THIS SECTION HAS DONE BY MOQBEL AND FUAD

Section B: Problem Analysis

Objects and Classes

CLASS : Person

Attributes:

- name
- gender

CLASS : USER

Attributes:

- userID
- email
- password
- preferences (genres, actors, directors)
- Movies (Movie class object)

Methods:

- register()
- login()
- updatePreferences()
- viewRecommendations()
- rateMovie()

CLASS : Movie

Attributes:

- movieID
- title
- genres (Genre class object)
- director
- cast
- releaseDate
- ratings (Rating class object)

Methods:

- getDetails()
- calcRating()
- addRating()

CLASS : RecommendationSystem

Attributes:

- recommendationAlgorithm (collaborative filtering)

Methods:

- generateRecommendations(userID)
- updateAlgorithm(algorithmType)

CLASS : Rating

Attributes:

- ratingID
- userID
- movieID
- score
- review

Methods:

- addRating()
- updateRating()
- getAverageRating(movieID)

CLASS : Genre

Attributes:

- genreID
- type

Methods:

- getMoviesByGenre()
- addMovieToGenre()

CLASS : Director

Attributes:

- directorID

Methods:

- getMoviesByDirector()

CLASS : Actor

Attributes:

- actorID

Methods:

- getMoviesByActor()

1- Identify Class Relationships

Association Relationships:

User and Movie: Users can rate multiple movies and view recommendations.

Justification: Each user interacts with the movie database through ratings and preferences, which are essential for generating personalized recommendations.

Movie and Rating: Movies have multiple ratings associated with them.

Justification: Ratings given by users are stored and associated with movies to calculate average ratings and improve recommendation accuracy.

Recommendation System and User: The recommendation system generates personalized movie suggestions for users.

Justification: The core functionality of the system is to provide personalized recommendations based on user data and preferences.

Movie and Genre: Movies belong to one or more genres.

Justification: Genre classification helps in content-based filtering and enhances the recommendation process.

Inheritance Relationships:

Person (Base Class) -> User, Director, Actor (Derived Classes)

Justification: Users, directors, and actors share common attributes like name and unique IDs, which can be generalized into a base class to avoid redundancy and promote code reuse.

Justification of Relationships

Association Relationships:

Users interact with movies primarily through ratings, which are crucial for generating recommendations. Thus, the User-Movie association is necessary for capturing user feedback and preferences.

The Movie-Rating association is fundamental for maintaining a record of all ratings given to a movie, enabling the system to calculate average ratings and use them in the recommendation algorithm.

The Recommendation System-User association is justified as the recommendation system needs user data to generate personalized suggestions.

Associating movies with genres helps in filtering and categorizing movies, improving the accuracy of content-based recommendations.

Inheritance Relationships:

Generalizing common attributes and methods into a base class (Person) for users, directors, and actors simplifies the system design and enhances maintainability by reducing code duplication.