Our project is a movie genre recommendation system built with python. The goal of the project is to help users discover movies they might enjoy based on their favorite genre.

We used a CSV file as our movie database, which includes details like the movie's genre, title, and a short synopsis. The program loads this data, displays all available genres, and asks the user to choose one. After that, it recommends movies that match the selected genre, along with a brief description of each.

Code implementation:

Moviehandler.py:

The "MovieHandler" script serves as the core module for managing and querying movie data within the recommendation system. It encapsulates all functionality needed to load, filter, and provide access to movie information.

Upon initialization, the class automatically calls the __load_movies method to read movie data from a CSV file (movieData.csv). The data is stored as a list of dictionaries, where each dictionary contains the genre, title, and synopsis of a movie.

The class provides several methods to interact with the data:

- get_all_genres() returns a set of all available movie genres.
- get_movies() returns the complete list of movies.
- get_movie(movie_title) searches for a specific movie by its title and returns its details.
- get_movies_by_genre(genre_name) filters and returns all movies that belong to the specified genre.

This module operates independently of the user interface and ensures a clean separation between data handling and user interaction. It is responsible for all data-related operations in the recommendation system and forms the backbone of the application.

UserInteractionHandler.py:

The "UserInteractionHandler" script is the user interface for the movie recommendation system. Its main role is to guide the user through selecting a preferred genre and displaying a list of corresponding movies retrieved via the "MovieHandler" class.

The script begins by instantiating the "Moviehandler" object, which loads movie data from the CSV file. It then retrieves and displays a sorted list of available genres to the

user. After prompting the user for a genre, the program performs input validation to make sure the genre exists in the CSV file. If the user input is valid, the movies and synopsis available in that genre will be listed, if not valid the user will receive an error message.

Result:

After the completion of the code the program successfully lists all available movies, genres and synopsis. The program prompts the user to input a genre of the available selection. If the genre is not in the csv list the program will give the user an error message. The program is able to read the CSV file and match genres regardless of casing improving usability.

Lessons learned:

1. The importance of case handling: initially, genre input was case-sensitive, which caused valid user inputs like “drama” instead of “Drama” to fail. Fixing this made the program user-friendly.
2. Error checking and input validation: Adding feedback for incorrect inputs enhanced the overall reliability of the program.
3. Working with CSV files: Understanding row parsing was essential.