downloading the main movie dataset including metadata, credits, and keywords.
2. https://www.themoviedb.org/documentation/api
   Used TMDB API for fetching movie posters based on movie ID.
3. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html
   Used for applying cosine similarity on feature vectors.
4. https://streamlit.io/
   For documentation and implementation of the frontend recommendation interface.
5. https://pandas.pydata.org/
   Used for understanding data manipulation functions.
6. https://numpy.org/doc/
   Library documentation used for mathematical operations and data handling.
7. https://scikit-learn.org/
   Reference for machine learning model, vectorisation, and similarity computations.
8. https://learn.microsoft.com/en-us/power-bi/
   Used for learning Power BI features, KPIs, charts, and dashboard techniques.

□ Python Libraries Referenced

- pandas – for data manipulation and analysis
- numpy – for mathematical operations and arrays
- sklearn.feature_extraction.text.CountVectorizer – for text vectorisation
- sklearn.metrics.pairwise.cosine_similarity – for similarity calculation
- pickle – for model serialisation
- streamlit – for web application UI
- requests – for calling TMDB API
- ast – for safely parsing list-like strings
- matplotlib / seaborn *(if used for testing plots)*

□ Tools and Platforms

- **Jupyter Notebook** – For code development and testing
- **Visual Studio Code** – For application integration and debugging
- **Power BI Desktop** – For designing interactive dashboards
- **Google Colab** *(optional if used)* – For cloud-based model testing
- **Streamlit CLI** – For app deployment
- **GitHub** *(if repo hosted)* – For code versioning and project publishing

□ Acknowledged Courses & Tutorials

1. *"Build a Movie Recommendation System"* – YouTube by Krish Naik
2. *LinkedIn Learning & Coursera content* – For foundational understanding
3. *Medium blogs and GitHub projects* – For logic inspiration in combining datasets
4. *Official documentation of Python libraries* – For syntax and functions

❖ **Note:**

All references have been used strictly for **educational and non-commercial** purposes, respecting the intellectual property of the original authors.