FACULTY OF COMPUTING
UTM Johor Bahru

SECJ1023-04 – PROGRAMING TECHNIQUE II

Semestre 2 – 2023/2024

Lecturer: Dr. Lizawati Mi Yusuf

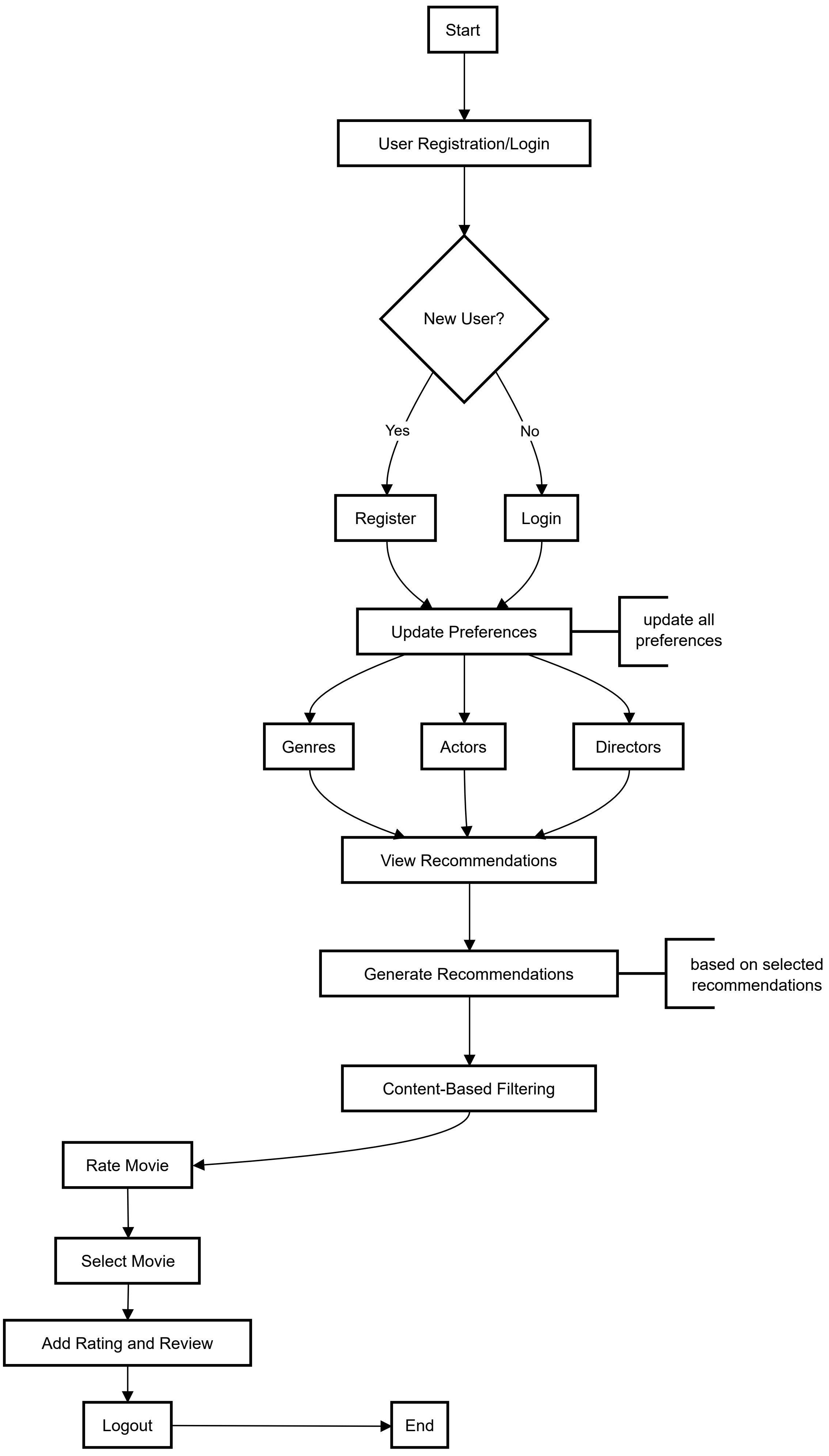
Group Name: No Name Group

Name	Matric No.
Mohammed Moqbel	A23CS4016
Ali Rida	A23CS4003
Kinan Fadi	A23CS4013
Fouad Mahmoud	A23CS0017

Section: 04

Section A: Flowchart

The flowchart maps out an intuitive process for a movie recommendation system, guiding users from sign-up to enjoying personalized movie suggestions. It all starts with the user either registering as a new member or logging in with existing credentials. New users fill in their information to create an account, while returning users simply log in. Once inside, users can update their preferences by choosing favorite genres, actors, and directors, which the system uses to tailor movie recommendations. These recommendations are generated using content-based filtering, ensuring they align with the user's tastes. Users can then browse these recommendations, select a movie, and get more details. After watching, they can rate the movie and add a review, contributing to the accuracy of future suggestions. The process ends with users logging out. This user-centric flow keeps the experience personalized and engaging, making sure that the movie recommendations always reflect the user's evolving preferences.


Start
User Registration/Login
New User?
Yes
No
Register
Login
Update Preferences
update all preferences
Genres
Actors
Directors
View Recommendations
Generate Recommendations
based on selected recommendations
Content-Based Filtering
Rate Movie
Select Movie
Add Rating and Review
Logout
End

Section B: Problem Analysis

Objects and Classes

CLASS : Person

Attributes:

- name
- gender

CLASS : USER

Attributes:

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

Methods:

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

CLASS : Movie

Attributes:

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

Methods:

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

CLASS : RecommendationSystem

Attributes:

- recommendationAlgorithm (collaborative filtering)
- user (User class object)

Methods:

- generateRecommendations(userID)
- updateAlgorithm()

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.

Movie and Actor: Movies have at least one genre

Justification: Actors of a movie will help to develop the recommendation algorithm.

Movie and Director: Movies have at least one genre Director

Justification: Directors of a movie will help to develop the recommendation algorithm.

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.

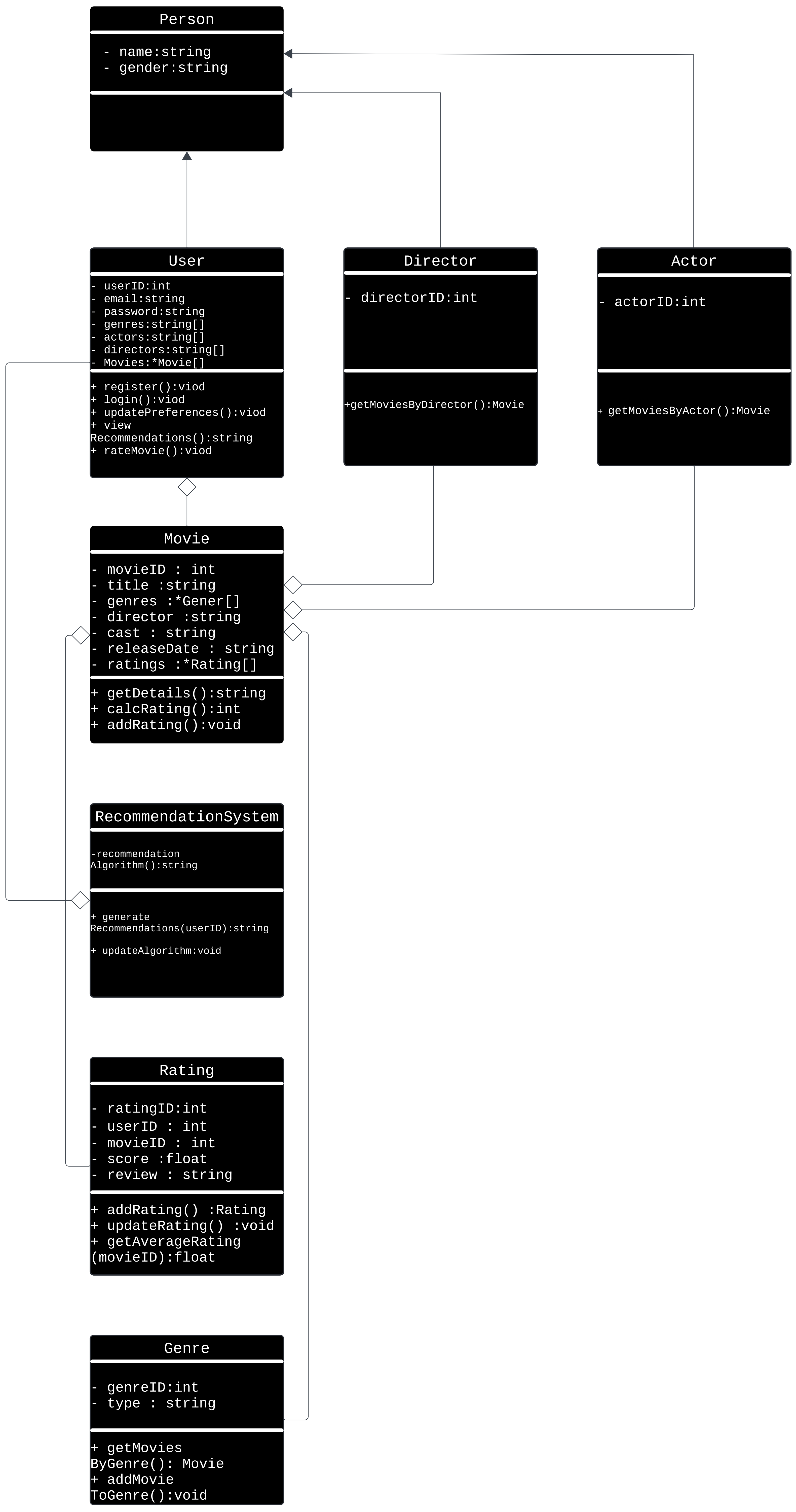
The Movie-Rating association is fundamental for maintaining a record of all ratings given to a movie, enabling the system to calculate average ratings and use them in the recommendation algorithm.

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.


Person
- name:string
- gender:string
User
- userID:int
- email:string
- password:string
- genres:string[]
- actors:string[]
- directors:string[]
- Movies:*Movie[]
+ register():viod
+ login():viod
+ updatePreferences():viod
+ view Recommendations():string
+ rateMovie():viod
Director
- directorID:int
+getMoviesByDirector():Movie
Actor
- actorID:int
+ getMoviesByActor():Movie
Movie
- movieID : int
- title :string
- genres :*Gener[]
- director :string
- cast : string
- releaseDate : string
- ratings :*Rating[]
+ getDetails():string
+ calcRating():int
+ addRating():void
RecommendationSystem
-recommendation Algorithm():string
+ generate Recommendations(userID):string
+ updateAlgorithm:void
Rating
- ratingID:int
- userID : int
- movieID : int
- score :float
- review : string
+ addRating() :Rating
+ updateRating() :void
+ getAverageRating (movieID):float
Genre
- genreID:int
- type : string
+ getMovies ByGenre(): Movie
+ addMovie ToGenre():void