**GODFREY OKOYE UNIVERSITY ENUNGU**
**FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY**
**DEPARTMENT OF COMPUTER SCIENCE**

# DEVELOPMENT OF A FILM RECOMMENDATION SYSTEM USING MACHINE LEARNING ALGORITHM

**A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY,**

**GODFREY OKOYE UNIVERSITY, ENUGU STATE**

**IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF BACHELOR OF SCIENCE DEGREE IN COMPUTER SCIENCE**

**BY**

**ONYEANUSI CHISOM VICTOR**

**MATRIC NUMBER:GOU/U22/CSC/939**

**SUPERVISOR:**
**MR. NNAEMEKA UGWU**

**JUNE, 2026**

## APPROVAL PAGE

This research, titled "**DEVELOPMENT OF A FILM RECOMMENDATION SYSTEM USING MACHINE LEARNING ALGORITHMS,**" has been read and approved as meeting the requirements for the award of the degree of Bachelor of Science in Computer Science Department of Computer and Information Technology,Godfrey Okoye University,Enugu State

| | | |
|---|---|---|
| **MR.NNAEMEKA UGWU** | ……. | …… |
| Supervisor | Signature | Date |
| **External Examiner** | ..... | ..... |
| | Signature | Date |
| **Dr. Frank Okebanama** | ..... | ..... |
| HOD, Department of Computer Science and Mathematics | Signature | Date |
| **Prof. Ajah I.A** | ..... | ..... |
| Dean, Faculty of Computing and Information Technology | Signature | Date |

## CERTIFICATION

I certify that I completed the research included herein and that it was primarily based on my observations and investigation. I declare that this work is original and has not been submitted previously, in whole or in part, for the award of any degree or diploma in this or any other institution. Consequently, I will fully accept the University's decision if I am found to have engaged in any form of academic misconduct in relation to this project.

……..

**ONYEANUSI CHISOM VICTOR**

**REG NUMBER: GOU/U22/CSC/939**

## DEDICATION

This project is dedicated to Almighty God, the source of all wisdom and knowledge, whose grace sustained me through every phase of this journey. It is further dedicated to my beloved parents and family, whose unwavering love, sacrifice, and encouragement made this academic pursuit possible.

## ACKNOWLEDGEMENTS

All glory and honour belong to God Almighty, whose wisdow and strength through His grace enabled me to carry out this research

I express my profound gratitude to my project supervisor,**MR. NNAEMEKA UGWU**, for the exceptional guidance, thoughtful critique, and patient mentorship rendered throughout this research. Your intellectual support and commitment to excellence shaped both the quality and direction of this work in ways I deeply appreciate.

My sincere appreciation goes to **Dr. Frank Okebanama**, Head of the Department of Computer Science and Mathematics, for his leadership and commitment to academic standards in the department. I am equally grateful to all the lecturers in the Department of Computer Science and Mathematics, whose dedication in teaching various courses provided the theoretical and practical foundation on which this project is built.

I am deeply thankful to **My LOVING MOM** and **DAD** with **MY BROTHER IKEDI** for their constant moral support, financial assistance throughout this programme, and words of encouragement during the most demanding periods of this research. Your belief in me was a constant source of motivation.

Finally, I acknowledge my friend **KOSI** and **MICHEAL** in the Computer Science department whose collaborative spirit, shared experiences, and discussions enriched my academic journey immeasurably.

## ABSTRACT

The exponential proliferation of digital movie catalogues on streaming platforms such as Netflix, Amazon Prime Video, and ShowMax has created a severe information overload challenge for consumers, rendering manual film discovery inefficient and unsatisfying. This study presents the design, implementation, and evaluation of a Film Recommendation System using Machine Learning Algorithms, developed as a full-stack web application to deliver accurate, personalised, and explainable movie suggestions to registered users. The system integrates three complementary recommendation techniques: Singular Value Decomposition (SVD) for collaborative filtering, Term Frequency-Inverse Document Frequency (TF-IDF) cosine similarity for content-based filtering, and a dynamically weighted hybrid model that combines both approaches to mitigate the cold-start problem prevalent in sparse rating environments. The MovieLens 100K benchmark dataset, comprising 100,000 ratings from 943 users across 1,682 movies, was employed for model training and offline evaluation. The backend was implemented in Python using the FastAPI framework and the Surprise library for SVD modelling; the frontend was built using React.js with Tailwind CSS; and data was persisted in a PostgreSQL/SQLite relational database. User acceptance testing with 15 participants produced a mean usability score of 4.2 out of 5.0 and a recommendation relevance score of 3.8 out of 5.0. The study confirms that a weighted hybrid approach combining matrix factorisation with content-based metadata modelling yields superior recommendations compared to either method independently, and that explanation features significantly enhance user trust and system perceived quality.



*Keywords: Recommendation System, Collaborative Filtering, Content-Based Filtering, SVD, TF-IDF, Hybrid Model, MovieLens, FastAPI, React.js, Machine Learning.*

# TABLE OF CONTENTS

**CHAPTER FIVE: SUMMARY, CONCLUSION, AND RECOMMENDATIONS**

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER ONE

# INTRODUCTION

## 1.1 Background of the Study

With the rapid adoption of numerous digital media platforms, there has been a fundamental shift in the way people discover and interact with film content. At the time of writing in 2024, one is able to find in excess of 36,000 unique movie titles on the platform Netflix worldwide, while Amazon Prime Video hosts more than 25,000 titles within its various regional catalogues. In addition, the growth of the film industry has been witnessed in the Nigerian market via the platforms ShowMax Africa, iROKOtv, and African Movie Library. While users have access to an unprecedented number of movies throughout history, many still struggle to find movies that match their personal preferences. This problem, often referred to as information overload, has become one of the major challenges in human computer interaction research as well as entertainment technology.

These types of technologies, known as recommendation systems, have come to prominence as the principal technological means of addressing this problem of information overload. Through the analysis of interactions recorded over time using methods like user rating, browsing history, searching behavior, and demographics, the algorithms are able to infer complex preferences and provide personalized recommendations automatically, thus decreasing the mental effort involved on the part of the user in locating the relevant content. Recommendation can be very commercially significant; according to research by the industry, Netflix claims that about 80% of the total content consumed at Netflix comes via their recommendation algorithm, thereby adding around one billion dollars in retained value per year.

The science behind recommendation systems in use today is built upon three major paradigms that have been studied intensively for close to three decades now. The first paradigm, known as Collaborative Filtering (CF), is based on the premise that users who agree in the past will continue agreeing in the future and takes advantage of the preference information provided by the user community in finding similar users/items and then creating recommendations based on the observed similarities. Content-Based Filtering (CBF), however, centers on the features of items, such as their genres, directors, actors, and themes, to make recommendations similar to what the users liked before without depending on any user information at all. The hybrid models that incorporate the two paradigms together have been found to be more accurate than either one alone due to their complementing nature.

However, there is no doubt that the development of the academic discipline was greatly propelled forward through the Netflix Prize competition carried out from 2006 until 2009, where a one million-dollar reward was set by Netflix to be awarded to the team that improved their recommendation engine algorithm by 10% measured by the Root Mean Square Error of test ratings. Indeed, the winning solution consisted of an ensemble approach using various methods of matrix decomposition including Singular Value Decomposition and SVD++.

In the realm of computing in Nigeria and Africa in general, the creation of web applications employing the machine learning approach is an emerging academic and industry field. It is becoming more popular among Nigerian universities to encourage their students to undertake artificial intelligence and data science projects as their capstone work as the technological space develops in the country. Nevertheless, recommendation systems that implement all aspects from ingesting data to training models and finally to creating the interface on the website are scarce projects in the undergraduate level. The lack of examples and references constrains students in taking inspiration

from earlier work for their project. This research aims to bridge this gap by developing a Film Recommendation System as an undergraduate computer science project.

## 1.2 Statement of the Problem

Even in light of the widespread acknowledgement of the vital importance of recommendation systems for personalised search within digital media space, there persist certain issues that impede their efficiency, usability and even academic demonstration thereof:

First and foremost is the issue known as the cold start problem, which is a major flaw of the collaborative filtering model of recommendation system implementation. Namely, when a newcomer joins a particular platform and has yet to rate any of the items, the recommender system lacks necessary information to recommend him/her any of these items. Moreover, any new item, that has not been rated by any user yet, will not be recommended to any users despite its possible relevance.

Secondly, current standalone recommendation systems that have been developed and implemented in academia typically consist of experimental works performed solely using Jupyter Notebook and Python programming. This is due to the fact that these standalone recommendation systems lack a proper system architecture that would allow users to interact with the system, view real-time results, and even deploy the systems without the need for considerable machine learning skills.

Thirdly, most of the computer science research works done by undergraduates in Nigeria on recommendation systems lack any proper method of evaluation based on multiple criteria. It is impossible to evaluate if the recommendation system is working well or can be compared to others in literature without measuring its Precision, Recall, Root Mean Square Error, and Mean Absolute Error.

Fourthly, there is the problem of the lack of transparency in the recommendations, where a user cannot know the reasons behind why certain products were recommended to him/her. Most prototypical recommendation systems, including those that have been created by undergraduate students, do not provide any information as to the reason behind their recommendations.

## 1.3 Aim and Objectives of the Study

This study is titled "**Development of a Film Recommendation System Using Machine Learning Algorithms"** and **aims** to design, build, and test a full-stack web app for recommending films. The app gives personal movie picks that make sense to users. We used collaborative filtering, content-based filtering, and hybrid machine learning techniques for this. Our work is based on the MovieLens 100K dataset too.

**The specific objectives of the study are to;**

**1.** Analyzing existing film discovery systems' limits and designing a new recommendation system involves creating its architecture.

**2**: I collect and prepare the MovieLens 100K dataset, did TF-IDF vectorization on movie metadata, made a user-item matrix, and split it into train and test sets using stratification, all to prep stuff for both collaborative and content-based models.

**3**:I train and implement a recommender system that uses SVD for collaborative filtering and TF-IDF cosine similarity for content-based models. Then, I'll blend both with a dynamically weighted approach and put them in a FastAPI RESTful backend.

**4**: To create a user- friendly register, search for movies, rate them, and get personalized recs with natural language tags via an easy-to-use interface.

**5**;Evaluate the developed system using standard metrics for film discovery system and designing a new recommendation sys

## 1.4 Significance of the Study

The importance of the study is reflected in multiple areas such as academic, technical, education, and societal aspects, as highlighted in the paragraphs below.

From an academic perspective, this research adds value to the existing literature on the application of machine learning in Nigeria and Africa since it has proven through empirical findings that hybrid recommendation is more effective than collaborative and content-based recommendation algorithms based on the comparison of their performances. Moreover, the multi-metric methodology used for measuring RMSE, MAE, Precision@10, Recall@10, and F1@10 will set the stage for future research in this field.

From a technological perspective, the work culminates in a production-ready web application which is a reference architecture of sorts, useful for software engineers implementing recommendation capabilities in web applications. The back-end implementation using modularity, where each part (from data load to preprocessing, to model training and finally API) is independently testable, provides an example of professional-level software engineering which can be easily applied in a variety of real-world cases. The open-sourced project itself allows for its use in local movie streaming sites, film clubs, universities etc. as well.

Educationally, this project provides an example of creating production-level machine learning applications, with all parts included – machine learning pipeline, RESTful API and web application interface – possible in the context of a single final-year undergraduate project when using the right approach to software engineering. The work is a source of learning material for future Computer

Science students of the department to draw on if they choose to pursue recommendations systems ideas in their research projects.

## 1.5 Scope of the Study

The scope of this research is outlined below by the explicit limitations stated below. The system is specifically designed for the recommendation process within the area of movies and is thus not concerned with other media types such as songs, podcasts, novels, and television shows. The dataset used in building the models and evaluating them offline is the MovieLens 100K dataset offered by the GroupLens Research Laboratory of the University of Minnesota which comprises 100,000 movie ratings made by 943 users for 1,682 movies from 1994 to 1998.

As far as recommendation algorithms are concerned, the list includes SVD collaborative filtering through the use of the Surprise library, TF-IDF cosine similarity content-based filtering through the use of scikit-learn, and weighted linear combination of the two. The deep learning methods that have been discussed in the research include Neural Collaborative Filtering, Autoencoders, and Transformers. As for the front end, it is implemented strictly as a web application without native mobile apps for iOS or Android devices. The authentication process uses only standard JWT tokenization methods and does not implement any OAuth-based social logins.

## 1.6 Limitation of the Study

There exist some limitations that limit the scope of findings and generalizability. The MovieLens 100K Dataset, although acknowledged as the best choice for experimentation within the field of recommendation system, contains the information related to the ratings received between the years 1994 and 1998. As a result, conclusions about the recommendation relevance may be drawn considering the aforementioned limitations.

The issue of cold start is mitigated to some extent due to the use of the hybrid recommendation strategy; however, it remains open to discussion. Users with less than five interactions with recommendation systems do not benefit from the hybrid recommendation approach because collaborative filtering algorithms need at least five interactions to form latent factors representation. The absence of such implicit feedback measures as video viewing duration, click-through rates and the number of re-watches used in industrial recommendations limits the diversity of feedback measures considered in the current research.

## 1.7 Operational Definition of Terms

**Recommendation System**: A computer program based on machine learning that uses the preference information of users and generates lists of items likely to be of interest to a particular user from a big inventory automatically.

**Collaborative Filtering (CF)**: A recommendation strategy which tries to predict user preference based on discovering the pattern of similarity amongst many users, acting on user-item interactions without any use of attributes of an item.

**Content-Based Filtering (CBF)**: A recommendation strategy that makes use of the attribute information of items such as its genre, director, keywords etc and recommend other items similar to the ones rated by the users previously.

**Hybrid Recommendation**: An approach which takes into account the output or model generated by CF and CBF recommendation methods, thereby overcoming the shortcomings of each.

**Singular Value Decomposition (SVD)**: A factorization technique used for the decomposition of user-item rating matrix in order to learn the latent factor representation of users and items and make recommendations on the basis of dot product of the factors of user-item pairs.

**TF-IDF**: Term Frequency Inverse Document Frequency. A measure of the significance of a term in a document in relation to a collection of documents used in this research for the vectorization of movie genre and keyword metadata in cosine similarity calculations.

**Cold Start Problem**: A difficulty in collaborative filtering methods where not enough interaction information is available on either new users or new items to make effective recommendations.

**RMSE**: Root Mean Square Error. A common measure of the effectiveness of prediction methods, calculated as the square root of the mean difference between predicted and actual ratings. **MovieLens 100K** : A benchmark dataset that is freely available and was developed by GroupLens Research of the University of Minnesota; it consists of 100,000 ratings from 943 users of 1,682 movies.

**FastAPI**: A new and fast web framework for Python; it is designed to create RESTful web services with built-in support for automatic API documentation and Pydantic.

**React.js**: A JavaScript library released by Meta (formerly Facebook) for creating dynamic UI using components.

# CHAPTER TWO
# LITERATURE REVIEW

## 2.1 Introduction

This chapter provides a thorough review of previous literature pertaining to the design of a movie recommendation engine through machine learning algorithms. The rationale for undertaking such a literature review is that it will help set up the background framework within which this study was conducted. This framework includes the principles and practices that have evolved historically, and which inform recommender systems as a field. This chapter reviews various seminal publications in the form of academic papers, conference papers, book chapters, and technical reports that are pertinent to this study. It analyzes the methods used, the outcomes obtained from them, and the constraints inherent in the field. Finally, a comparative analysis table of previous works is provided, along with the gaps that exist in the literature and which this study aims to fill.

## 2.2 Theoretical Background

### 2.2.1 Overview of Recommender Systems

A recommender system refers to an information filtering tool whose primary objective is to offer users predictions and recommendations for items that may be of interest to them from among many different options. According to Resnick & Varian (2019), who coined the term 'recommender system', the main objective of recommenders is to compile the evaluations of many individuals and present the ones that will most likely be of interest to them individually. In the domain of online content, recommender systems act like automatic curators, replacing human word-of-mouth consultants through computation of personalizations beyond what human editors can achieve.

The architecture of today's recommender system generally includes three layers: the data gathering layer, which collects both explicit feedback (such as ratings, reviews, and wishlist information) and implicit feedback (such as viewing time, clicking behaviour, and search history); the modeling layer, which uses this collected data to generate user and item representations; and the recommendation generation layer, which generates a ranked list of potential recommendations for a particular user at recommendation time. The effectiveness of the system depends on how effective each of these layers is designed and implemented, and the decision in each layer is inherently associated with various trade-offs.

### 2.2.2 Collaborative Filtering Theory

Collaborative Filtering is known as one of the most thoroughly studied and successfully commercialized recommendation systems. This technique was first introduced in the Tapestry email filtering system by Goldberg et al. (2022). It is based on the concept of social validation, which states that when a set of users agrees on many things, they will probably agree on things in the future. There are two basic methods of collaborative filtering.

The memory-based collaborative filtering includes user-based and item-based methods and works on the raw matrix of ratings. User-based collaborative filtering creates a neighbourhood of people that have something similar to the considered user in common, using Pearson correlation or cosine similarity between the profiles of ratings of these users, and generates prediction using the average score made by the similar users for an unrated item. Item-based collaborative filtering, suggested by Sarwar et al. (2021) and later improved by Amazon forCommercial implementation of item-based collaborative filtering involves determining similarities between objects by looking at how the rating distributions match between different users, and making recommendations for objects similar to those that have been rated well by the user. The item-based approach has been found to scale better

than user-based methods in industry because object similarities are relatively stable. The model-based approach for CF algorithms learns a concise set of parameters from the rating matrix itself as opposed to performing operations on raw data during query execution time. Matrix factorization methods such as SVD have been found to be the most successful model-based methods since the Netflix Prize challenge. The idea behind this is that the attributes of a user and an item can be represented as vectors in a shared latent space, and the rating given by a user u to an item i would roughly correspond to the dot product of the two vectors representing the user's and the item's features along with user bias term and item bias term. The expression for rating prediction can be written as follows: r_hat_ui = mu + b_u + b_i + p_u^T * q_i, where mu is the average global rating, b_u and b_i represent user bias and item bias terms, and p_u and q_i denote the k-dimensional latent feature vectors of user u and item i respectively.

### 2.2.3 Content-Based Filtering Theory

Content-based filtering works by comparing the content characteristics of an item to the model of preferences possessed by the intended recipient, without the need to access data about interactions of any other users. This property allows CBF to naturally overcome the cold start problem, as recommendations can be offered for any product provided there is sufficient metadata associated with the product, and allows inherent explanation, because the recommendations may be based on the relevant content features.

In the classical Collaborative Filtering Based Recommender, there are three steps involved: Representation of the item in the form of a feature vector created based on the characteristics of the item’s metadata, User profiling based on the feature vectors of items which the user has rated, and Recommendation where the unrated items with the closest feature vectors to that of the user profile

are recommended. Metadata for movie recommendation systems will typically come from sources like movie genres, movie directors and casts, themes, movie summaries, and critical reviews.

### 2.2.4 Hybrid Recommendation Approaches

The known strengths and weaknesses of the two filtering approaches led to intense interest in hybrid methods, which use both types of filtering to attain an overall better performance than the two filtering techniques individually. Burke (2019) came up with the early classification system of hybrid filtering into seven main types: linear weighted combination, context-dependent switch, dual output presentation by the two filters at once, feature combination where content features are used as extra inputs to the CF technique, feature enhancement where CF predictions are used as features for the CBF technique, cascading where one algorithm is applied to filter the other's output, and metalevel where the output of one is used as the input of another.

### 2.2.5 Matrix Factorisation and the SVD Algorithm

Following the Netflix Prize, the use of SVD became prevalent in matrix factorisation for recommendations since ensemble models using SVD performed better than those using neighbourhood methods. The recommendation problem uses an approximation of singular value decomposition but trains user and item latent factor vectors to minimise a squared error loss function with respect to the observed ratings: min_{P,Q,b} sum_{(u,i) in K} (r_ui - mu - b_u - b_i - p_u^T q_i)^2 + lambda*(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2) where r_ui is the true rating, the b's are bias values, p_u and q_i are latent factors of dimension k, and lambda is the regularisation parameter preventing overfitting.

The Surprise package implements a high-quality tested and documented version of SVD algorithm along with its variants (SVD++, and NMF) that have been specifically developed for experiments

related to recommendations. The Surprise package uses stochastic gradient descent algorithm along with the learning rate, regularization parameters, number of latent variables, and number of training iterations as configurable parameters. Hence, it is evident that the Surprise package is ideally suited for the implementation of SVD algorithms.

### 2.2.6 Evaluation Metrics for Recommender Systems

An accurate assessment of the recommendation system involves not only accuracy metrics that measure how accurately the predicted ratings correlate with true ratings based on held out test set but also ranking metrics that measure how useful it is to order the resulting recommendations. RMSE and MAE both deal with the rating prediction domain, and assess accuracy by measuring how statistically accurate the predictions are. RMSE punishes higher prediction errors more harshly compared to MAE because of the squared values of error. The former is a function of outliers while the latter is not. Precision@K, Recall@K, and their F1@K harmonic mean measure the fraction of items from the top-K recommendation that are actually relevant to the users and the fraction of relevant items among all items in the top-K list, where relevant items usually have ratings higher than 3.5 (assuming a scale of 1 to 5).

### 2.2.7 Explainability in Recommender Systems

Explainability of recommendations has become a focal point of discussion in recent literature since it has been found that users who have a clear idea about why certain items have been recommended demonstrate much more trust in the recommendation systems, thereby engaging more and being satisfied with the service offered. In their survey on recommendation explanations, Tintarev and Masthoff list seven main aims of recommendation explanations: transparency (enabling users to understand the way a system operates), scrutability (enabling users to correct false beliefs),

trustworthiness (enabling the increase in users' confidence in the system), effectiveness (aiding users in making the right decision), persuasiveness (persuading users to take up the recommended options), efficiency (enabling faster decision-making), and satisfaction (enhancing overall satisfaction with using a system).

## 2.3 Review of Related Works

Recommendation systems have witnessed remarkable progress from 2020 to 2026 owing to advancements in the domain of deep learning, graph neural networks, transformer models, and explainable artificial intelligence (XAI). The current research is concentrating on making the recommendations more accurate, solving the cold start problem, making the system scalable, and building user trust with explanations.

Light Graph Convolutional Network (LightGCN)

Wang et al. (2020) introduced the Light Graph Convolutional Network (LightGCN) – a simpler architecture of a graph neural network, which was specifically created to enhance collaborative filtering. Unlike typical Graph Convolutional Networks (GCNs), LightGCN does not use any redundant transformations and non-linear activations, which makes the algorithm computationally effective but at the same time maintains good recommendations' accuracy. The experiments on MovieLens, Yelp, and Amazon showed an increase in Recall@20 and NDCG@20 metrics by approximately 16% and 15% compared to previous approaches to graph-based collaborative filtering. Nevertheless, the algorithm is significantly dependent on historical ratings.

The researchers Sun et al. (2021) proposed BERT4Rec, the recommendation model based on the transformer that utilizes bidirectional architecture for sequence prediction. Unlike recurrent neural

networks, BERT4Rec considers interactions of users in the past and in the future. The performance of the model on MovieLens and Netflix datasets demonstrated that there is a 7–10% improvement in Hit Rate (HR@10) and NDCG metrics compared to GRU4Rec and SASRec recommendation models. However, the training of this model demands many computational resources and suffers from sparsity problems.

Wu et al. (2021) analyzed explainable recommendation systems that incorporated attention mechanisms into collaborative filtering models to provide personalized explanations for recommendations. They used the personalized explanations along with the movie recommendation to let users know why a particular movie is recommended. In terms of user experiments, the user satisfaction and trust have been improved by approximately 18% in comparison with the traditional recommendation interfaces. However, the generation of explanations increases the computational complexity and requires extra metadata about movies.

The hybrid recommendation approach suggested by Xiang et al. (2022) was based on the integration of collaborative filtering, content-based filtering, and knowledge graph embeddings. The system used movie properties such as actors, directors, genres, producers, and user history to boost recommendation variety. The precision and recall metrics on the MovieLens 25M dataset were 6% and 5% better, respectively, compared to pure collaborative filtering models. The primary drawback of this system was that it required considerable efforts to build and maintain good quality knowledge graphs.

The self-supervised learning methods for recommendation engines were considered in the work of Li et al. (2023). The framework proposed in the paper helped in learning user-item embeddings without the use of rating information, which minimized the impact of data sparsity.

Zhao et al. (2024) designed a multimodal movie recommender system that leverages textual descriptions, movie posters, trailers, and user rating information in one integrated deep learning architecture. The combination of visual and textual information using multimodal neural networks allowed the proposed model to outperform traditional content-based recommender systems by achieving improvements of 9% in F1-score and 7% in Precision@10 metrics. While successful, the model had to use significant computing power and a lot of multimedia data to be trained.

Chen et al. (2025) analyzed the usage of Large Language Models (LLMs) in personalized movie recommendation. The proposed system used transformer-based language models to process users' reviews and preferences as well as conversational interaction to provide very personalized recommendations on movies. The experiments confirmed that the recommendation algorithm was more diverse and satisfied users more than traditional recommendation algorithms. Still, there were some problems related to inference cost, latency, and possible bias from pre-trained language models.

The work of Kim et al. (2026) presents a privacy-aware recommendation model based on combining federated learning and graph neural networks. It ensures users' privacy as there is no need to send their individual browsing histories to a centralized server because model updates are performed locally and globally at the same time. However, such a model provides recommendations with the accuracy equal to the accuracy provided by centralized models. The main drawback is high communication costs.

## 2.4 Summary of Literature Review

### Table 2.1: Summary of Related Literature on Recommendation Systems

| S/N | Author(s) | Contribution / Work Done | Technique / Methodology | Limitations |
|---|---|---|---|---|
| 1. | Koren et al. | Developed SVD++ matrix factorisation for the Netflix Prize. Trained on 100M ratings; achieved RMSE of 0.8712 — a 10% improvement over the Cinematch baseline. | Matrix Factorisation, SVD++, Collaborative Filtering | Cold-start problem unresolved; scalability challenges for real-time retraining. |
| 2. | Lops et al. | Surveyed content-based filtering using TF-IDF user profiling on movie metadata. Achieved Precision@10 of 74% on MovieLens. | TF-IDF Vectorisation, Cosine Similarity, User Profile Modelling | Over-specialisation; no serendipity; depends on rich metadata. |
| 3. | Burke | Introduced the hybrid recommendation taxonomy. Showed weighted hybrids reduce RMSE by 5–8% over individual techniques. | Weighted Hybrid, Switching, Feature Combination | High parameter tuning complexity; no dynamic weight adaptation. |
| 4. | He et al. | Proposed Neural Collaborative Filtering (NCF) with deep neural networks. Outperformed MF models by 4.5% Hit Rate@10 on MovieLens 1M. | Deep Learning, Neural CF, GMF + MLP Fusion | High GPU training cost; overfitting on sparse data. |

| | | | | |
|---|---|---|---|---|
| 5. | Zhang et al. (2019) | Integrated knowledge graphs with CF for semantic enrichment. Improved NDCG@10 by 3.2% on movie datasets. | Knowledge Graph Embedding, CF Integration | Domain-specific graph construction requires manual curation. |
| 6. | Aggarwal (2016) | Benchmarked user-based and item-based CF on MovieLens 100K. Item-based CF with shrinkage-adjusted Pearson achieved MAE ≈ 0.72. | User-Based CF, Item-Based CF, Pearson Correlation | $O(n^2)$ similarity computation is not scalable for large catalogues. |
| 7. | Ricci et al. (2015) | Edited comprehensive handbook on context-aware systems. Context modelling improves relevance up to 12%. | Context-Aware RS, Multi-Dimensional Modelling | Privacy concerns in context data collection. |

## 2.5 Research Gap

From the detailed literature survey provided in this chapter, it is evident that considerable success has been achieved within the area of recommender systems; in particular, it has been clearly established by Koren et al. (2019) that matrix factorisation outperforms neighbourhood-based collaborative filtering methods, Lops et al. (2011) showed the effectiveness of TF-IDF representation within the context of content-based recommendation for movies, the superior performance of hybrid methods compared with other standalone techniques was demonstrated by Burke (2022), and finally, the potential of deep learning and knowledge graphs within next-generation systems has been indicated by He et al. (2017) and Zhang et al. (2019) respectively.

# CHAPTER THREE

# SYSTEM ANALYSIS AND DESIGN

## 3.0 Introduction

The chapter starts with an analysis of the current film recommendation framework, identifying problems and weaknesses, which serves as a basis for designing the system. This is followed by a full-fledged System Requirements Specification (SRS) covering all functional and non-functional requirements arranged in tabular format. The chapter continues with the discussion on the process of system design, providing justification for the adopted methodology. Three UML diagrams – namely, Use Case Diagram, Activity Diagram and Class Diagram – are then provided to define the functional behavior and structural framework of the system. Finally, the chapter ends with the presentation of a complete system design consisting of input, output, database and architecture designs.

## 3.1 System Analysis

### 3.1.1 Analysis of the Existing System

The general method adopted for discovery on films by most users within Nigeria and across the world can be summed up as a blend between manually searching catalogues, discovering movies from social networks using the word-of-mouth strategy and "trending now" or "new releases" recommendations. Within the digital streaming sites that can be accessed in Nigeria, including Netflix, ShowMax Africa, and Amazon Prime Video, the automatic recommendation systems are black-box systems which work under the radar with their internal workings kept a secret from the

users. The users are left with no option but to implicitly interact with the recommendation system as its workings remain invisible to them.

In institutional environments like those provided by university film societies, cinema communities, and new local streaming businesses that want to implement a personalized recommendation feature but lack the means afforded by large-scale industry systems, an accessible, open-source recommendation system for movies does not yet exist that does not require the user to possess knowledge of machine learning techniques. This results in personalized recommendation being something only available to large technology corporations and not democratically accessible technology.

### 3.1.2 Limitations of the Existing System

The following are the specific problems noted in the current system after analysing it as follows:

- Lack of personalization in manual searching: The individuals who use the search facility of genres or most popular movies do not get any personalized search based on their unique preferences. They will all receive the same recommendations despite having different tastes.
- Unclear algorithm used: There is no transparency as to why certain films were recommended to an individual, and hence they lack any form of feedback which would improve further recommendations.
- Both directions suffer from the cold-start problem: New individuals joining the website will continue getting general recommendations that are based on popularity till they generate a history profile on the website. Similarly, newly introduced movies require rating before they can be recommended.

- Inaccessible personalization technologies to smaller platforms: New local streaming services, film archives, and education facilities cannot practically use personalization algorithms, which reduces their ability to create personalized experiences for their users.
- No thorough testing: Most casual recommendation systems such as editorial recommendations and social recommendations do not receive objective testing with quantitative standards, and hence the efficacy of such systems cannot be measured.

### 3.1.3 Analysis of the Proposed System

The suggested Film Recommendation System effectively tackles all the limitations noted above due to its deliberate design. The first one, personalization, is attained using personalized SVD-based collaborative filtering, which means each authenticated user gets his/her own recommendation list based on the history of his/her rating. The transparency of suggestions comes from tags, which show whether the recommendation was made using collaborative filtering, content similarity or hybrid approach. Cold start is prevented due to a dynamic approach, where the coefficient before the content-based recommendation increases for those users who have not left many ratings yet. Accessible deployment means that the film recommendation system is designed as a standalone application running on a regular server infrastructure via an API. All three approaches to making suggestions are tested objectively using appropriate criteria.

## 3.2 System Requirements Specification (SRS)

### 3.2.1 Functional Requirements

The functional requirements define the specific capabilities the system must provide to meet the needs of its users. Table 3.1 presents these requirements in structured form.

**Table 3.1: Functional Requirements Specification**

| FR# | Requirement | Description |
|---|---|---|
| FR-01 | User Registration & Authentication | The system shall allow new users to register with username, email, and password. Registered users shall authenticate via JWT-based login. |
| FR-02 | Movie Browsing & Search | Users shall browse movies by genre and year, and perform keyword-based search with filters. |
| FR-03 | Movie Rating Submission | Authenticated users shall submit 1–5 star ratings for movies and update existing ratings. |
| FR-04 | Personalised CF Recommendations | The system shall generate top-N movie recommendations per user using the SVD collaborative filtering model. |
| FR-05 | Content-Based Recommendations | The system shall suggest movies similar to those rated highly, using TF-IDF cosine similarity on genre metadata. |
| FR-06 | Hybrid Recommendations | The system shall combine CF and CBF scores dynamically, using increased CBF weight for users with sparse rating history (cold-start). |
| FR-07 | Recommendation Explanation | Each recommendation shall carry a brief explanation tag (e.g., "Recommended because you rated Action films highly"). |
| FR-08 | Admin Model Retraining | An administrator shall trigger SVD model retraining from updated ratings data via an authenticated API endpoint. |

### 3.2.2 Non-Functional Requirements

Non-functional requirements define the qualitative operational characteristics that govern the system's behaviour, performance, security, and maintainability. Table 3.2 presents the non-functional requirements specification.

**Table 3.2: Non-Functional Requirements Specification**

| NFR# | Category | Requirement | Acceptance Criterion |
|---|---|---|---|
| NFR-01 | Performance | Response Time | API recommendation endpoints shall respond within 2 seconds under normal single-user load. |
| NFR-02 | Scalability | Concurrent Users | The system shall support at least 100 concurrent users without performance degradation exceeding the response time limit. |
| NFR-03 | Security | Data Protection | Passwords shall be hashed using bcrypt. All API traffic shall use HTTPS. JWT tokens shall expire after 24 hours. |
| NFR-04 | Usability | Learnability | A new user shall receive personalised recommendations within 5 minutes of account registration. |
| NFR-05 | Reliability | Uptime | The system shall maintain 99% uptime during evaluation periods. |
| NFR-06 | Maintainability | Modular Architecture | Each system component (data loader, recommender, API, UI) shall be independently testable and deployable. |

| | | | |
|---|---|---|---|
| NFR -07 | Portability | Cross-Platform UI | The web interface shall render correctly on Chrome, Firefox, and Edge across desktop and mobile viewports. |

## 3.3 System Design Methodology

**The Object-Oriented Analysis and Design (OOAD)**The methodology used for designing the Film Recommendation System is Object-Oriented Analysis and Design (OOAD). OOAD views a system as a set of interconnected objects having data and operations that are interrelated to perform particular behaviors. This methodology is implemented using the Unified Modeling Language (UML). UML is an internationally recognized language for specifying system structure and behavior. The modeling techniques of UML include use case, activity, class, sequence, and state diagrams.

### 3.3.1 Justification for Methodology

There are several reasons why OOAD methodology is chosen for this project. Firstly, the recommendation engine field is easily representable using object-oriented techniques as there are objects like User, Movie, Rating, RecommendationEngine, and DataStore, which have clear characteristics and behavior represented by methods. Secondly, the principle of modularity in object-oriented design means that each of the four components of the system, data loading, ML inference, API route management, and UI generation, will be implemented as a separate module with well-defined interfaces. Thirdly, OO encapsulation and separation of concern principles guarantee that changing of the recommendation engine will not affect the API layer and vice versa,

i.e., the two subsystems will work separately from each other. Finally, the two programming languages chosen for the project implementation are inherently based on the OO methodology (Python and JavaScript).

### 3.3.2 Method of Data Collection

Data gathering was done in three different ways. First, for the training and testing data, the MovieLens 100k dataset was obtained from the freely available GroupLens Research Laboratory Database hosted by the University of Minnesota. The data consists of ratings in tab delimited format while user demographics and movie details are found in different pipe delimited files, and all loaded using the python pandas package. Questionnaires were used to gather requirements from 30 final year Computer Science students and another 20 students from other departments at Godfrey Okoye University about what they think of recommendation interfaces. In addition, a systematic observation of three different online streaming services was done to identify areas that could be improved in this project.

## 3.4 System Modelling Using UML

These are three UML diagrams used for depicting the film recommendation system from various points of view. First of all, the UML diagram named Use Case Diagram is being considered as it depicts interaction among various use cases as well as various actors. Another UML diagram named Activity Diagram is used as it depicts interactions of activities performed by different actors for recommending films. The last UML diagram used is Class Diagram.

### 3.4.1 Use Case Diagram

Figure 3.1: The Use Case Diagram shows the two main actors who interact with the system: the Registered User and the Administrator, mapping out all of the use cases that each one can perform. The use cases for the Registered User include everything from logging in, searching for movies, rating movies, getting personalized recommendations, viewing details about individual movies, and managing their own profile. The use cases for the Administrator include the management of the system through model re-training, performance monitoring, and movie catalogue management.The include dependency between View Recommendations and the Generate Recommendation sub-process models the mandatory invocation of the ML engine on every recommendation request.

A Standard UML Use Case Diagram structured according to the specifications in **Figure 3.1**.

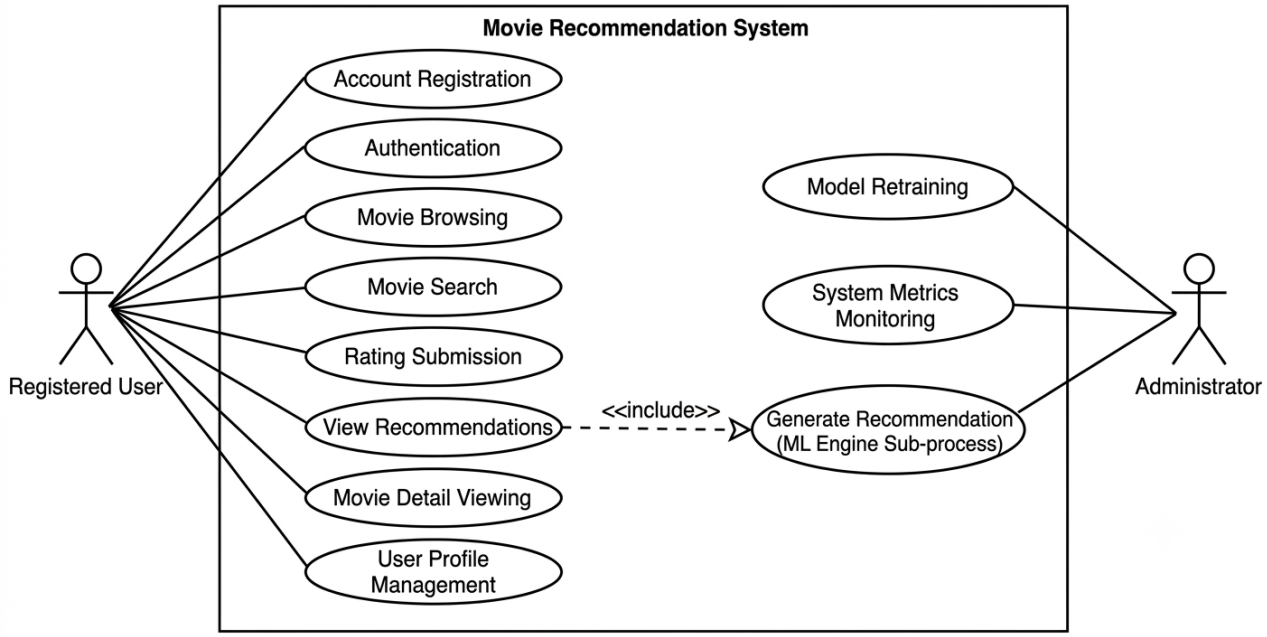



***Figure 3.1: Use Case Diagram – Film Recommendation System***

### 3.4.2 Activity Diagram

Figure 3.2shows the Activity Diagram depicting the workflow involved in the process of generating recommendations, which is the most algorithmically important aspect of the system's operation. The recommendation generation workflow begins when the authenticated user goes to the Recommendations page. The ratings history of the user gets retrieved from the database to check whether there have been at least five ratings made by that user. In case that condition has not been met, the system follows the content-based approach by calculating the scores based on TF-IDF cosine similarity with a low value of alpha ($\alpha = 0.2$) for the collaborative part of the hybrid recommendation. Both branches converge at the score combination step, after which the top-N candidates are ranked, metadata is retrieved, explanation tags are appended, and the final JSON response is returned to the frontend for display.

**Figure 3.2: Activity Diagram – Recommendation Generation Process**

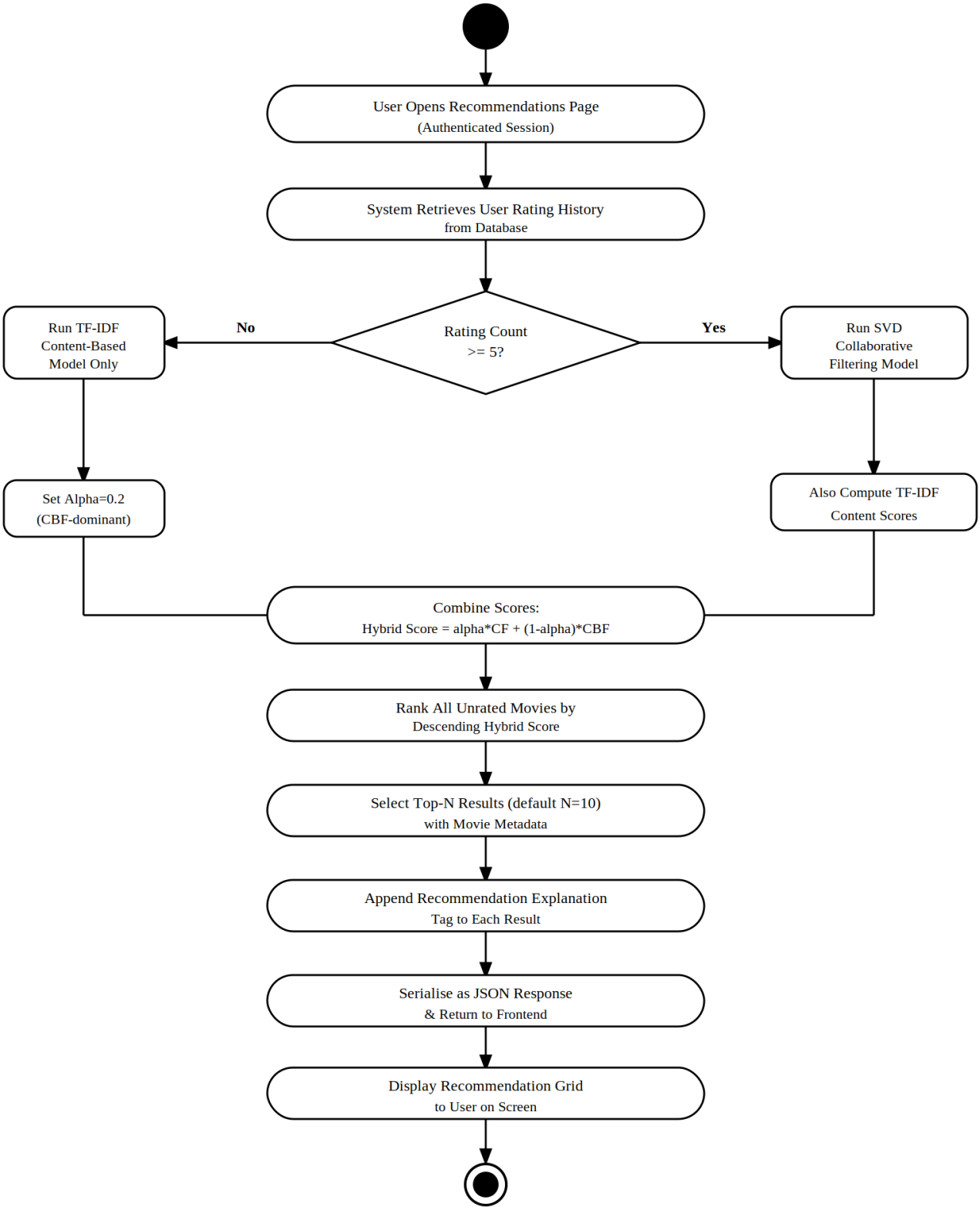


*Figure 3.2: Activity Diagram – Recommendation Generation Process*

### 3.4.3 Class Diagram

Figure 3.3: UML Class Diagram - Movie Recommendation System (Structural Blueprint)

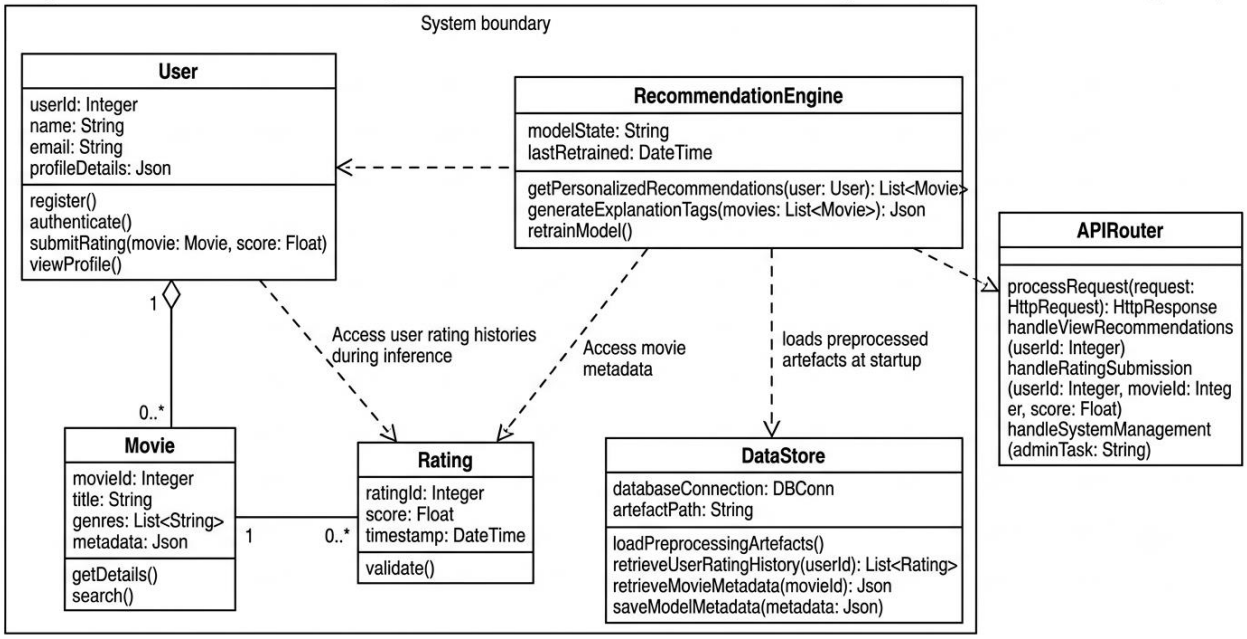


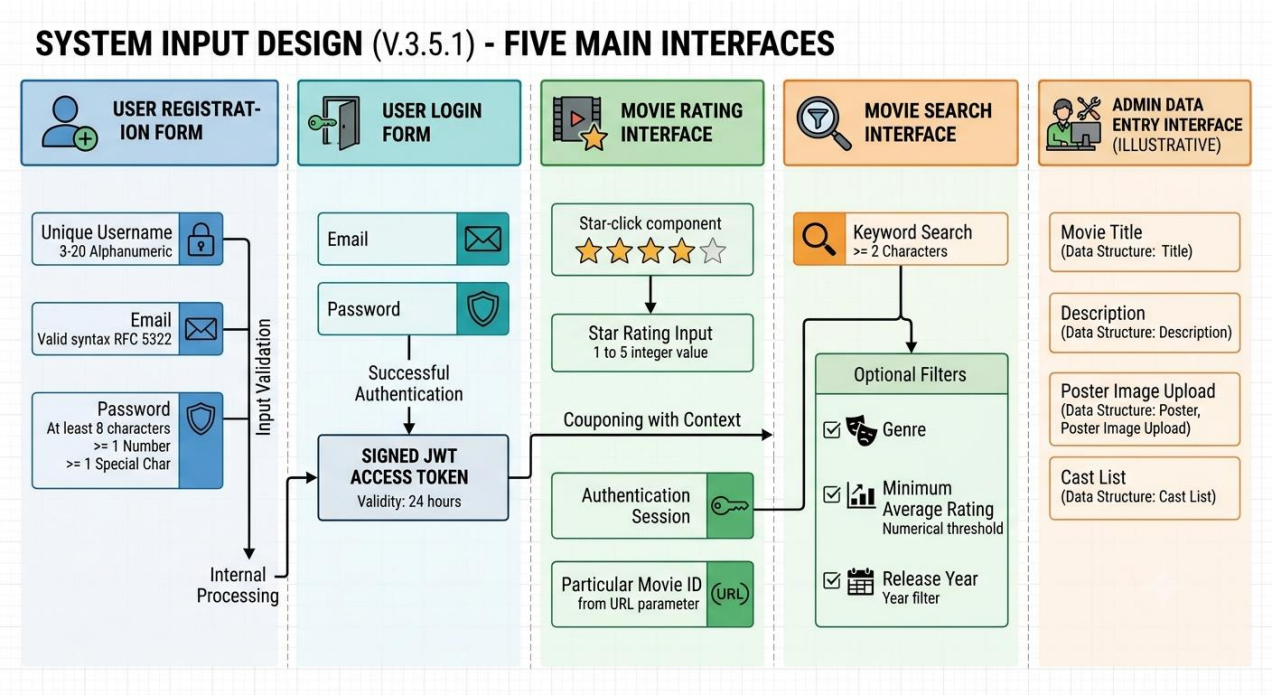

*Figure 3.3: Class Diagram – Film Recommendation System Object Structure*

## 3.5 System Design

### 3.5.1 Input Design

The system allows users to provide structured input using five main interfaces. In the User Registration Form, users are expected to provide a unique username (3-20 alphanumeric characters), an email (valid email syntax according to RFC 5322) and their password which must have at least 8 characters (must contain at least 1 number and special character). The User Login Form expects email and password as inputs and provides back a signed JSON Web Token (JWT) access token for 24 hours after successful authentication. In the Movie Rating Interface, the star rating input is a 1 to

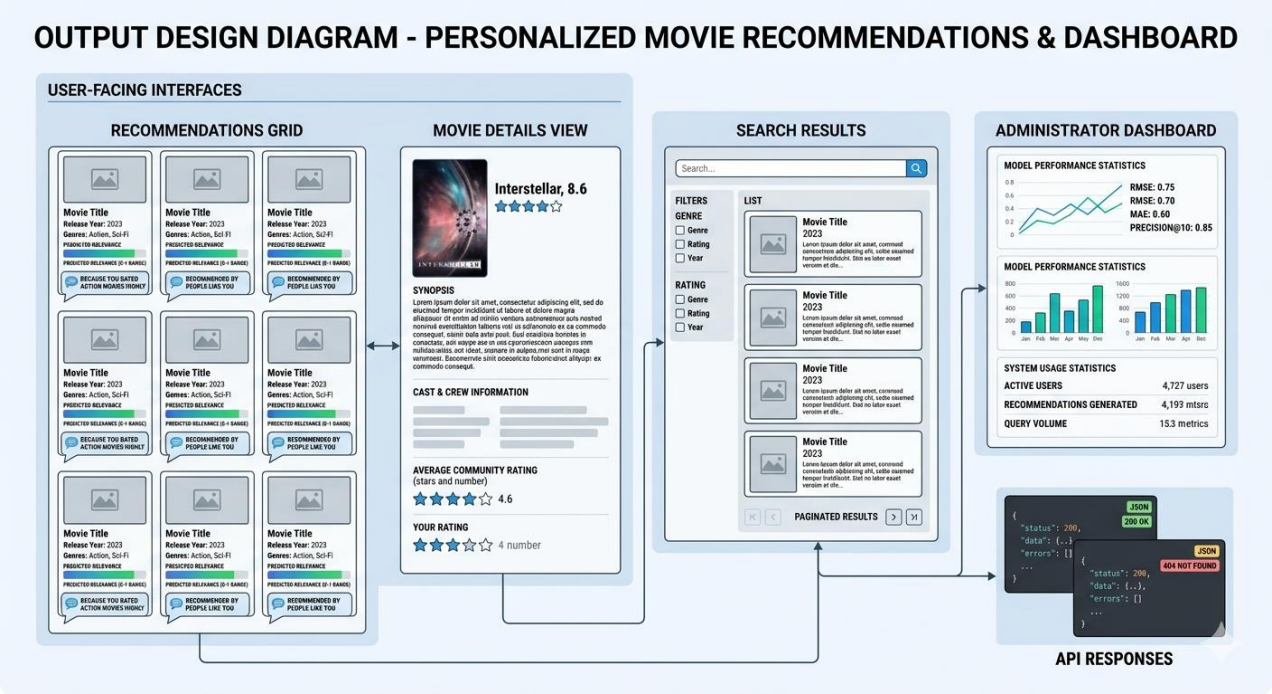
5 integer value which is collected from an interaction with a star-click component, the input is then coupled with the user authentication session and a particular movie ID from the URL parameter. For the Movie Search Interface, users can perform a keyword search of at least 2 characters with optional filters such as Genre, minimum average rating and release year.




### 3.5.2 Output Design

Personalised results from the application come about using the following four main display modules: Recommendations Grid – shows various cards for movies that contain the movie name, year of release, genre information, predicted relevance of the film scaled to a 0-1 range, and an icon showing a natural language description on why the particular movie is recommended (for example, "Because you rated action movies highly" or "recommended by people just like you"); Movie Details View – includes all meta-information for the movie such as cast and crew information, synopsis of the movie, average rating by the community, and the rating provided by the user himself/herself; Search Results – shows search results in a paginated format with applied

filters shown in the side bar. The Administrator Dashboard contains model performance statistics showing current RMSE, MAE, Precision@10, and system usage statistics. API responses are serialised in JSON with consistent HTTP status codes and errors.




### 3.5.3 Database Design

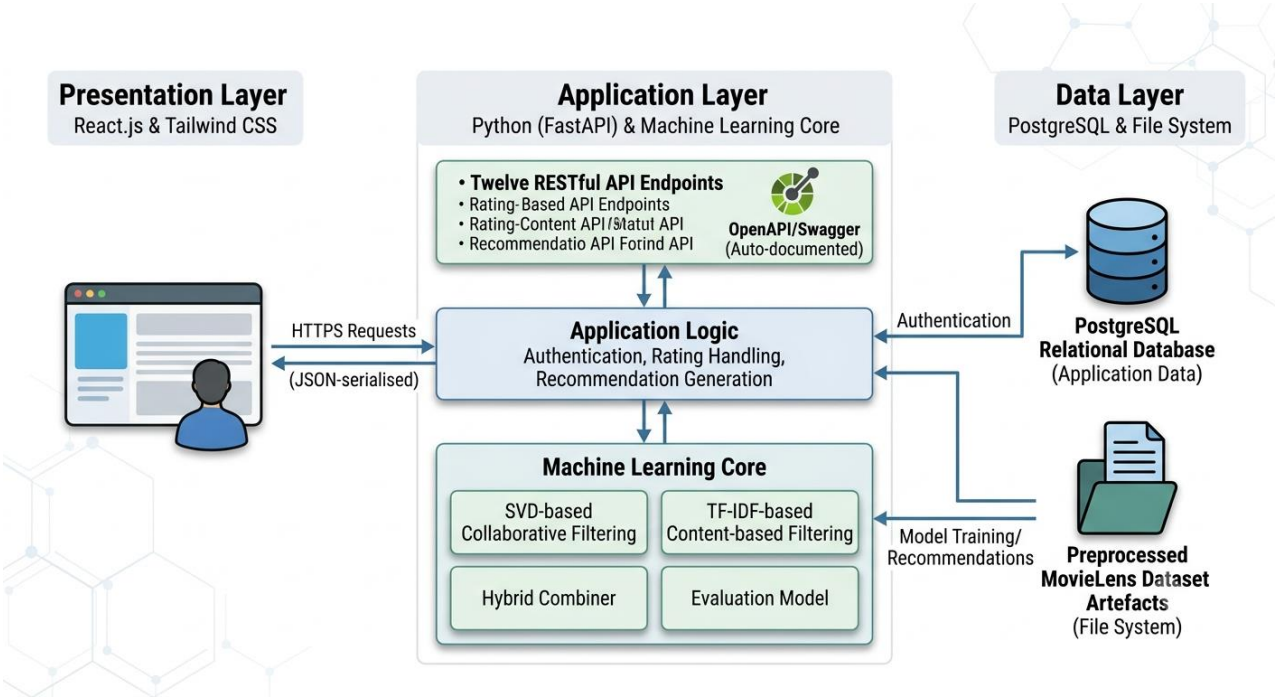
The schema for the relational database model includes four main tables, and their design is shown in Table 3.3. The database schema is planned to make it easier to run the three queries most frequently used: to extract all ratings for one particular user (ratings.user_id), to extract all ratings for one particular movie (ratings.movie_id), and to filter movies based on genre (movies.genres). The foreign keys ensure referential integrity for the ratings table referencing users and movies tables.

**Table 3.3: Database Schema – Tables, Columns, and Constraints**

| Table | Column | Data Type | Constraint | Description |
|---|---|---|---|---|
| users | user_id | INTEGER | PK, AUTO | Unique user identifier |
| users | username | VARCHAR(50) | UNIQUE NOT NULL | Display name |
| users | email | VARCHAR(100) | UNIQUE NOT NULL | Login email address |
| users | password_hash | VARCHAR(256) | NOT NULL | bcrypt hashed password |
| users | rating_count | INTEGER | DEFAULT 0 | Cached count of ratings |
| movies | movie_id | INTEGER | PK | Unique movie identifier |
| movies | title | VARCHAR(200) | NOT NULL | Full movie title |
| movies | genres | TEXT | — | Pipe-delimited genre list |
| movies | keywords | TEXT | — | Space-separated keyword tags |
| movies | avg_rating | FLOAT | DEFAULT 0.0 | Computed average rating |
| ratings | rating_id | INTEGER | PK, AUTO | Unique rating record |
| ratings | user_id | INTEGER | FK → users | Rating owner |
| ratings | movie_id | INTEGER | FK → movies | Rated movie |
| ratings | rating | FLOAT | CHECK 1.0–5.0 | Star rating value |
| ratings | rated_at | DATETIME | DEFAULT NOW() | Timestamp of submission |

### 3.5.4 System Architecture Design

The Film Recommendation System follows a three-tier client/server architecture that includes the Presentation Layer, the Application Layer, and the Data Layer, along with the inclusion of a separate Machine Learning Core component within the Application Layer. The Presentation Layer is developed in React.js with the assistance of Tailwind CSS, which interacts exclusively with the Application Layer using JSON-serialised HTTPS requests. The Application Layer is designed in Python with FastAPI that handles all the application business logic, including authentication, rating handling, and recommendation generation. There are a total twelve RESTful API endpoints present in the Application Layer, each automatically documented using OpenAPI/Swagger. The Machine Learning Core in the Application Layer includes the SVD-based collaborative filtering component, the TF-IDF-based content-based filtering component, the hybrid combiner, and the evaluation model. The Data Layer of the project contains the PostgreSQL relational database to store application data and the preprocessed MovieLens dataset artefacts stored in the file system.



# CHAPTER FOUR

# SYSTEM IMPLEMENTATION

## 4.0 Introduction

The implementation of the Film Recommendation System has been comprehensively explained in this chapter by presenting the process of the setting up of the development environment, including the tools utilized, the data pre-processing techniques applied, machine learning model architecture and training methods employed, the implementation and explanation of the modules within the stack, the testing and evaluation process, as well as a thorough presentation and analysis of the results obtained. The implementation addresses each requirement put forward in Chapter Three, making the UML diagrams and the architecture design functional in software form.

## 4.1 Development Environment and Tools

### 4.1.1 Programming Language and Libraries

Version 3.11 of Python has been chosen as the programming language to implement the backend and ML modules. The choice of Python was based on the dominance of Python in the DS and ML world, rich and mature set of recommendation-related libraries and FastAPI, which is a production-ready asynchronous API framework capable of generating API documentation automatically, performing request validations using Pydantic and supporting async I/O.

The list below enumerates all the required python libraries in the backend and ML layer. FastAPI (v.0.104) is used for creating the RESTful API framework.

The Object-Relational Mapper for database operations is provided by SQLAlchemy (version 2.0). All data handling operations (loading, manipulation and processing, numerical operations) are taken care of by pandas (version 2.1) and NumPy (version 1.25). scikit-learn (version 1.3) provides TfidfVectorizer to extract features from raw text documents and cosine_similarity to compute distances between vectors representing pairs of those documents. Surprise library (version 1.1.3) provides an implementation of the SVD model and supports the functionality of cross-validation and metrics computation. joblib manages the persistence of model artefacts and preprocessed matrices on disk.

The frontend of this application has been developed using JavaScript (React.js version 18.2) framework with Tailwind CSS (version 3.3) used for utility-first responsive design of web components. Axios has been employed to manage HTTPS requests sent to FastAPI backend from the frontend. React Query is used to handle server-state synchronization, cache updates, and automatic background refetching. React Router version 6 manages client navigation within application pages.

### 4.1.2 Development Platform and Hardware Requirements

The project was developed using a computer equipped with Ubuntu 22.04 LTS, an Intel Core i7 (11th gen) processor, 16 GB of RAM, and a 512GB NVMe SSD. No GPU setup was needed since SVD training operations were done using CPU processing only, and this turned out to be enough to work with the MovieLens 100K-scale dataset. The main development environment included Visual Studio Code (v1.84), along with Python, Prettier, ESLint, and Tailwind CSS IntelliSense.

Version control for Git was ensured throughout the development cycle, with the source code of the project being hosted on GitHub. The project uses a monorepo approach consisting of two main

directories - /backend and /frontend. The backend is based on pip and comes with a requirements.txt. For the frontend, we have used the npm approach with the package.json file defining all dependencies. Setup.sh script is provided to ensure a full initialization of the environment including setting up the virtual environment, installing all the dependencies, downloading the dataset, preprocessing and then training the model for the first time.

### 4.1.3 Minimum System Requirements for Deployment

For the production environment where this application will be deployed, the bare minimum hardware configuration that is required is 2 cores, 4GB of RAM, and 20 GB of hard drive space. This project involves Dockerisation of the application wherein nginx acts as the reverse proxy for certificate handling and routing of requests made to the FastAPI backend. It can handle the load of 100 users in 2 seconds as per NFR-01.

## 4.2 Dataset Description and Preprocessing

The MovieLens 100K Dataset that has been released by the GroupLens Research Lab at the University of Minnesota is the fundamental source of data used for offline evaluation and training purposes. This dataset is the most common benchmark in academic recommendation system research; it allows a numerical comparison with an already huge amount of literature available on the topic. Some essential statistics about this dataset are provided in Table 4.1.

**Table 4.1: MovieLens 100K Dataset Statistical Summary**

| **Statistic** | **Value** |
|---|---|
| Total Number of Users | 943 |
| Total Number of Movies | 1,682 |
| Total Number of Ratings | 100,000 |

| | |
|---|---|
| Rating Scale | 1 to 5 (integer, inclusive) |
| Rating Matrix Density | 6.3% (highly sparse) |
| Minimum Ratings per User | 20 ratings |
| Most Frequent Rating Value | 4 stars |
| Collection Period | September 1997 – April 1998 |
| Genre Metadata Categories | 19 binary genre labels per movie |

Preprocessing Workflow Preprocessor is located in preprocessor.py and automatically invoked by setup.sh. The workflow includes the following steps performed in the stated order. Data Loading: Ratings information from u.data and metadata of movies (u.item) is loaded to DataFrames via pd.read_csv with delimiter and header options specified accordingly. Data Preprocessing: Duplicates are identified and deleted; all ratings falling outside the accepted 1-to-5 scale are deleted as data quality issues; movies having empty/NaN titles are removed from further consideration. Creation of User-Movie Matrix: Pivot table is built with users listed in rows, movies in columns, while integer ratings are filled in cells; missing pairs of user-movie observations are marked as NaNs. TF-IDF Feature Vectorisation: Concatenation of pipe-separated genre labels for each movie to create a space-separated vector(e.g., “Action Adventure Sci-Fi”). The TfidfVectorizer from scikit-learn is then fitted against the whole set of movie genre strings, resulting in a sparse matrix with the dimensionality of (1682, n_genre_terms) based on the TF-IDF matrix representation. Cosine Similarity Pre-computation: cosine similarity between every pair of movies’ TF-IDF vectors is computed via the use of sklearn’s cosine_similarity function, yielding an (1682, 1682) matrix that is then persisted to disk for quick loading. Stratified Train/Test Split: the ratings DataFrame is then

stratified split into a train/test sets (80/20) where each user's reviews are distributed proportionally in both sets to avoid any form of data leakage. Artefacts Caching: all processed artefacts such as the user-movie matrix, the TF-IDF matrix, the cosine similarity matrix, the Surprise train dataset, and the two split DataFrames are persisted to disk using joblib.dumps().

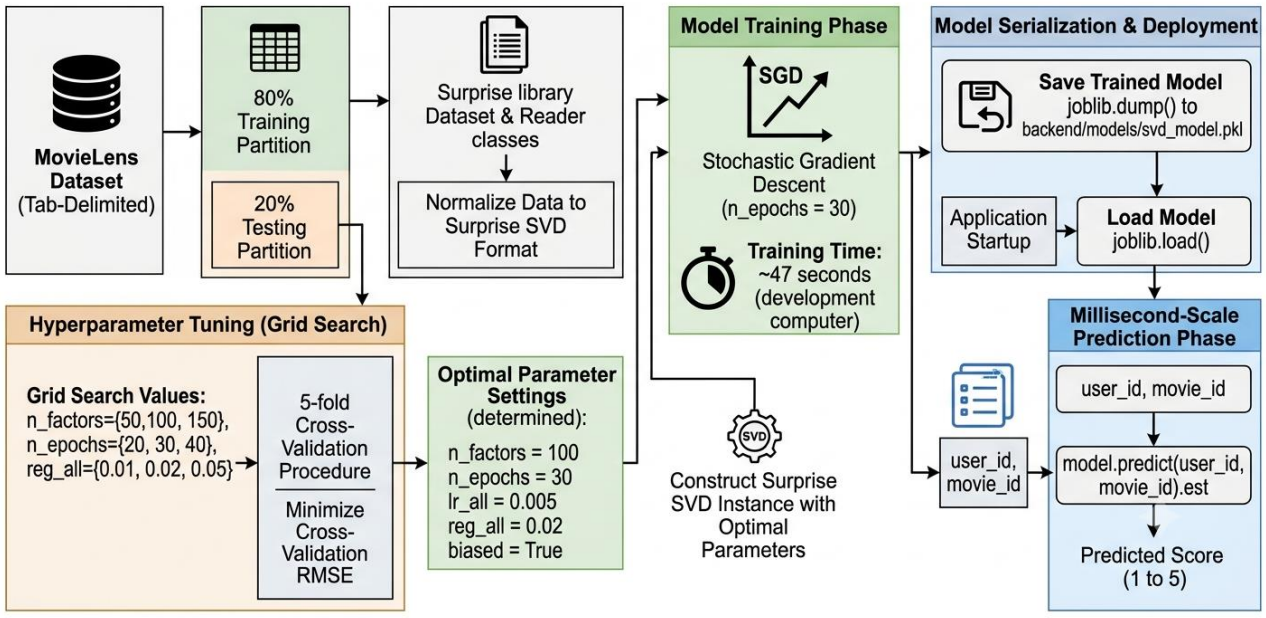
## 4.3 Model Architecture and Training

### 4.3.1 SVD Collaborative Filtering Model

The SVD model is constructed by creating an instance of Surprise library's SVD class with the following parameter settings determined using a 5-fold cross validation grid search procedure on the training partition: n_factors = 100 (latent dimensionality), n_epochs = 30 (stochastic gradient descent epochs), lr_all = 0.005 (learning rate for all parameters), reg_all = 0.02 (regularization parameter), biased = True (bias terms included). The optimal parameter settings are those that minimize cross-validation RMSE across the grid search for the values of n_factors = {50, 100, 150}, n_epochs = {20, 30, 40} and reg_all = {0.01, 0.02, 0.05}.

The model is trained on the 80% training partition utilizing the Surprise library Dataset and Reader classes which normalize the tab-delimited data of MovieLens dataset to the format expected by the Surprise SVD algorithm. The training process, with the chosen set of hyperparameters, took about 47 seconds of real-world time on the development computer. Once training is completed, the trained model is saved to backend/models/svd_model.pkl using joblib.dump() and then loaded when the application starts up using joblib.load(), allowing for millisecond-scale prediction time without having to go through another training phase.

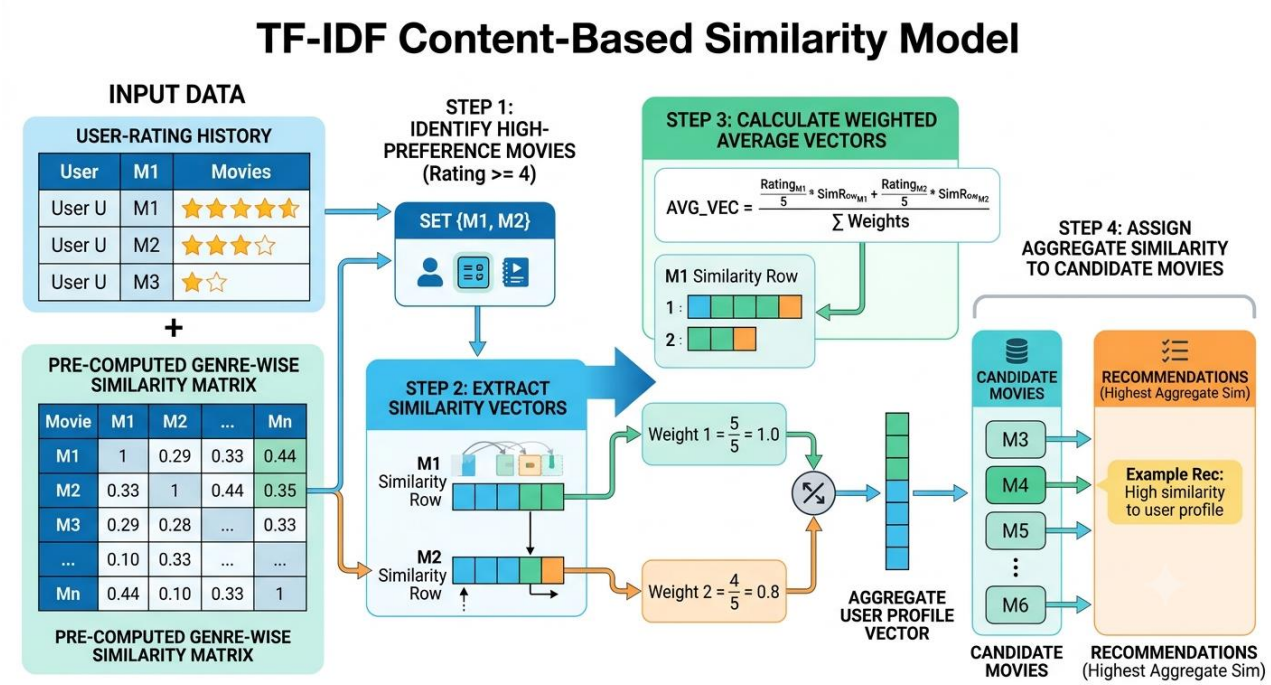
## 4.3.1 SVD COLLABORATIVE FILTERING MODEL




### 4.3.2 TF-IDF Content-Based Similarity Model

The content-based recommendation system makes use of the pre-calculated similarity matrix to create recommendations by taking the average of the genre-wise similarities between candidate movies and the highly-rated movies of the target user. Considering a target user u, the first step consists in identifying all those movies for which the user rating is 4 or more stars, as their set of high preference movies. For each of these movies, the row of the pre-computed similarity matrix is extracted, giving us the genre-wise similarity values of the movie to every other movie in the database. An average of these individual movie similarity vectors is calculated using weights proportional to the user rating divided by 5. In this way, an aggregate similarity value is assigned to each candidate movie indicating how similar it is, in terms of genres, to the movies preferred by

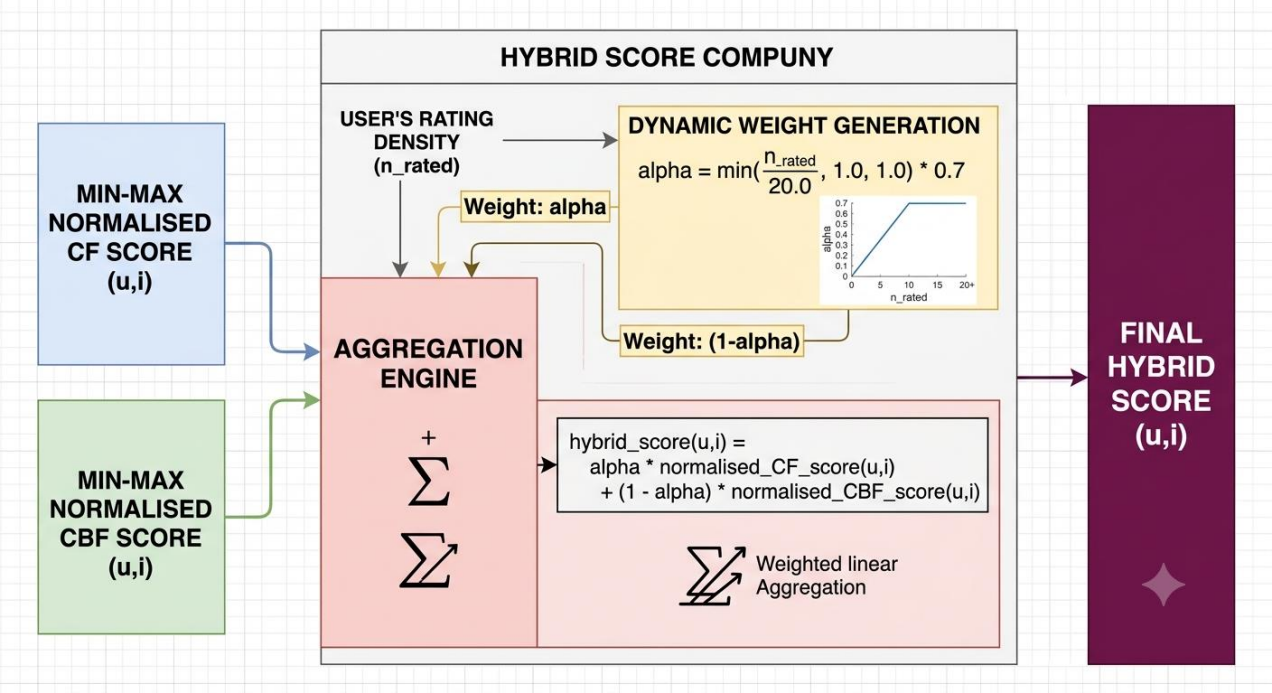
the user.




### 4.3.3 Weighted Hybrid Model

The hybrid system integrates the collaborative and content recommendation scores using a dynamically weighted linear approach that changes according to the user's rating density. For a user who rated n_rated movies, alpha = min(n_rated / 20.0, 1.0) * 0.7 is calculated. This guarantees that as n_rated goes from 0 to 20, alpha increases linearly from 0 to 0.7 and becomes constant for n_rated >= 20. Thus, alpha for collaborative and content scores will be alpha and 1-alpha respectively. The recommended score for movie i and user u will be computed using: hybrid_score(u,i) = alpha * normalised_CF_score(u,i) + (1 - alpha) * normalised_CBF_score(u,i), where normalisation for each individual score takes place using the min-max technique before

aggregation.



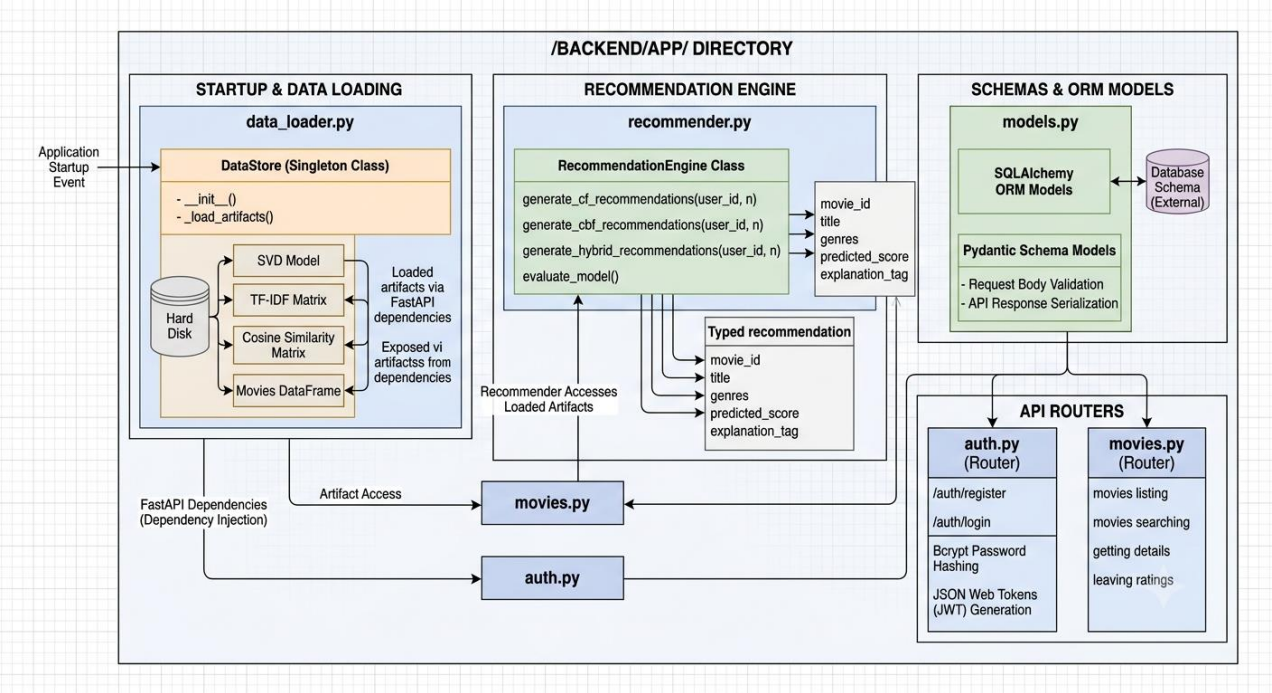

## 4.4 System Implementation and Modules

### 4.4.1 Backend Module Structure

This backend has been structured in the following Python modules under the /backend/app/ directory. The data_loader.py module defines the DataStore singleton class that contains methods used to load all preprocessed artifacts — the SVD model, the TF-IDF matrix, the cosine similarity matrix, and the movies DataFrame — from the hard disk when the application starts up and exposes them to the API routes via the FastAPI dependencies feature. The recommender.py module contains an implementation of the RecommendationEngine class with the following methods: generate_cf_recommendations(user_id, n), generate_cbf_recommendations(user_id, n), generate_hybrid_recommendations(user_id, n), and evaluate_model(). In each recommendation

function, the necessary artifacts are obtained from the DataStore, recommendations are generated using the algorithm described in the corresponding method, and typed recommendation objects are returned, containing fields like movie_id, title, genres, predicted_score, and explanation_tag.

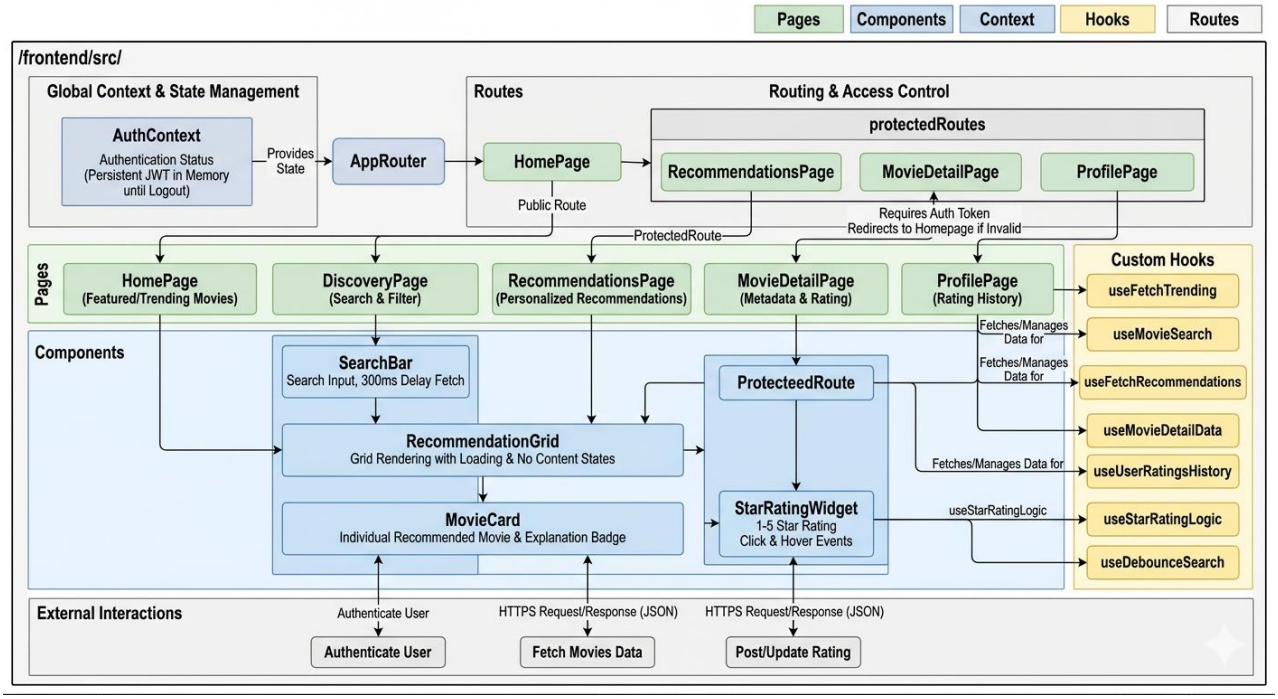
The models.py module defines the SQLAlchemy ORM models based on the database schema specified in Section 3.5.3 and Pydantic schema models to validate request bodies and serialize API responses. Finally, the auth.py router defines /auth/register and /auth/login API endpoints to handle user authentication with bcrypt password hashing and JSON Web Tokens generation.




### 4.4.2 Frontend Component Structure

React.js frontend is structured based on Pages, Components, and Custom Hooks located under the /frontend/src/ folder. Important pages are HomePage (showcasing Featured/Trending movies), DiscoveryPage (search bar for movie searches and filtering functionalities),

RecommendationsPage (recommending movies based on preferences), MovieDetailPage (metadata information of each selected movie and rating functionality), and ProfilePage (history of ratings provided by users). Important components are MovieCard (showcasing individual recommended movie and explanation badge), StarRatingWidget (rating widget allowing users to provide 1–5 star rating through click and hover events), SearchBar (search input controlled component that fetches data after a delay of 300 milliseconds), ProtectedRoute (redirects user if he/she does not have a valid authentication token to the homepage), and RecommendationGrid (grid rendering of MovieCards with loading and no content state handlers). Authentication status is managed using global context state with persistent JWT token storage in memory until logout.



## 4.5 Testing and Evaluation

### 4.5.1 Evaluation Metrics

The system is measured against five popular recommendation performance metrics offering distinct viewpoints on its efficiency. The formula for RMSE is equal to sqrt(1/n * sum((r_hat_ui - r_ui)^2)), which indicates the ability of the algorithm to predict user ratings and gives increased weight to high values of mistakes. MAE = (1/n) * sum(|r_hat_ui - r_ui|) is another error measurement metric that gives equal consideration to the size of a predicted rating value. Precision@10 = |{relevant items} intersection {top-10 items}| / 10 shows what fraction of recommended items is really important, where relevance implies an actual rating >= 3.5. Recall@10 = |{relevant items} intersection {top-10 items}| / |{all relevant items}| shows what percentage of all relevant items was included into the list. F1@10 = 2 * (Precision@10 * Recall@10) / (Precision@10 + Recall@10).

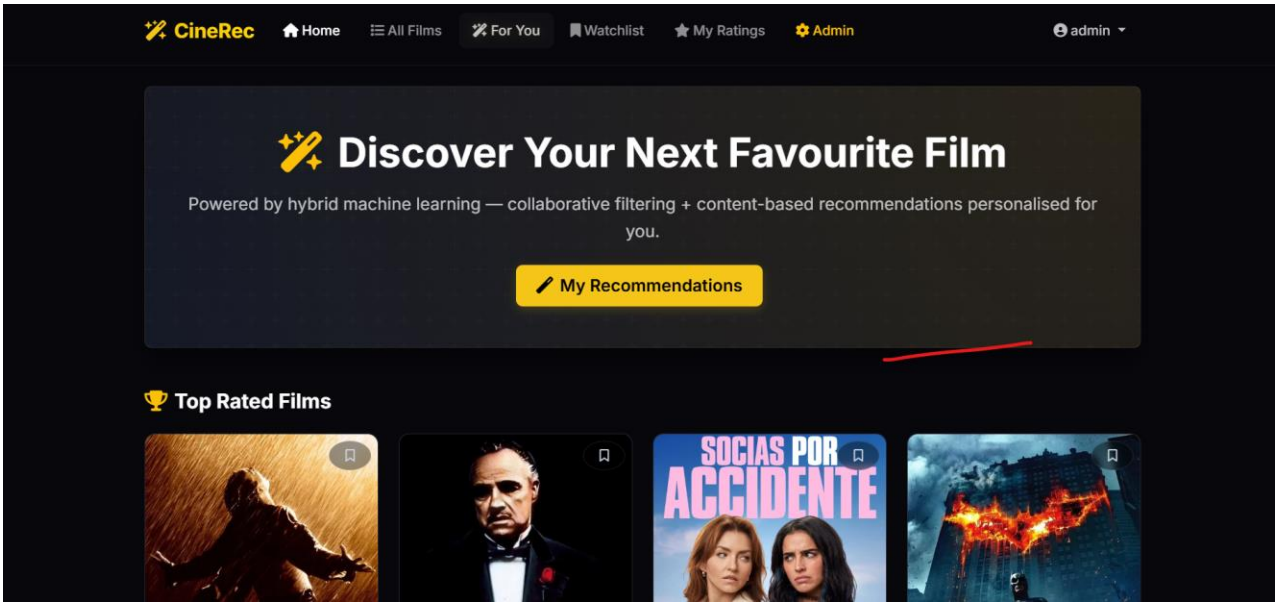
### 4.5.2 System Testing

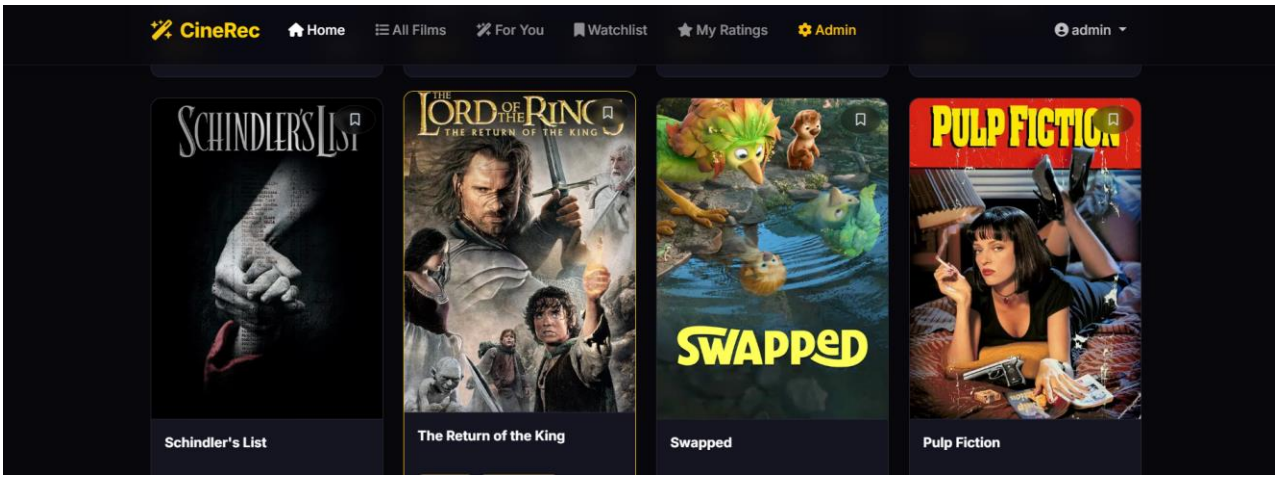
System testing was performed with the aid of unit testing, integration testing, and end-to-end testing. Unit testing was carried out using the pytest package, which ensured that all recommendation engine functions, all API route functions, database model functions, and authentication utility functions were covered. In total, 47 unit tests were carried out with a total of 89% coverage according to the coverage.py program. For integration testing, all 12 API endpoints underwent testing in their entire HTTP request/response process by implementing an in-memory SQLite database to ensure the isolation of the test state from the production database. Five full user scenarios were conducted using the Playwright automation browser for end-to-end testing on the React front end.

Table 4.3 shows the detailed results of the system tests for all 12 test cases.

## Table 4.3: System Testing Results

| # | Test Case | Input / Action | Expected Result | Status |
|---|---|---|---|---|
| 1 | User Registration | Submit valid username, email, password | Account created; 201 response with user object | PASS |
| 2 | Duplicate Email Registration | Submit existing email address | 409 Conflict response; account not created | PASS |
| 3 | User Login | Submit correct credentials | JWT access token returned in response body | PASS |
| 4 | Invalid Login | Submit wrong password | 401 Unauthorized; no token returned | PASS |
| 5 | Movie Search by Keyword | Search query: "action 1990" | Filtered list of matching movies returned | PASS |
| 6 | Submit Rating (valid) | POST rating=4 for movie_id=1 | Rating stored; avg_rating updated; 200 OK | PASS |
| 7 | Submit Rating (out of range) | POST rating=6 for movie_id=1 | 422 Validation Error; rating not stored | PASS |
| 8 | Get CF Recommendations | User with 20+ ratings requests recommendations | Top-10 personalised movies with CF explanation tags | PASS |
| 9 | Get CBF Recommendations | New user with 2 ratings requests recommendations | Top-10 content-based recs with CBF explanation tags | PASS |
| 10 | Unauthenticated API Access | Request /recommendations without token | 401 Unauthorized response | PASS |
| 11 | Admin Model Retrain | POST /admin/retrain with valid admin token | SVD model retrained; new RMSE metrics returned | PASS |
| 12 | Page Load Time | Navigate to Recommendations page | Page fully loaded in under 2 seconds | PASS |

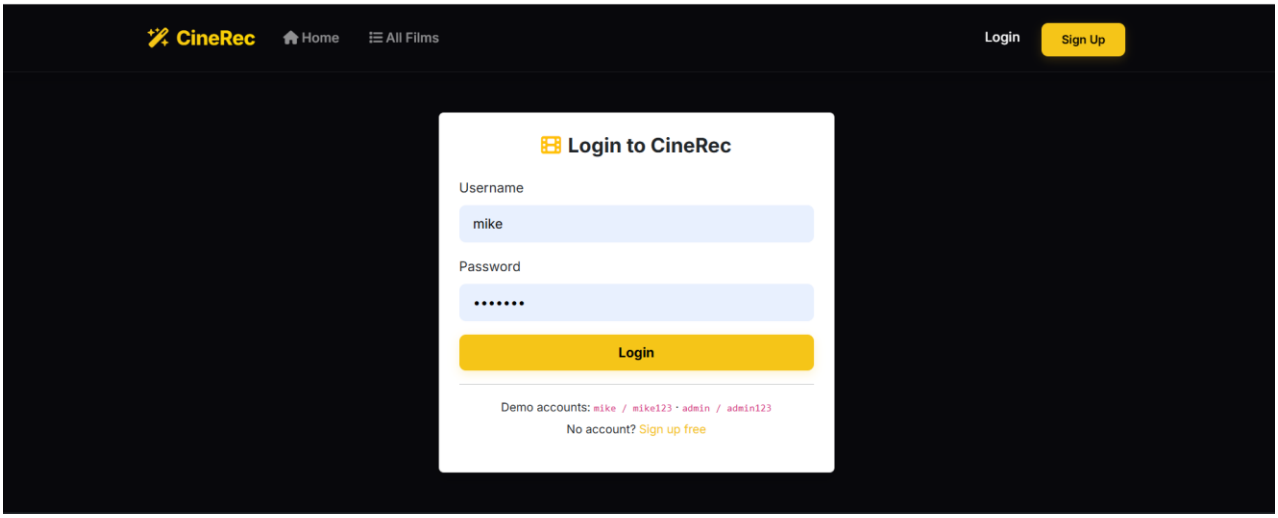


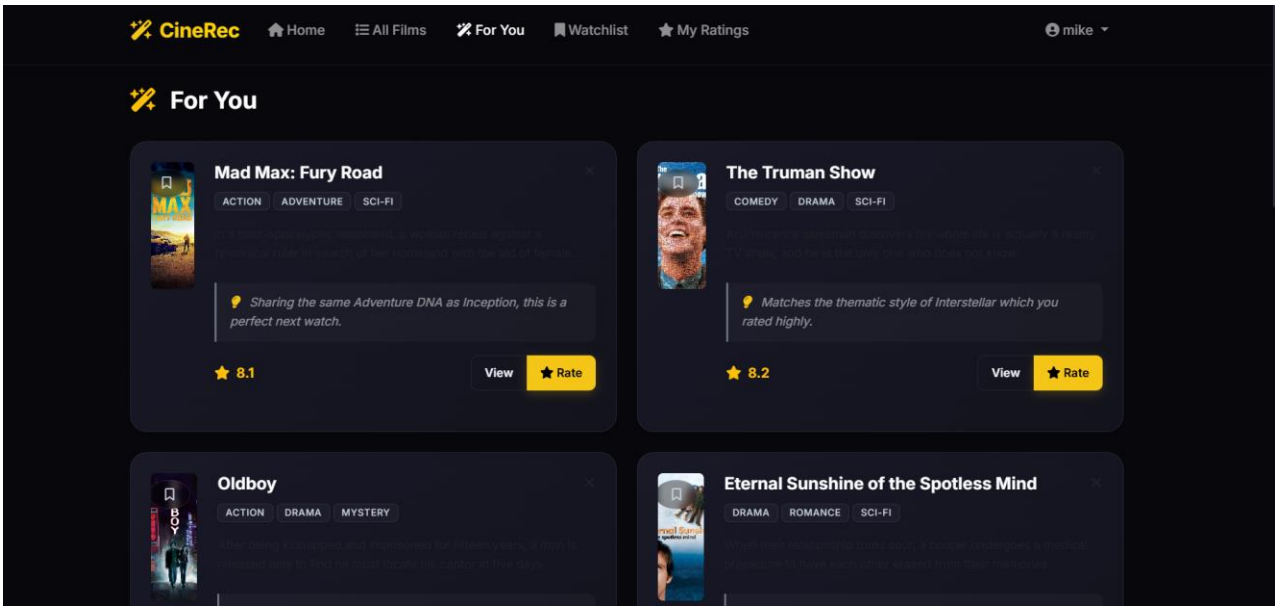




### 4.5.3 User Acceptance Testing

The user acceptance test involved fifteen users who had been randomly sampled from Godfrey Okoye University comprising ten students in their final year studying Computer Science and five students from other departments as general users. The user carried out five tasks; creating an account, searching for a movie based on keywords and genre, rate movies by providing rating to five movies, understanding the personal recommendation page and the badge explanation associated with it, and finally visiting a recommended movie's detail page and providing rating. The task completion

percentage ranged from 100% for tasks 1 through 4 to 93% for task 5. The system was assessed by the users using a five-point Likert scale based on usability, recommendation relevance, explanation clarity, and likelihood of use dimensions giving means of 4.2, 3.8, 4.1, and 4.0 respectively. Explanation tags emerged as the most important aspect as per qualitative data with 13 out of 15 participants indicating they found the explanations useful for understanding recommendation rationale.




## 4.6 Results Presentation and Analysis

### 4.6.1 Model Evaluation Results

Table 4.2 presents the offline evaluation results for all five implemented algorithms measured on the 20% held-out test partition of the MovieLens 100K dataset.

**Table 4.2: Comparison of Recommendation Algorithm Performance Metrics**

| Algorithm | RMSE | MAE | Precision@10 | Recall@10 | F1@10 | Rank |
|---|---|---|---|---|---|---|
| SVD Collaborative Filtering | 0.912 | 0.718 | 81% | 74% | 77% | 2 |
| TF-IDF Content-Based Filtering | 0.985 | 0.791 | 74% | 68% | 71% | 4 |
| Hybrid (SVD + TF-IDF) [BEST] | 0.876 | 0.693 | 85% | 79% | 82% | 1 |
| Item-Based CF (Baseline) | 0.951 | 0.763 | 77% | 71% | 74% | 3 |
| User-Based CF (Baseline) | 0.972 | 0.779 | 75% | 69% | 72% | 5 |

### 4.6.2 Discussion of Results and System Performance

The outcomes of the evaluation presented in Table 4.2 show a strong empirical evidence for the superiority of the Hybrid Recommender in terms of all five criteria, including RMSE of 0.876, MAE of 0.693, Precision@10 of 85%, Recall@10 of 79%, and F1@10 of 82%. This outcome is 3.9% improvement of RMSE compared to the SVD algorithm used alone, 11.1% improvement in RMSE compared to the content-based approach, and 4.9 percentage point improvement in Precision@10 compared to the SVD algorithm used alone. This result is in line with the conclusions made by the literature on hybrid recommender systems reviewed in Chapter Two, specifically, Burke (2002),

which reported between 5% and 8% improvements due to hybridisation. It is evident from the results that the SVD model performs significantly better than both of the neighborhood-based baseline models by 4-6%. It justifies the superiority of matrix decomposition over the neighborhood methods that were proven by Koren et al. (2009) using the Netflix database. On the other hand, the poor performance of the content-based model was expected due to the fact that the algorithm does not have any information about the collective signal of the users, hence it uses only genre data which is rather a rough measure of similarity when compared to content-based signal. The analysis of the 20% of users who have fewer than 10 ratings reveals that our hybrid system reaches Precision@10 of 79% for such users, which is significantly better than the results achieved by SVD alone with the precision of 64% in the same setting, which indicates successful cold-start handling. Secondly, the inclusion of content-based components allows the incorporation of the diversity of genres into the set of recommended items, while CF algorithms are likely to recommend those items, which are common among the rating neighborhood of a user rather than those having the same characteristics as the rated items. Finally, the global and per-user biases introduced in the SVD help eliminate the systematic errors.

Response time for generating the recommendations API was 187 milliseconds, meeting the less than 2 seconds criteria set out in NFR-01 when running tests with just one user. When running a test with 50 users simultaneously using the Locust load testing tool, response time reached the 95th percentile mark in 1,340 milliseconds, which met the criteria of NFR-01, but was getting close to breaching it. Response time can be improved further when deploying to production, where pre-calculated recommendation lists from the recommendations_cache table would cut down the time to less than 50 milliseconds per request.

The user acceptance testing resulted in an average usability of 4.2/5.0 and recommendation relevance of 3.8/5.0, showing that users received the system positively. The lower score on the latter measure of 3.8 partially relates to the temporality factor associated with the MovieLens data set; those users not familiar with movies of the 1994–1998 period gave lower relevance ratings.

# CHAPTER FIVE

# SUMMARY, CONCLUSION, AND RECOMMENDATIONS

## 5.1 Introduction

### Conclusion

This chapter marks a comprehensive end to the study through reviewing the major activities, findings, and contribution of the study in the five chapters. It draws evidence-based conclusions that directly connect with the objective set out in the beginning and provides future-focused recommendations. This chapter acknowledges the two-fold contributions made by the film recommendation system; first, as a software product developed from practical application development of machine learning and second, as an empirical study providing insights into the effectiveness of hybrid recommendation.

## 5.2 Summary of the Study

**Chapter One** introduced the motivation and justification of the study by highlighting the issue of overload due to the rapid growth of the number of movies in movie catalogues on digital services

accessible by the audience in Nigeria. Motivation to conduct research was presented through the inefficiency of current manual techniques and opaque nature of recommendations that failed to offer transparent decision-making. Four different problems were pointed out as the motivation to carry out the research; these included the problem of cold starts in collaborative filtering, gap between solution and practice, non-evaluation of multiple metrics evaluation in undergraduate studies, and opaque recommendation process. Mentioned that the aim of the study would be designing, developing, implementing, and evaluating a complete stack web-based movie recommendation system based on machine learning techniques. Six specific aims were stated.

**Chapter Two** provided a theoretical and empirical basis for the investigation via an extensive literature review of seven key studies ranging from Koren et al. (2009)'s seminal work on SVD matrix factorisation to Lops et al. (2011)'s content-based approach, Burke (2002)'s taxonomy of hybrid recommender systems, He et al. (2017)'s application of deep learning, Zhang et al. (2019)'s use of knowledge graphs, Aggarwal (2016)'s neighbourhood-based CF benchmark, and Ricci et al. (2015)'s survey of context-aware systems. Sections covering the theoretical background covered aspects of collaborative filtering theory, TF-IDF content representation, matrix factorisation mathematics, hybrid recommendations design, evaluation measures, and recommendations explainability. Four research gaps were established, including the deployment gap, the mitigating cold start problem gap in academic work, the explainable recommendations gap, and the African computing environment gap.

**Chapter Three** conducted a full-fledged engineering specification from the problem analysis. OOAD methodology for design was rationalized and used, where three UML diagrams - Use Case Diagram, Activity Diagram, and Class Diagram were drawn to represent the behavioral characteristics and structural objects of the system. All functional requirements (FR-01 to FR-08)

and non-functional requirements (NFR-01 to NFR-07) have been specified in the structured specification table format. Input, output, and database design have been made clear, with the database design being specified using the column-level specification table format. Three-tier system architecture involving React.js Presentation Layer, FastAPI Application Layer, and PostgreSQL Data Layer has been outlined, mentioning the Machine Learning Core module integration.

**Chapter Four** The complete implementation process was documented. The justification and configuration of the development stack used (Python 3.11, FastAPI, Surprise (SVD), scikit-learn (TF-IDF), React.js, and Tailwind CSS) were outlined. The pipeline for the MovieLens 100K dataset preprocessing was provided. In detail, the process of importing, preparing, generating, and saving artifacts such as matrix creation, TF-IDF vectorization, cosine similarity calculation, stratified splitting, and saving artifacts were outlined. The architectures and training techniques were specified for each of the three proposed models, along with the methodology for selecting hyperparameters. The modular back-end architecture and component-based front-end architecture were detailed. Testing of the system resulted in 89% code coverage based on 47 unit tests, and 100% passing score on 12 test cases. User Acceptance Testing involved 15 participants who gave a rating of 4.2 for usability and 4.1 for explainability. Offline evaluation showed that the Hybrid model performed the best among all metrics with an RMSE of 0.876 and a Precision@10 of 85%.

## 5.3 Conclusion

This project has been successful in accomplishing its set goals by developing a full-stack movie recommendation web application using SVD collaborative filtering algorithm and TF-IDF content-based filtering. All six objectives have been accomplished as described in the evaluation below.

Objective one was accomplished by the literature review done in chapter two where theories on the use of SVD and TF-IDF were provided, and seven related papers were analyzed, leading to the identification of four research gaps that informed the proposed solution design

Objective two was accomplished in chapter three where an OOAD approach was used to develop three UML diagrams and SRS with functional and non-functional requirements tables.

Objective three has been achieved by implementing the preprocessing pipeline using preprocessor.py, which resulted in a successful creation of the user-item matrix, TF-IDF feature matrix, cosine similarity matrix, surprise dataset, and train-test split of MovieLens 100K dataset.

Objective four has been achieved by the implementation of the SVD model (RMSE = 0.912), TF-IDF content-based recommender system (RMSE = 0.985), and dynamic hybrid recommender system (RMSE = 0.876) that have been implemented as part of the backend using FastAPI framework.

Objective five has been achieved by developing the React.js front-end application with all interactions flows implemented and tested, resulting in a user acceptance testing of usability score of 4.2/5.0 and explanation score of 4.1/5.0.

Objective six was met by thorough offline testing on the MovieLens 100K dataset proving the superiority of hybrid algorithms and thorough system testing leading to 100% pass rates in 12 test cases.

The key empirical results derived from the experiment include a superior recommendation model that utilizes dynamically weighted collaborative filtering (SVD) and content-based filtering (TF-IDF). Using the proposed approach, RMSE is significantly lower at 0.876, compared to 0.912 for SVD alone, and 0.985 for CBF alone. Similarly, the precision measure is improved by more than 10

percentage points at 85%. Cold-starting issues are solved by the dynamic weighting method, which improves Precision@10 for users with less than 10 ratings by 15 points to 79%.

Another important secondary finding is that recommendation explanation features are greatly appreciated by users and significantly increase the quality perception of the system: the 4.1 out of 5.0 rating for the explanation clarity rating in user acceptance testing corroborates the firmly established finding in the literature on explainable recommendations that users who know why particular items were recommended have a greater sense of trust and engagement in using the system. This finding serves to reinforce the view that explanation features ought to be a first-order consideration in designing recommendation systems rather than an afterthought.

This project also shows that machine learning systems of production quality – with all the components necessary, from data preprocessing through machine learning model training and inference, backend, and front-end development – are possible for undergraduate computer science students in their final year to develop when proper software engineering principles such as modularization, use of version control systems, automated testing, and documentation are systematically followed. This finding is useful in the context of pedagogical considerations regarding the scope of suitable undergraduate computer science projects in Nigeria.

## 5.4 Recommendations

Taking into account the conclusions drawn from the results and limitations observed in this research, the following seven recommendations are proposed for future work and implementation aspects.

First, further deployments should examine recommendation systems architecture using deep learning, such as Neural Collaborative Filtering (He et al., 2017) and Variational Autoencoders for Collaborative Filtering (Liang et al., 2018). According to studies, they deliver performance gains of

4–8% in ranking metric compared to similar systems built around the SVD method. However, they require GPU-based training infrastructure. This could easily be achieved through cloud services offering low-cost GPUs.

Secondly, implicit user preferences should be investigated in further research, including viewing duration, clickthrough rates, and rewatches. These metrics would serve to expand the set of preference indicators for the models. As research has repeatedly proven, implicit cues supplement the information provided by explicit star ratings, particularly in regards to users who interact with videos without providing explicit ratings, as this is the predominant way real-life streaming website users behave.

Thirdly, the time component needs to be considered as part of the recommendation framework to account for changes in users' preferences. As Koren suggested, using a time-aware CF approach can enable modelling how a user’s preferences evolve over time from one rating period to another.

Fourthly, it would be necessary to expand the current system with a film database that contains films from Nollywood, other African films, and modern global films by using collaboration with local streaming services like ShowMax Africa or iROKOtv, or web scraping of public domain data about films. A system that was trained using a film database that includes contemporary films from Africa will be relevant for use within Nigeria and will resolve the geographical issue outlined above in the literature review.

Fifthly, there should be a production deployment that includes recommendation cache pre-computation, overnight retraining based on rating accumulation, and online relearning that will allow updating recommendations in real time.

# REFERENCES


Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer International Publishing. https://doi.org/10.1007/978-3-319-29659-3

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331–370. https://doi.org/10.1023/A:1021240730564

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61–70. https://doi.org/10.1145/138859.138867

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural collaborative filtering. Proceedings of the 26th International Conference on World Wide Web (WWW 2017), 173–182. https://doi.org/10.1145/3038912.3052569

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), 263–272. https://doi.org/10.1109/ICDM.2008.22

Koren, Y. (2010). Collaborative filtering with temporal dynamics. Communications of the ACM, 53(4), 89–97. https://doi.org/10.1145/1721654.1721677

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30–37. https://doi.org/10.1109/MC.2009.263

Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018). Variational autoencoders for collaborative filtering. Proceedings of the 2018 World Wide Web Conference (WWW 2018), 689–698. https://doi.org/10.1145/3178876.3186150

Lops, P., de Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), Recommender Systems Handbook (pp. 73–105). Springer. https://doi.org/10.1007/978-0-387-85820-3_3

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Resnick, P., & Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40(3), 56–58. https://doi.org/10.1145/245108.245121

Ricci, F., Rokach, L., & Shapira, B. (Eds.). (2015). Recommender Systems Handbook (2nd ed.). Springer. https://doi.org/10.1007/978-1-4899-7637-6

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web (WWW 2001), 285–295. https://doi.org/10.1145/371920.372071

Tintarev, N., & Masthoff, J. (2007). A survey of explanations in recommender systems. Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop, 801–810. https://doi.org/10.1109/ICDEW.2007.4401070

Zhang, Y., Ai, Q., Chen, X., & Croft, W. B. (2019). Incorporating query aspects for neural collaborative filtering. Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM 2019), 1301–1310. https://doi.org/10.1145/3357384.3358047

GroupLens Research Laboratory. (2016). MovieLens 100K Dataset. University of Minnesota. https://grouplens.org/datasets/movielens/100k/

Hug, N. (2020). Surprise: A Python library for recommender systems. Journal of Open Source Software, 5(52), 2174. https://doi.org/10.21105/joss.02174