Movie recommendation program documentation

Chapter 1: Introduction and Problem Statement:

Overview:

The movie recommendation system is a C++ program designed to provide personalized movie recommendations to users based on their preferences. The purpose of this project is to create a user-friendly and efficient system that assists in discovering films they are likely to enjoy. The recommendation program uses object-oriented programming principles to handle movie data and offer suggestions to the users.

Technologies Used:

- C++ programming language
- Microsoft Excel for storing movie data
- Object-Oriented Programming (OOP) concepts

Problem Statement:

The movie recommendation system aims to address the challenge of movie selection and provide recommendations to users. With this much of movies available, it can be overwhelming for people to choose a movie that matches their interests. The goal is to develop a program that asks for the user preferences and suggests movies that align with their tastes, to enhance their movie-watching experience.
By creating this movie recommendation system, users will benefit from:

- Time saving: Users can quickly discover movies they are likely to enjoy without manually searching through a large movie catalog.
- Exploration of diverse movies: Users can explore movies they may not have discovered otherwise, expanding their cinematic experience.

Done By : Basel Al-Raoush Neptune code:SDR1AO

Chapter 2: Solution Design and Implementation

Chosen Solution:

The chosen solution for the movie recommendation system is a C++ program that utilizes object-oriented programming (OOP) concepts. The system's design incorporates various functions to handle movie data, generate recommendations, and interact with users. The program reads movie data from an Excel file, provides options to view, explore movie details, and generates recommendations based on user preferences.

Implementation Details:

Class Structure:

- Movie Class: Represents a movie and stores its attributes such as title, genre, year, and rating. It includes member functions to access and modify movie details.

Functionality: Algorithms and Techniques

1. ***std::vector<Movie*> ReadMoviesFromFile(const std::string& filename);***
    - This function takes a string parameter **filename**, opens the Excel file, reads the data, and stores each row as a pointer Movie object.
    - It creates a vector to store all the Movie objects and returns the vector.
2. ***void printMovieList(const std::vector<Movie*>& movies);***
    - This function receives a vector of pointers to Movie objects and iterates through the vector, printing the details of each movie.
    - It displays the title, genre, year, rating, and any other details of the movie.
3. ***void printMovieDetails(const Movie* movie);***
    - This function receives a pointer to a Movie object and accesses its attributes to print detailed information about the movie.
    - It displays the title, genre, year, rating, and all the details of the movie.
4. ***Movie* generateRandomMovie(const std::vector<Movie*>& movies);***
    - This function takes a vector of Movie objects as a parameter and iterates through the vector.
    - It creates a new vector called **Vec** and adds movies with a rating above 8 to it.
    - Using random number generation, it generates a random index within the size of **Vec** and returns a pointer to a random movie from the new vector (which contains movies rated higher than 8).

5. ***std::vector<Movie*> GetTopRatedMoviesByYear(const std::vector<Movie*>& movies);***
    - This function takes a vector of Movie objects as a parameter.
    - It prompts the user to enter a year and searches the vector for movies with the same year.
    - It adds matching movies to a new vector called **top10**.
    - Using the **sort** function from the algorithm library, it sorts the movies in **top10** by rating in descending order.
    - Finally, it reduces the size of **top10** to 10 by using the **pop_back()** function from the vector library, returning only the top 10 movies with the highest ratings.
6. ***Movie* GenerateRandomMovieByGenreAndYear(const std::vector<Movie*>& movies);***
    - This function takes a vector of Movie objects as a parameter.
    - It displays the available genres in the file.
    - It then asks the user to enter a year and creates a vector called **matchingMovies** to store movies that match the chosen genre and year.
    - Using the same logic as generating a random movie from a vector, it generates a random number within the size of **matchingMovies** and returns a pointer to a random movie from that vector.

7. ***std::vector<Movie*> searchMovies(const std::vector<Movie*>& movieList);***
    - This function takes a vector of Movie objects as a parameter.
    - It displays a menu to the user, asking what they want to search for: title, year, or writer name.
    - Based on the user's choice, it creates a vector to store the matching results.
    - It then returns the vector of movies that match the search criteria.
8. ***int viewMenu();***
    - This function displays the menu of options available to the user.
    - It prompts the user to enter a choice and returns the selected option as an integer.

9. ***bool handleMenu(int choice, std::vector<Movie*>& movies);***
    - This function takes the user's choice (as an integer) and the vector of Movie objects as parameters.
    - It handles the user's choice using a switch-case statement.
    - Depending on the selected option, it performs the corresponding action.
    - The function returns a Boolean value indicating whether the program should continue running or exit.

Chapter 3: External Solutions and Deviations

External solutions:

The implementation of the project relied solely on the standard libraries provided by the C++ programming language. No external solutions were used.

Deviations from the Original Project Plan:

During the development of the movie recommendation system, there were a few deviations from the original project plan. These deviations include the use of vectors instead of linked lists to store the file data and adding more functionalities.

Use of Vectors instead of Linked Lists

In the original project plan, linked lists were proposed as the data structure to store the movie data. However, during the implementation , I decided to use vectors instead. The decision to use vectors was because of the convenience and flexibility they offer.

Additional Functionality

The initial plan focused on generating random movie recommendations based on user preferences. However, I realized that users may also benefit from the ability to search for specific movies based on various criteria, such as title, year, or writer name. To get this, I created the search movies functions to allow users to search the movie list and get movies that match their search criteria. This additional functionality enhances the system's functionality and helps users to explore movies beyond random recommendations.

Chapter 4: Testing and Verification

Testing

testing was performed to validate the functions of the movie recommendation program. Each function and method were tested independently, using a variety of test cases to cover different scenarios. This approach helped identify any logical errors, boundary cases, or unexpected behaviors within the code.

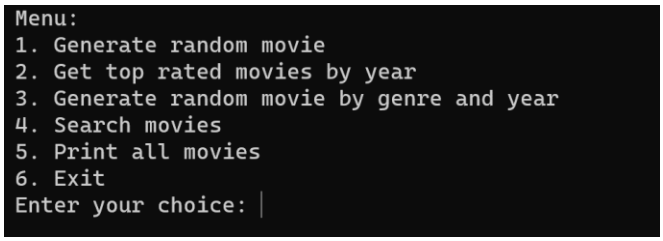
Verification

The verification process involved confirming that the movie recommendation program met the specified requirements. It ensured that the system's functionalities were correctly implemented, the user interface was friendly, and the recommendations provided relevant and accurate movie suggestions.

user interface:

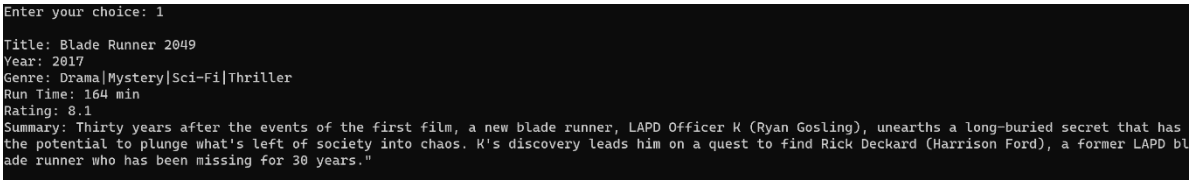
Upon launching the application, a user-friendly menu will be presented on the screen, providing a variety of options for the user to choose from. Each option is assigned a unique number for easy selection. To interact with the system, the user simply needs to enter the corresponding number associated with their desired option.

The menu serves as a central hub for accessing the various features and functionalities of the application. It offers a straightforward and intuitive way for users to navigate through the system and perform specific actions based on their preferences.



Generating random movie:

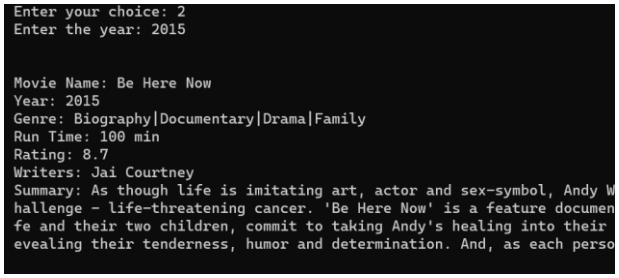
By selecting option (1) from the menu, users can generate a random movie recommendation. The recommended movie will be displayed on the screen. Afterward, the menu will reappear, allowing users to choose another option or exit the program. This feature provides an element of surprise and discovery, expanding users' movie-watching experiences.



Top rated movies by year:

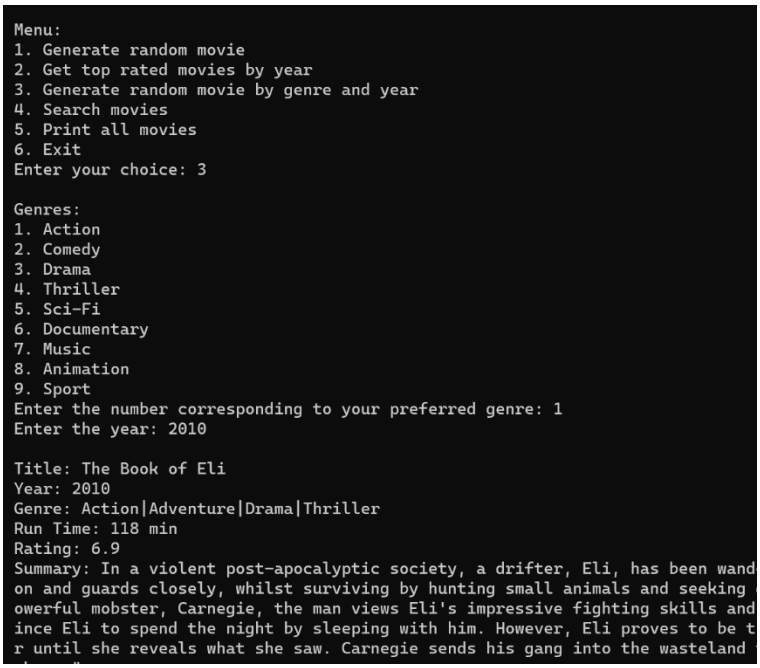
To access the feature of retrieving the top-rated movies by year, users should enter option (2) from the main menu. Once selected, the system will prompt the user to enter a specific year.

Upon receiving the user's input, the application will search for movies released in the provided year and compile a list of the highest-rated movies. The list will be sorted based on the movies' ratings, showcasing the top-rated movies first. After presenting the list, Users can then proceed to choose another option or exit the program from the menu.



Generating a random movie by genre and year:

Selecting option (3) from the menu will present users with a list of available genres. Users can enter the corresponding number for their preferred genre. The system will prompt them to enter a specific year.



Based on the selected genre and year, the application will generate a random movie recommendation that matches the specified criteria. The recommended movie will be displayed on the screen.
After presenting the random movie, the main menu will reappear, allowing users to choose another option or exit the program.

Searching for movies:

To utilize the search feature, users can select option (4) from the menu. This will display a menu of search options, including searching by title, year, or writer.

For instance, if the user wants to search for a movie by its title, they can input the movie's name in the search prompt. As an example, let's consider the search term "batman."
The application will search through the available movies and display all the movies that match the search criteria (in this case, movies with "Batman" in the title). The matching movie results will be presented on the screen.
Following the display of the search results, the main menu will reappear, offering users the choice to select another option or exit the program.
This search functionality enables users to find movies based on specific criteria, making it easier to discover movies they are interested in watching.

```
Menu:
1. Generate random movie
2. Get top rated movies by year
3. Generate random movie by genre and year
4. Search movies
5. Print all movies
6. Exit
Enter your choice: 4
What would you like to search?
1. Movie title
2. Year
3. Writer name
1
Enter the movie title: batman


Movie Name: Batman: Gotham by Gaslight
Year: 2018
Genre: Action|Adventure|Animation|Crime|Fantasy
Run Time: 78 min
Rating: 6.8
Writers: James Krieg
Summary: In an age of mystery and superstition, how would the people of Gotham react
 Elseworlds tale re-imagines the Dark Knight detective in Victorian times and pits h


Movie Name: Batman vs. Two-Face
Year: 2017
Genre: Action|Animation|Comedy
Run Time: 72 min
Rating: 6.1
Writers: Michael Jelenic
```

After each user choice and corresponding action, the main menu will be displayed again, allowing users to decide whether to continue using the program or exit.
To exit the program, users can enter option (6) from the menu. Selecting this option will terminate the program.

```
Menu:
1. Generate random movie
2. Get top rated movies by year
3. Generate random movie by genre and year
4. Search movies
5. Print all movies
6. Exit
Enter your choice: 6

C:\dev\Projects\MovieRecommendation\x64\Debug\MovieRecommendation.exe (process 30144) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Error Handling:

The movie recommendation system is designed to handle expected errors and ensure smooth user interaction. To achieve this, the program employs a menu-based approach, where all user choices are required to be numeric and correspond to specific functionalities.

By enforcing a numeric input format, the program reduces the likelihood of errors caused by incorrect user entries. Additionally, exceptional handling techniques are utilized to prevent any unintended program behavior resulting from invalid user input.

Through the careful implementation of error handling mechanisms, the program strives to provide a robust and user-friendly experience, minimizing the occurrence of errors and ensuring the program operates as intended.

```
Menu:
1. Generate random movie
2. Get top rated movies by year
3. Generate random movie by genre and year
4. Search movies
5. Print all movies
6. Exit
Enter your choice: x
Invalid! please enter a number within the options available!
```

```
Menu:
1. Generate random movie
2. Get top rated movies by year
3. Generate random movie by genre and year
4. Search movies
5. Print all movies
6. Exit
Enter your choice: 2
Enter the year: 3000
please choose a year between 2000 ~ 2018!
Enter the year: x
invalid! please enter a numeric year!
Enter the year:
```

Chapter 5: Results, Discussion, and Future Enhancements

The movie recommendation system project has successfully addressed the problem of providing movie recommendations to users based on their preferences.
During the development process of the movie recommendation system, several challenges were encountered. These challenges included:

- **Data Parsing and File Handling:** reading movie data from an Excel file and handling file input/output operations required careful consideration and implementation. appropriate techniques were used to read and store the data correctly.
- **Algorithm Design and Implementation:** Designing and implementing algorithms for random movie generation, sorting, and searching required analysis and consideration of efficiency. Iterative testing and refinement were performed to ensure accurate results.
- **User Experience :** Creating a user-friendly interface, menu options, and information presentation. Feedback from students was essential to enhance the system's usability and user satisfaction.

Feedbacks

Feedback received from users was improving the system. Users appreciated the accuracy of movie recommendations. The user interface was generally well-received for its simplicity and clarity. Valuable suggestions for improvement were provided, leading to enhancements in the user interface.

Future Enhancements

To further enhance the movie recommendation system, several future improvements and enhancements can be considered:

1) **Mobile Application Development:** Developing a mobile application

2) **Integration with External APIs:** Integrating the system with external APIs, such as movie databases or streaming platforms, would provide more collections of movie data. This would enable access to real-time movie information, including ratings, reviews, and up-to-date releases.

3) **expand program capabilities:** To expand the program's capabilities to a broader audience, a future enhancement plan is to introduce a new abstract class called "Media" with two derived classes: "Movies" and "TV Shows". This addition will enable the recommendation system to provide recommendations for both movies and TV shows, offering users a wider range of entertainment options.

By incorporating TV shows into the recommendation algorithm, users will have the opportunity to discover TV series based on their preferences. This enhancement will require modifying the existing codebase, implementing new functionalities, and updating the user interface to accommodate TV show recommendations.

Chapter 6: Conclusion and References

The movie recommendation system project has been a fulfilling journey, successfully addressing the problem of providing movie recommendations.
Challenges were encountered and overcome, including data parsing and file handling, algorithm design and implementation, and user experience design. Feedback from users and testers played a crucial role in refining the system and improving its usability.

During the project, several valuable lessons were learned:

- Flexibility in Implementation: Being open to exploring different data structures and approaches, such as using vectors instead of linked lists, can lead to more efficient and convenient implementations.

References

The following references were consulted during the project:

- **Stack Overflow (https://stackoverflow.com):** Stack Overflow was used as a resource to find solutions to specific coding problems and troubleshoot issues encountered during the development process.
- **ChatGPT:** The ChatGPT language model was utilized as a learning resource to gain insights into various algorithms and programming concepts.
- **Coding Communities:** Participation in coding communities allowed for discussions and knowledge sharing, providing valuable insights and guidance.