॥वसुधैव कुटुम्बकम्॥

DBMS PROJECT REPORT

MOVIE RECOMMENDATION SYSTEM

SUBMITTED BY

Aashmit McKenzie 22070122036

Dhruva Kashyap 22070122055

DEPARTMENT OF COMPUTER SCIENCE

SYMBIOSIS INSTITUTE OF TECHNOLOGY

INDEX

1. Introduction

In this age of entertainment consumption, navigating one's way can be a tedious task considering the vast array of movies available on the net. To address this challenge, we have created a movie recommendation system that integrates data from various leading platforms like IMDb, Rotten Tomatoes, and many others.

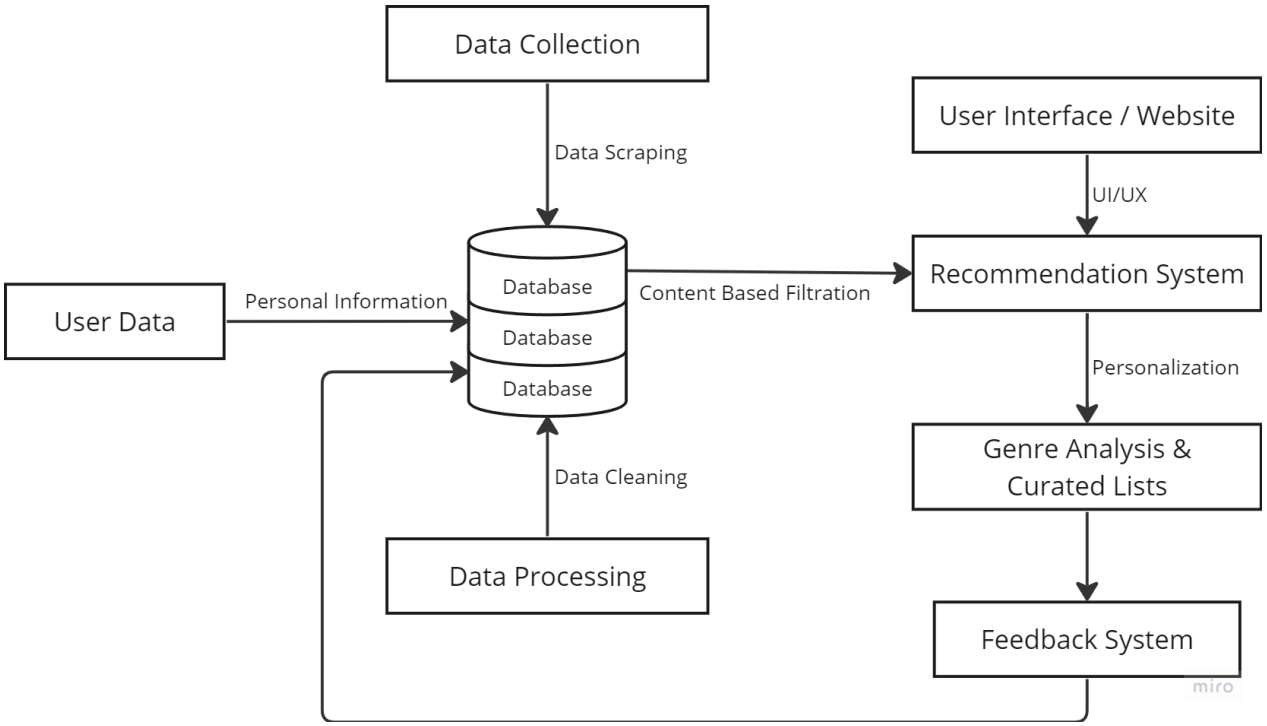
Our project aims to enhance the movie-watching experience by using various data management and analysis techniques to calculate average ratings and generate personalized recommendations based on user preferences and past ratings. By collecting and reviewing data from multiple reliable sources, our system provides users with a consolidated perspective on the quality and appeal of different movies.

Moreover, our system goes beyond simple recommendations by carefully selecting and creating lists of top-rated films within each genre. This helps to cater to the diverse preferences of movie enthusiasts. In this project, we want to show the use of database management and data analysis, with the help of recommendation algorithms to simplify the process of movie selection and elevate the enjoyment of cinematic offerings.

2. Problem Statement

In today's Digital era finding movies that suit our individual preferences is particularly challenging due to the number of options available and the lack of personalized recommendations. Existing sites often fall short in this aspect and fail to give an aggregate rating and a curated list of specific genres. By simplifying the process of discovering quality cinematic offerings, such a system aims to enhance the movie-watching experience for audiences worldwide.

3. System Architecture



4. Modules

1. Data Collection Module: Responsible for collecting data from various sources like IMDb Rotten Tomatoes and various other sites.

2. Database Management Module: Stores collected movie data in a relational database management system and manages entities such as movies, users, ratings, genres, etc.

3. Data Processing Module: Collects movie data to compute aggregate ratings and generate additional insights. Handles data cleaning, transformation, and feature engineering tasks.

4. Recommendation System Module: This is the main core component responsible for generating personalized movie recommendations for users. Implements collaborative filtering, content-based filtering, or hybrid recommendation.

5. User Interface Module: Provides a user-friendly interface for users to interact with the system.

6. Genre Analysis and Curation Module: Analyses movie genres and identifies top-rated movies within each genre. Curates lists of top-rated films.

7. Feedback and Evaluation Module: Collects user feedback and ratings to improve the accuracy of movie recommendations. Monitors user interactions with the system and identifies areas for improvement.

5. Functional Requirements

1. Movie Data Collection: The system should gather data from multiple sources, such as IMDb and Rotten Tomatoes, including movie titles, ratings, reviews, genres, directors, and actors.

2. Data Processing and Analysis: The system should process and analyze the collected data to compute aggregate ratings, identify trends, and extract relevant features.

3. Recommendation Generation: The system should generate personalized movie recommendations for users based on their preferences, historical ratings, and viewing history.

4. Genre Identification and Curation: The system should identify movie genres and curate lists of top-rated movies within each genre for user exploration.

5. User Interaction: Users should be able to input their preferences, provide feedback on recommendations, and explore top-rated movies within specific genres.
6. User Feedback and Evaluation: The system should collect and analyze user feedback and ratings to improve the accuracy of recommendations and the overall user experience.

7. User Interface: The system should provide a user-friendly interface for easy navigation, movie browsing, and interaction with the recommendation engine.

8. Integration with External Services: The system should integrate with external services for user authentication, data retrieval, and other functionalities as needed.

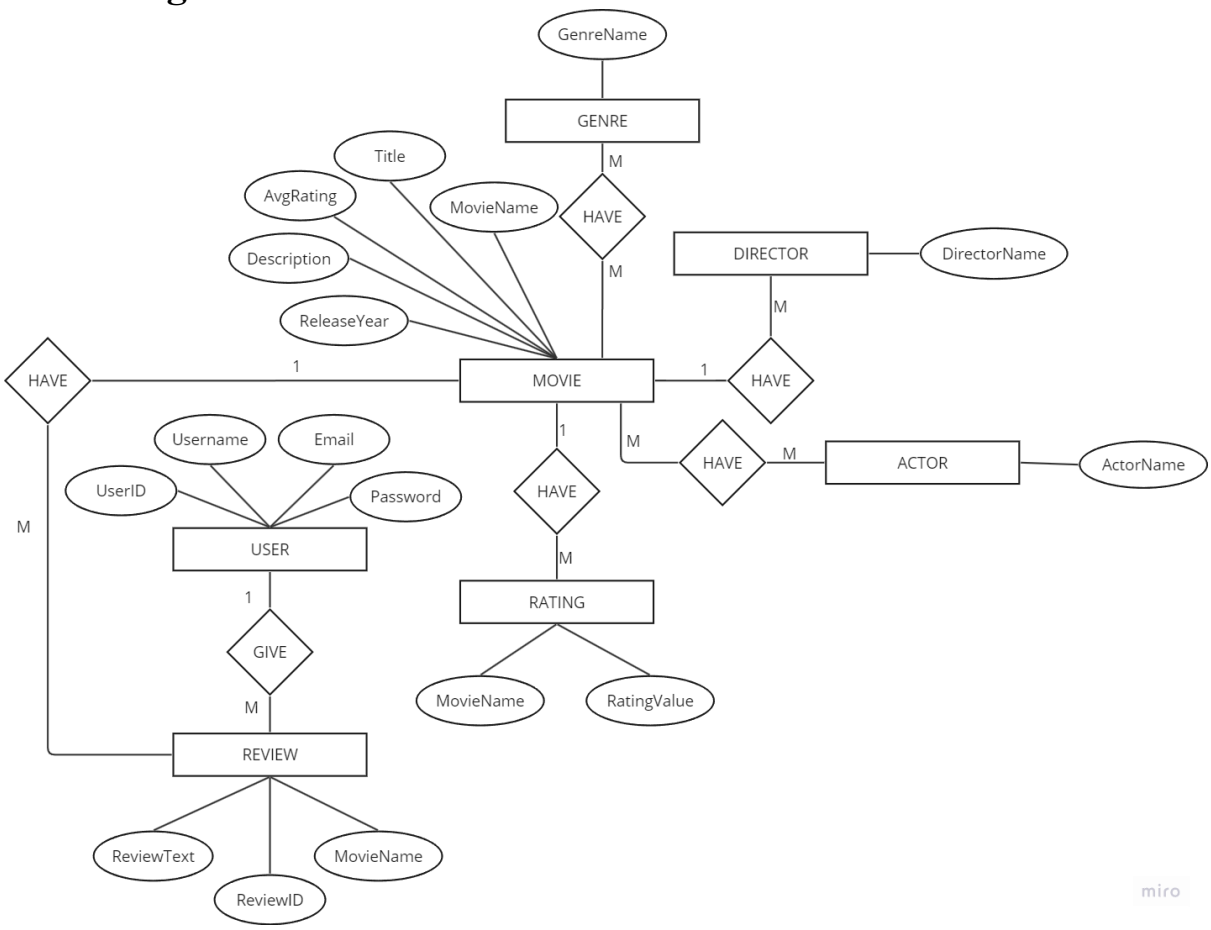
9. Performance and Scalability: The system should be capable of managing a large volume of movie data efficiently.

10. Security: The system should implement robust security measures to protect user data, prevent unauthorized access, and ensure data privacy and integrity.

6. Entities, Relationships and Attributes

Entity	Attributes	Relationship
Movie	ID, Title, Release Year, Description, Average Rating, Director, Actors, Genres.	Many-to-Many with Genre, Many-to-Many with Actor, One-to-Many with Director, One-to-Many with Rating.
User	ID, Username, Email, Password, Preferences.	One-to-Many with Review, One-to-Many with Preference.
Rating	ID, Movie ID, User ID, Rating Value, Timestamp.	Many-to-One with Movie.
Genre	ID, Name.	Many-to-Many with Movie.
Director	ID, Name.	Many-to-One with Movie.
Actor	ID, Name.	Many-to-Many with Movie.
Review	ID, Movie ID, User ID, Review Text, Timestamp.	Many-to-One with Movie, Many-to-One with User.

7. EER Diagram



8. Relation Schema

Primary Key – Underlined
Foreign Key – Italics

1. User

UserID	UserName	Email	Password

   - Primary Key – UserID

2. Movie

MovieName	Title	AvgRating	Description	ReleaseYear

   - Primary Key – MovieName

3. Rating

MovieName	RatingValue

   - Foreign Key – *MovieName*

4. Genre

GenreName

5. Director

DirecctorName

6. Actor

ActorName

7. Review

ReviewText	ReviewID	*MovieName*

   - Primary Key – ReviewID
   - Foreign Key – *MovieName*

9. Application of Codd's Rules

1. The Information Rule: All data must be stored in tables within the RDBMS.
   Applicable: The system stores all data in tables within the RDBMS, adhering to the relational model.

2. Guaranteed Access Rule: Each data element must be accessible by specifying the table name, primary key (or keys), and attribute name.
   Applicable: Users can access data elements by querying tables using SQL commands, specifying table names, primary keys, and attributes.

3. Systematic Treatment of Null Values: The RDBMS must support a systematic handling of NULL values to represent missing or unknown information.
   Applicable: NULL values represent missing ratings, reviews, or preferences for certain movies or genres.

4. Dynamic Online Catalog Based on the Relational Model: The RDBMS must provide an online catalog containing metadata about the database schema, relationships, constraints, and access privileges.
   Applicable: The RDBMS maintains a catalog containing information about tables, columns, data types, indexes, and constraints.

5. Comprehensive Data Sub-language Rule: The RDBMS must support a comprehensive data sub-language enabling users to define, manipulate, and query data using a standardized syntax.
   Applicable: SQL serves as the comprehensive data sub-language for defining database schema, manipulating data, and querying data.

6. View Updating Rule: Any view that is theoretically updatable must also be updatable through the system.
   Not Applicable: Views are theoretically updatable, but it depends on the complexity of the view and the underlying data.

7. High-level Insert, Update, and Delete: The RDBMS must support high-level insert, update, and delete operations allowing users to modify data at the table level without specifying low-level access paths.
   Applicable: Users can perform insert, update, and delete operations on tables using SQL commands without specifying low-level access paths.

8. Physical Data Independence: Changes to the physical storage structures of the database should not require changes to the application's data manipulation code.
   Not Applicable: Changes in physical storage structures may affect the application's data manipulation code.

9. Logical Data Independence: Changes to the logical structure of the database should not require changes to the application's data manipulation code.
   Not Applicable: Changes in the logical structure of the database may require changes to the application's data manipulation code.

10. Integrity Independence: Integrity constraints should be stored in the catalog and not be tied to application programs.
    Applicable: Integrity constraints are defined within the database and enforced by the RDBMS, independent of application programs.

11. Distribution Independence: Database distribution should not be visible to users and should not affect the conceptual schema.
    Not Applicable: The system may not involve database distribution, or it may not be transparent to users.

12. Non-subversion Rule: If a system provides a low-level (record-at-a-time) interface, it must also provide an equivalent high-level (set-at-a-time) interface that allows users to access the same data.
    Applicable: SQL provides both low-level and high-level interfaces for accessing data, allowing users to access data at different levels of granularity.