SECJ-1023   PT2

FINAL PROJECT (GROUP 13)

NAME	MATRIC NO.
Ali Reda Ali	A23CS4003
Mohammed Moqbel Ali	A23CS4016
Knan Fadi Dawarh	A23CS4013
Fouad Mahmoud Fouad	A23CS0017

Movie Recommendation System Proposal

Synopsis and General Idea

The Movie Recommendation System is designed to assist users in discovering new movies that match their tastes. It aims to make the process of finding movies more enjoyable through the use of sophisticated algorithms that improve over time. The system is intended to be user-friendly, ensuring that anyone can navigate and benefit from the recommendations. It will cater to a wide range of preferences, from blockbuster enthusiasts to indie film aficionados, ensuring that everyone finds something they love.

System Objectives and/or Purpose

1. **Help users discover new movies aligned with their tastes**: The system learns from user preferences to suggest movies they are likely to enjoy. By understanding the user's past interactions and preferences, it can provide highly relevant recommendations.

2. **Make finding movies easier and more enjoyable**: The interface and recommendation engine are designed to streamline the movie discovery process, reducing the time users spend searching for a movie to watch.

3. **Use clever computer techniques to improve the recommendations over time**: Machine learning algorithms analyze user interactions to refine recommendations continually. As more data is gathered, the system's accuracy and relevance improve.

4. **Create a simple and friendly system that anyone can use**: The design focuses on usability, ensuring a seamless experience for users of all ages. The interface will be intuitive, making it easy for even the least tech-savvy individuals to navigate and find movies they enjoy.

How to Use the System

1. **Sign Up**: Users create an account with a username and password. This process is quick and straightforward, requiring minimal information to get started.

2. **Tell Us What You Like**: Users select their favorite movie genres and actors during profile setup. This initial input helps kickstart the recommendation process.

3. **Explore Movies**: Users can search for movies or browse through the system's recommendations. The search function allows for filtering by genre, release year, and more.

4. **Rate and Review**: After watching a movie, users can rate it and leave a review. This feedback is crucial for the system to understand user preferences better.

5. **Get Personalized Suggestions**: The system utilizes user ratings and reviews to suggest movies they might enjoy. These suggestions become more accurate over time as the system learns more about the user's tastes.

6. **Stay Updated**: Users receive updates on new releases and personalized recommendations. Notifications can be customized to the user's preferences, ensuring they stay informed about movies that interest them.

Problem Statement

With the huge movie availability in these digital times, the huge choice becomes daunting for a viewer to find the movies that best fit his or her taste. Traditional ways of film discovery via scrolling through long lists or simple, generalized recommendations are usually unsatisfying in terms of user experience. As a result, the users become frustrated and start wasting time in search of those movies that best suit their tastes. What's more, a system that continues to learn and improves upon the set of recommendations by user feedback is needed to guarantee relevance and further upgrade the movie-viewing experience over time.

Solving the Problem and Contributing to Sustainable Development

Solving the Problem:

This is a movie recommendation system that solves the problem of overwhelming choices of movies through sophisticated machine learning algorithms that give personalized recommendations. The system, based on user preference analysis, past interactions, and ratings, tailors suggestions to suit individual tastes. This greatly reduces the time and effort spent by users in search of movies, hence enhancing their viewing experience.

Contribution to Sustainable Development:

The Movie Recommendation System contributes to sustainable development in cultural diversity and reduction of digital waste:

Cultural Diversity:

The system recommends movies individually to its users, fostering a tendency in them to watch more independent and foreign films. Exposure to this type of diverse content will help to strengthen cultural appreciation and understanding.

Digital Waste Reduction: An efficient set of recommendations reduces the time users spend browsing, in turn reducing the digital carbon footprint from long online searches. What's more is that the fact it refines its suggestion over time ensures the system keeps providing increasingly relevant content for the users and makes their activity on the internet more streamlined.

System Benefits

Better User Experience:

Personalization: I guarantee movies suggested in accordance with his/her unique taste for a more delightful and relevant viewing experience for each individual user.

User-Friendly Interface: Easy navigation through the system to find movies according to choose, spite of age and any technical level.

Time Efficiency:

Quick Discovery:

Effective recommendation engine that takes less of the end-user's time searching for movies.

Streamlined Process: The process comprises only information and user feedback, optimizing recommendations so that movie discovery can be done quicker, yet more effectively.

Continuous Improvement:

Learning Algorithms: Not only does this enable machine learning from user interactions, thereby improving upon its recommendations with time—increasingly high in terms of accuracy and relevance—but user feedback in the form of ratings and reviews also gets fed back into the system. This refines its capability to predict and cater to individual tastes. Expanded Film Investigation:

Diverse Recommendations: Exposure to more movies, in particular, hidden gems, enables a user to appreciate different genres and styles of filmmaking. It furthers independent filmmaking and allows the user to break out of the mainstream blockbuster mindset. Engagement and Retention:

Increased engagement on account of a personalized and easy-to-use approach, enhancing user engagement and satisfaction.

Retention: The users will return and build a loyal user base, hence creating a community of movie buffs.

Reporting

We will track the following metrics to evaluate and improve the system:

1. **Accuracy of Recommendations**: Comparing system suggestions with user preferences and ratings. This helps us measure how well the system understands and predicts user tastes.

2. **User Interaction Frequency**: Monitoring how often users interact with the system and rate movies. Higher interaction rates indicate higher engagement and satisfaction.

3. **Trends in User Preferences**: Analyzing user data to identify trends and enhance recommendations. This includes tracking popular genres, actors, and viewing times.

4. **User Feedback**: Collecting feedback to identify areas for improvement. User suggestions and complaints will be used to make iterative improvements to the system.

5. **Overall User Satisfaction**: Measuring user satisfaction to ensure a positive experience. This will be assessed through surveys and feedback forms.

Conclusion

Our project aims to simplify the process for movie lovers to find their next favorite film. By leveraging user preferences and sophisticated algorithms, we aspire to create a fun and personalized movie-watching experience for everyone. Our ultimate goal is to enhance movie nights and make them more enjoyable for all users. We believe that by continually learning from user interactions, we can offer increasingly accurate and satisfying recommendations, making our system an indispensable tool for movie enthusiasts.
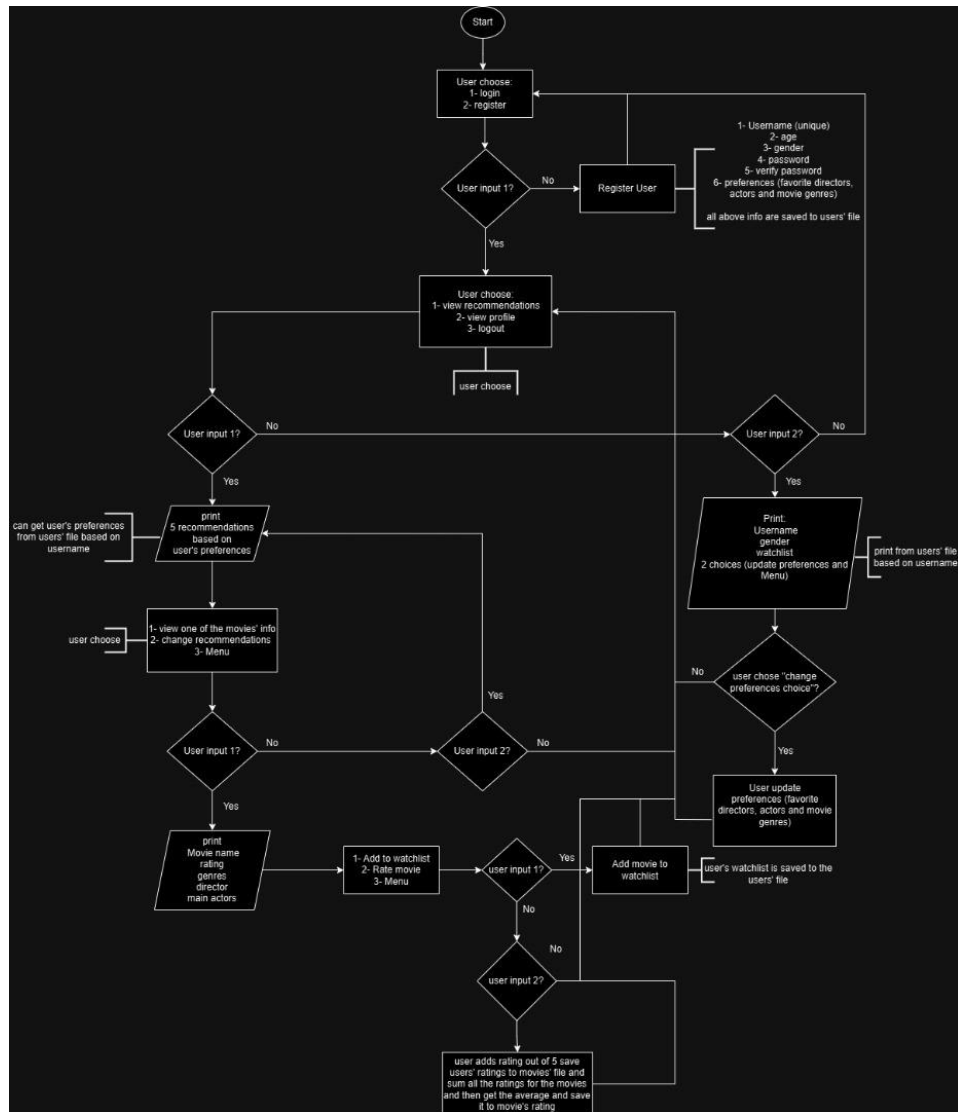
The following flowchart describes the interaction flow of a movie recommendation system. The following description describes each step in detail and how it is integrated into the overall design of the system.

Start

1.User Chooses:

1 - login

2 - register

This is the first decision point where the user will be led either to login with the credentials of an existing account or to a registration to create a new account.

Registration Process

2.Register User:

-Collect user details such as:

-Username (unique)

-Age

-Gender

-Password

-Re-enter password

-Preferences (favorite directors, actors, and movie genres).

-Store these data into a file for the user's future use.

Post-Login

3.User Selects:

1 - view recommendations

2 - view profile

3 - logout

Once the user is logged in, they will be asked to view the movie recommendations available and view or edit their profile, else log out.

Viewing Recommendations

4. **View Recommendations:**

-The system looks up the user's preference from his file.

-Prints 5 recommendations based on these preferences.

-Users have the option to:

1 - display movie info

2 - update recommendations

3 - menu

Viewing Movie Information

5. **Display Movie Info:**

When the users choose to display movie information, the system returns information to present:

-Movie title

-Rating

-Genre

-Director

-Main actors.

> o   The users then receive options:

1 - add to watchlist

2 - rate movie

3 - menu

Adding to Watchlist

6.Add to Watchlist:

> o   The movie is included in the watchlist file of a user, which is stored in their file.

Rating a Movie

7. Rate Movie:
> o   The users rate the movie on a scale of 5.
> o   The system records this rating in the movie file and updates the overall rating by calculating the average rating from all the users.

Changing Recommendations

8. Update Recommendations:
> o   The user can change their preferences, that is, their favorite directors, actors, and movie genres.
> o   The updated preferences are all stored in a user file, and the new recommendation will be based on these updated preferences.

Viewing and Updating Profile

9. Display Profile:

- System prints user's profile details as follows:
  - Username
  - Gender
  - Watchlist
- Users can:
  - Edit their preferences
  - Go back to the main menu

Logging Out

10. **Log out:**
    - Logs a user out of the session, taking a user back to the beginning.

Flowchart Visualization

Here's a more detailed breakdown of the flowchart elements in relation to the provided description:

1. **Start and User Choice:**
   - Initial decision point for login or registration.

2. **Registration Path:**
   - Collects detailed user information and saves it for future recommendations.

3. **Post-Login User Choices:**
   - Users can choose to view recommendations, profile, or logout.

4. **Recommendation Process:**
   - Provides movie recommendations to the user according to preference inputs, and the recommendations are displayed.

5. **Detailed Movie Info:**
   - In-depth information about the selected movies.

6. **Watchlist and Ratings:**
   - Allows adding movies to watchlists or rating the movie so that proper next-time recommendations can be made.

7. **Profile Management:**

o   The users can update their profile and preferences.

OOP concepts used on the code

1. **Encapsulation**

   Encapsulation is the concept of bundling data and methods that operate on that data within a single unit, often a class, and restricting access to some of the object's components. This is achieved through access specifiers like private, protected, and public.

   Classes: Person, User, Movie, Genre, Director, Actor, and Recommendation System encapsulate data and functions related to their specific purposes.

   Private Data Members: Attributes like email, password, ratings in the User class, and similar attributes in other classes, are encapsulated and only accessible through public methods.

2. **Inheritance**

   Inheritance is a mechanism where a new class inherits properties and behavior from an existing class.

   Base Class: Person

   Derived Classes: User, Director, and Actor inherit from Person. This means User, Director, and Actor classes inherit attributes name and gender and can use methods like getName() and getGender() from the Person class.

3. **Polymorphism**

   Polymorphism allows objects to be treated as instances of their parent class rather than their actual class. This concept is not explicitly shown in the code, but the structure allows for polymorphic behavior.

   Function Overriding: In the derived classes, methods can be overridden to provide specific implementations. Although not directly demonstrated, the structure supports this.

4. Composition

Composition is a design principle where a class is composed of one or more objects from other classes, indicating a "has-a" relationship.

User Class: Contains vectors of genres, actors, and directors, indicating that a user has preferences for multiple genres, actors, and directors.

Movie Class: Contains vectors of genres and cast, indicating that a movie has multiple genres and actors.

Recommendation System Class: Contains vectors of User*, Movie*, Genre*, Director*, and Actor*, indicating it manages collections of these objects.

Additional Concepts

1. Utility Functions

Validation Functions: is Valid Email and is Valid Password are utility functions that validate email and password formats using regular expressions and length checks, respectively.

2. User Interaction

Input/Output Operations: The main function demonstrates basic user interactions through console input and output operations, allowing users to register, log in, view profiles, get recommendations, and rate movies.

3. Summary

The code effectively uses OOP principles to create a robust recommendation system that manages users, movies, genres, directors, and actors. Encapsulation is used to protect and manage data, inheritance to reuse and extend functionality, polymorphism to allow for flexibility and future extensions, and composition to build complex types from simpler ones.

UML

User

- email: string
- password: string
- genres: vector<string>
- actors: vector<string>
- directors: vector<string>
- ratings: vector<pair<int, int>>

+ getDirectors(): vector<string>
+ getActors(): vector<string>
+ getGenres(): vector<string>
+ getRating(movieID: int): int
+ rateMovie(movieID: int, rating: int)
+ User(name, gender, email, password)
+ getEmail(): string
+ getPassword(): string
+ setGenres(genres: vector<string>)
+ setActors(actors: vector<string>)
+ setDirectors(directors: vector<string>)

Genre

- genreID: int
- type: string
- movies: vector<Movie*>

+ getMovies(): vector<Movie*>
+ addMovie(movie: Movie*)
+ Genre(genreID, type)
+ getGenreID(): int
+ getType(): string

Recommendation System

- users: vector<User*>
- movies: vector<Movie*>
- genres: vector<Genre*>
- directors: vector<Director*>
- actors: vector<Actor*>

+ getActors(): vector<Actor*>
+ getDirectors(): vector<Director*>
+ getGenres(): vector<Genre*>
+ getUsers(): vector<User*>
+ getMovies(): vector<Movie*>
+ saveActors(filename: string)
+ addUser(user: User*)
+ addMovie(movie: Movie*)
+ generateRecommendations(user: const User&): vector<Movie*>
+ updateAlgorithm(algorithm: string)
+ loadUsers(filename: string)
+ saveUsers(filename: string)
+ loadMovies(filename: string)
+ saveMovies(filename: string)
+ loadDirectors(filename: string)
+ saveDirectors(filename: string)
+ loadActors(filename: string)

Person

- name: string
- gender: string

+ Person(name, gender)
+ getName(): string
+ getGender(): string

Movie

- movieID: int
- title: string
- genres: vector<string>
- director: string
- cast: vector<string>
- releaseDate: string
- ratings: vector<pair<int, int>>

+ Movie
  (movieID, title, genres,
  director, cast, releaseDate)
+ getMovieID(): int
+ getTitle(): string
+ getGenres(): vector<string>
+ getDirector(): string
+ getCast(): vector<string>
+ getReleaseDate(): string
+ addRating(userID: int, rating: int)
+ getAverageRating(): double

Director

- directorID: int
- movies: vector<Movie*>

+ Director(name, gender, directorID)
+ getDirectorID(): int
+ addMovie(movie: Movie*)
+ getMovies(): vector<Movie*>

Actor

- actorID: int
- movies: vector<Movie*>

+ Actor(name, gender, actorID)
+ getActorID(): int
+ addMovie(movie: Movie*)
+ getMovies(): vector<Movie*>