**Report: Movie Recommendation System**

1. **Introduction:**

The Movie Recommendation System (MRS) is designed to manage movie data and provide users with functionalities such as adding, updating, and displaying movie information, as well as sorting and filtering options. This report outlines the implementation details and functionalities of the system based on the provided code.

1. **Problem Solving Approach**
   * **Requirement Analysis**
     + **Understanding the Problem:** The system aims to facilitate movie management and recommendation based on user interactions and stored data.
     + **Defining User Interaction:** User interactions are defined through a menu-based interface, allowing users to perform various actions such as adding, updating, and viewing movie information.
   * **Modular Design**
     + **Identifying Components:** The system is divided into modular functions, each responsible for a specific task, such as loading movies, adding new movies, and displaying sorted movies.
     + **Functionality Mapping:** Functions are designed to address specific requirements, enhancing code organization and maintainability.
   * **Implementation Strategy**
     + **File Handling:** Functions for loading and saving movie data from/to a text file ensure data persistence across sessions.
     + **User Input Validation:** Input validation mechanisms are incorporated to ensure the correctness of user-provided data.
     + **Error Handling:** Error handling mechanisms handle potential issues such as file not found errors or invalid user input.
   * **Testing and Iteration**
     + **Testing Functionality:** Each function is tested individually to ensure its correctness and robustness under different scenarios.
     + **User Testing:** User interactions are simulated to identify usability issues or bugs, leading to iterative refinements.
   * **Refinement:** 
     + Based on testing feedback, the system is refined iteratively to improve functionality and user experience.
2. **Implementation Details (Functions)**

* **`load\_movies()`**
  + **Task:** This function loads movie data from a text file into memory as a list of dictionaries.
  + **Implementation:** The function reads movie data from the file, parses it into dictionary format, and returns a list of movies.
* **`save\_movies()`**
  + **Task:** This function saves movie data from memory back to the text file.
  + **Implementation:** It writes movie data in CSV format to the file, including a header line and data for each movie.
* `**add\_movie()`**
  + **Task:** Allows users to add new movie information.
  + **Implementation:** Users are prompted to input details such as movie name, writer, rating, and revenue, with validation and duplicate handling.
* **`update\_movie()`**
  + **Task:** Allows users to update existing movie information.
  + **Implementation:** Users provide the movie name and year to identify the movie to update, and then update various fields as needed.
* **`display\_sorted\_movies()`**
  + **Task:** Displays movies sorted based on specified criteria.
  + **Implementation:** Movies are loaded and sorted based on user preferences or predefined attributes, then displayed.
* **`save\_genre\_info()`**
  + **Task:** Saves movie information for a certain genre into a new file.
  + **Implementation:** Users specify a genre, and movies belonging to that genre are filtered and saved to a new file.
* **`display\_specific\_info()`**
  + **Task:** Displays specific information about movies based on user choice.
  + **Implementation:** Users can choose to view top-selling movies, total revenue per year, average revenue per year, or total number of movies per author.
* **`view\_top\_selling\_movies()`**
  + **Task:** Displays the top 5 selling movies in the last 10 years.
  + **Implementation:** Movies are filtered based on the last 10 years and sorted by user rating and year before displaying the top 5.
* **`print\_total\_revenue\_per\_year()`**
  + **Task:** Prints the total revenue for all movies per year.
  + **Implementation:** Calculates total revenue for each year by summing up revenues of all movies in that year, then prints the result.
* **`print\_average\_revenue\_per\_year()`**
  + **Task:** Prints the average revenue of all movies per year.
  + **Implementation:** Calculates average revenue for each year by dividing total revenue by the number of movies in that year, then prints the result.
* **`print\_total\_movies\_per\_author()`**
  + **Task:** Prints the total number of movies per author.
  + **Implementation:** Counts the number of movies for each author and prints the result.
* **`display\_all\_movies()`**
  + **Task:** Displays information for all movies.
  + **Implementation:** Loads all movies from the file and prints their details for user viewing.
* **`main()`**
  + **Task:** Main entry point of the program.
  + **Implementation:** Displays a menu of options for user interaction and calls appropriate functions based on user input to perform tasks.

1. **Screenshots**

* **Function: `load\_movies()`**



* **Function: `save\_movies()`**

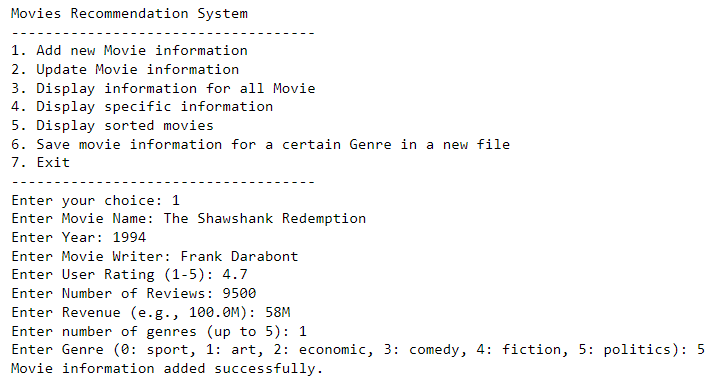
**A screen shot of a computer program

Description automatically generated**

* **Function: `add\_movie()`**

**A screen shot of a computer program

Description automatically generated**

****

* **Function: `update\_movie()`**

**A computer screen shot of a program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

* **Function: `display\_sorted\_movies()`**

****

**A computer screen shot of text

Description automatically generated**

* **Function: `save\_genre\_info()`**

**A computer screen shot of a program code

Description automatically generated**

* **Function: `display\_specific\_info()`**

**A screenshot of a computer program

Description automatically generated**

* **Function: `view\_top\_selling\_movies()`**

**A computer code with many colorful text

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated**

* **Function: `print\_total\_revenue\_per\_year()`**

**A computer code with text

Description automatically generated with medium confidence**

**A screen shot of a computer program

Description automatically generated**

* **Function: `print\_average\_revenue\_per\_year()`**

**A computer screen shot of a program

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

* **Function: `print\_total\_movies\_per\_author()`**

**A screen shot of a computer code

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

* **Function: `display\_all\_movies()`**

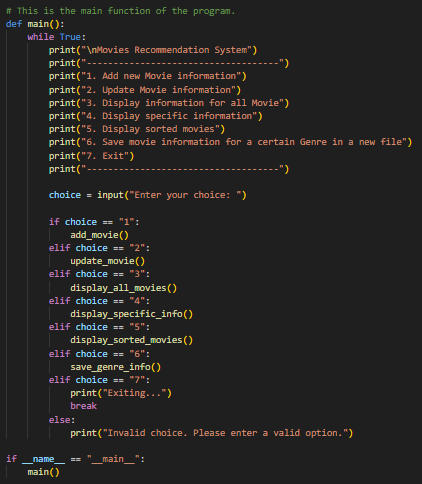
**A white screen with text

Description automatically generated**

**A computer screen shot of text

Description automatically generated**

* **Function: `main()`**

****

1. **Conclusion**

The Movie Recommendation System was successfully implemented using a systematic problem-solving approach. Through modular design, careful implementation, and iterative refinement, the system provides users with a robust and user-friendly platform for managing and exploring movie data. Further enhancements and features could be added to improve the system's functionality and usability.