CPT111 CW3

December 2025

Contents

1 Introduction

On modern digital platforms, users rely more on movie recommendation systems to receive more personalized services. With the abundance of movie resources available online, users find it difficult to make a choice. For instance, if a user is interested in horror movies but lacks a recommendation system, they would have to spend a lot of time searching for films that meet their needs. Therefore, the role of our recommendation system becomes even more significant. Additionally, platforms like NETFLEX have also enhanced their service capabilities through this recommendation technology, demonstrating the feasibility of this technology.

The purpose of this project is to complete a movie recommendation and tracking system using Java. This system enables users to log in, browse all the provided movies, manage personal viewing lists, record their viewing history, and receive movie recommendations based on their preferences. The CSV file will store all the information about movies and users, and the system will read and update this content during operation.

The technologies used in this project include object-oriented programming, where encapsulation, classes and objects, and modular design are mainly employed. Moreover, file input/output and exception handling have been integrated into this project. Overall, this project extensively utilizes the knowledge taught in the module.

2 System Design and Class Structure

2.1 Class Structure Description

This system is mainly built around several model classes, one data processing class, one recommendation component, and a set of JavaFX controller classes for managing interactions. These classes together form the core structure of the system and illustrate the way data operates within the system.

The movie class is the basic data model table for movies. Each movie object stores basic attributes such as title, type, rating, and description. This class serves as the basic information unit used in browsing history, viewing list management, tracking, and recommendation functions. Since the movie class is loaded from CSV and is frequently passed, this class is one of the key models of the system.

The user class represents a single user of the system. This class stores the collection list and viewing history record, as well as the username and password. These lists are composed of movie objects, reflecting the user's activities within the application. The user class provides access to these lists, which is important for saving user progress and generating personalized recommendations.

The FileHandler class is responsible for handling the data persistence in the system. This class is responsible for reading movie and user information when the application starts. When changes occur, it also updates the user's collection

list and history record. By concentrating file input and output operations in one class, the system can clearly separate data storage and application logic, which also improves the maintainability of the system.

The RecommendationEngine class implements the recommendation logic of the system. This component can be used to analyze movie genres and determine which genre has the highest frequency of occurrence. Based on this simple frequency-based method, it generates a list of recommended movies that match the user's preference type. Although this algorithm is relatively simple in design, this class encapsulates the recommendation logic and makes it independent of the user interface.

Multiple JavaFX controller classes are used to manage the interaction part of the system. The LoginController is responsible for user authentication during login and is then transitioned to the main interface after verification. The BrowseMoviesController can display all movie resources and allow users to add movies to the collection list or watch history. The HistoryController shows the movies that the user has watched before, while the WatchlistController is responsible for managing the movies that the user intends to watch. The RecommendationController displays the recommended movies generated by the RecommendationEngine. The navigation between these parts is coordinated by the MainController (the main controller), which acts as a central hub.

In addition to the basic user login function, we also support user registration and password modification. The registration function is handled by a dedicated registration service class, which can add new users to the system. The password modification function enables users who meet the conditions to update their passwords, and access control is based on their levels.Besides, the system also enables users with higher levels to have the option to customize the interface, such as choosing themes.

2.2 Key Libraries and Technologies Used

This system utilizes multiple standard Java and JavaFX libraries to accomplish tasks such as data processing, file operations, security, and user interface development. The system relies on classes in the java.util package to complete aspects of data structures and basic collections, such as List, ArrayList, Map, HashMap, and Set. These collections are used to manage and store dynamic data. Additionally, the movie list, user viewing history, and collection list also possess more flexible data access capabilities.

We use classes in the java.io package to implement file input and output operations. For example, InputStream, FileInputStream, InputStreamReader, and BufferedReader are used to read CSV files line by line, while OutputStream, FileOutputStream, OutputStreamWriter, and BufferedWriter are used to update and rewrite data files. This design enables CSV files and source code to be stored independently and still be reliably accessed during execution. To ensure the basic security of user credentials, the system uses the MessageDigest class in the java.security package. Using this class reduces the risk of directly storing passwords in files. Besides,The passwords in the system are not stored in plain-

text form. Before being written to the CSV file, the passwords are processed using the hash algorithm provided by the MessageDigest class.

The system builds the graphical interface using JavaFX. The Application class serves as the entry point for JavaFX programs, while Stage and Scene provide top-level windows and scene containers. Layout classes like VBox and HBox are used to organize interface components, and TableView combined with PropertyValueFactory is used to display movie information in a table format. Event handling is implemented using ActionEvent and EventHandler, supporting user interaction, such as button clicks and navigation.

These libraries collectively form the technical foundation of the system, achieving the division of labor for data processing, application logic, and user interface design.

3 Recommendation Algorithm

3.1 Input Data

The recommendation algorithm mainly relies on two data sources. The first one is the user viewing history, which is stored in the user class as a list of movie objects. Each movie contains attributes such as title, genre, rating, and description. This algorithm mainly bases its recommendations on the genres of the movies in the viewing history. The second one is the complete movie list, where all the movies are obtained from the CSV file loaded by the file processing program. After determining the movie types that the user prefers, the system will extract movies of the matching types from the movie list. These two inputs can cooperate with each other. The user viewing history can reflect the user's preference types, while the movie list can provide movies that meet the needs based on the preference types.

3.2 Algorithm Logic

The recommendation method of this system is based on the frequency of movie types. Its logic starts by checking the user's viewing history, which is stored in the user class in the form of a list of movie objects. For the movies the user has watched, the system intelligently extracts the type of the movie and calculates the frequency of each type. The type with the highest frequency will be set as the user's preferred type. After determining the preferred type, the algorithm processes the complete movie list loaded from the CSV file by the file processing class. Then, it traverses all available movies and selects those that match the type. This filtered list is all the movies that meet the user's preferences. Therefore, this algorithm mainly runs in two steps. The first step is to analyze the frequency of each type, and the second step is to filter out the available movies based on the known preferred types. Although this algorithm is simple and focuses on movie types, it combines the user's historical records with the movie dataset to provide recommendations that reflect the

user's preferences.

In addition, the system also has some functions to enhance the availability and robustness of the engine.

Firstly, the movie list will be displayed through the JavaFX TableView component. This component has built-in sorting functionality, and users can manually sort the movies by rating, title, etc. by clicking the corresponding table headers. Therefore, the movie rating, as a filtering condition for recommendations, becomes redundant. To simplify the recommendation process, this filtering function has been removed.

Secondly, the recommendation engine can sort based on the frequency of types in the viewing history, rather than only considering the single most frequent type. It will recommend from the highest-ranked type. If all the movies of this type have been added, the system will switch to another type. This mechanism ensures that when the user has selected all the movies of the preferred type, the recommendation list will not be empty.

Finally, when all the movies of the preferred types have been watched or added to the list, the engine will not return any movie results. At this time, the system will display a window to prompt the user to explore more interesting movies. This can prevent users from being confused by an empty recommendation page.

4 Testing Methods and Results

4.1 Testing Approach

During the development process, we conducted thorough tests to evaluate the accuracy of the system. The purpose of the tests was to ensure that the results of each project module met expectations and to guarantee smooth collaboration between projects. Additionally, since the project involved interactive elements and generating recommendations, the tests focused mainly on the consistency of behavior and data.

During the development process, we checked each core function through repeated tests, such as browsing movies, user login, managing watchlists, and generating recommendations. The implementation of the tests was through interaction with JavaFX and observing the correct responses of the system to different types of user operations. In addition, we also paid attention to unexpected situations, such as incorrect user passwords during login or attempts to add non-existent movies to the list.

This process also involved the verification of file operations. After performing the operations, we checked the contents of the CSV files to ensure whether they were updated or stored. For example, after adding a movie to the list, we would check if the entry was displayed in the file. This ensured that the FileHandler executed accurately and that the data remained consistent in the system. In summary, the combination of gradual manual testing and repeated verification can more effectively ensure the stable operation of the system.

4.2 Test Cases

To assess the stability of the system, we conducted multiple representative test cases during the development process. These test cases focused on the core functions of the program, such as user login, browsing movies, managing watchlists, recording viewing history, and generating recommendations.

The first test scenario was the user login section. The system was tested with valid and invalid credentials to ensure that only registered users could log in and use the program. When entering incorrect usernames or passwords, the system would prompt for access, and otherwise allow the user to enter the main interface. Another test case involved browsing and selecting movies. When loading the movie list from a CSV file, the system would correctly display the included movies. Similarly, the operations of selecting a movie and adding it to the watchlist or history record were also tested to ensure that the data would be updated in the user's corresponding list and written back to the CSV file correctly after the operation.

The management of watchlists was tested by adding and deleting movies. We repeated these operations to ensure that the system could continuously update the user's stored data and that the list would always reflect the latest status.

For the test of viewing history, we followed a similar process. The test ensured that the movies added to the history record could be immediately displayed on the history page, and the system would store the updated information and provide it for subsequent sessions to use.

Finally, for the recommendation function, we conducted tests on users with different viewing histories. The system can analyze the stored historical records and generate recommendation lists based on the more frequently occurring types. Additionally, to further improve the system, we also conducted tests on users with very little or no history. Although the recommendations are limited due to insufficient data, the system's behavior still meets expectations.

Through these test cases, it has been proven that the core function of the system operates correctly under normal circumstances, and the data between the interface, components, and CSV storage also works as expected.

4.3 Screenshots of Program Execution

This picture shows the first interface of our system, which is the first interface that all users will see when they open the system. This interface includes functions for login, new user registration, and logout.

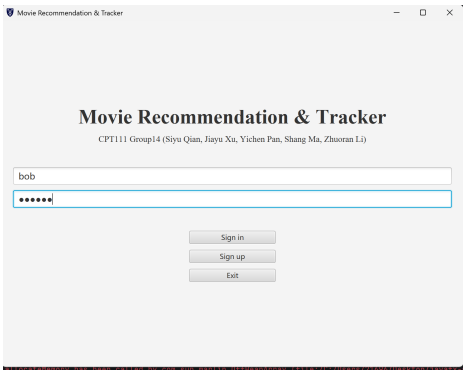

Figure 1: login interface

This picture shows the interface for new users to register.

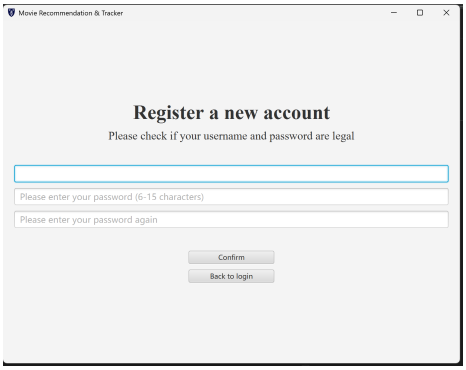

Figure 2: new user

This picture shows the main interface of the system. The main interface features four functions: viewing history, viewing list, recommendation engine, and browsing movies. Below this interface, there are also functions for exiting, changing passwords,upgrading the account and switching themes. Moreover, after entering each function interface, there is a button to return to the main interface.

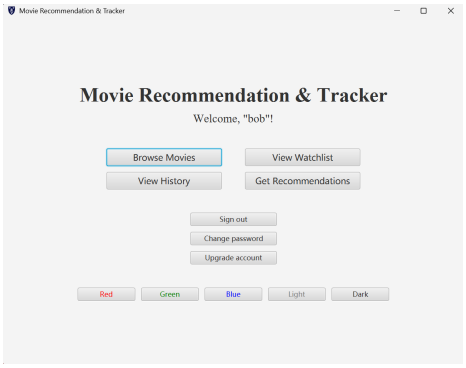

Figure 3: main interface

This picture shows that the system enables users with higher levels to have the option to customize the interface, such as choosing themes.

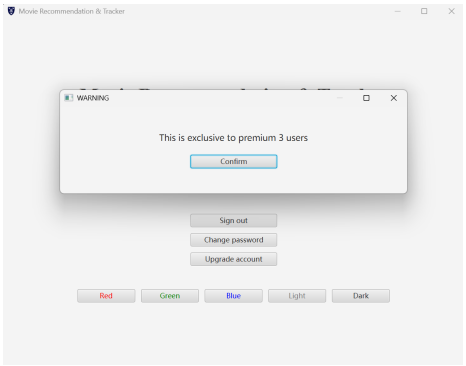

Figure 4: switch theme

These two pictures show browsing the movie list and adding movies to the user's watch list. The first picture shows the movie list, and the second picture shows the pop-up window that will appear after a movie is successfully added to the user's watch list.

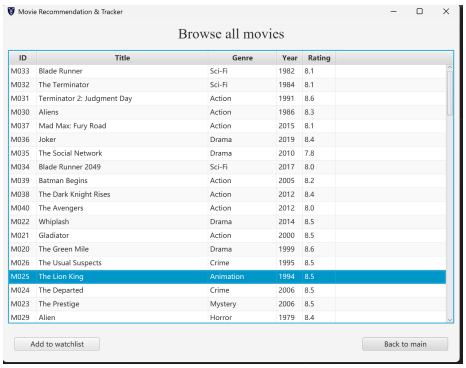

Figure 5: browse movie

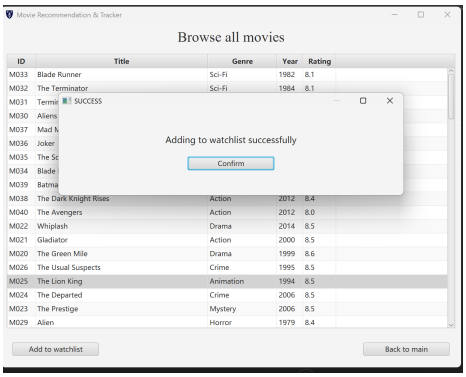

Figure 6: add movie successfully

This picture shows that the maximum number of videos that a first-level user can borrow is five. To enjoy more videos, the account needs to be upgraded.

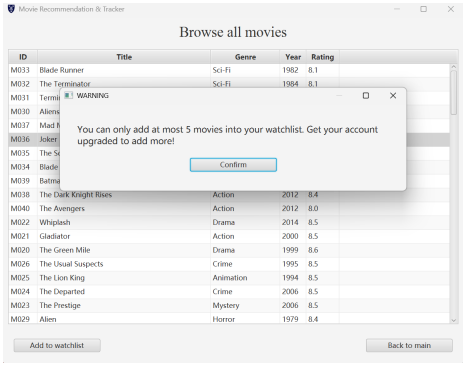

Figure 7: limited function

The following picture shows the user's watch list. Below the interface, there are two options: to remove movies from the list and to mark movies that have been watched. Once the operation is successful, a window will pop up to notify the user.

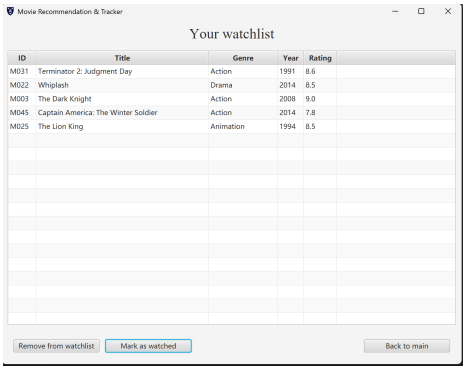

Figure 8: watchlist

This picture shows the viewing history. Just like in the viewing history, there is also the function of removing movies.

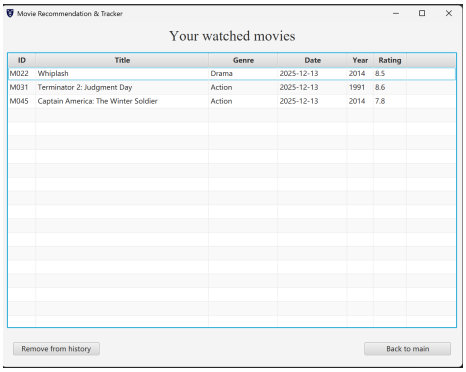

Figure 9: history

This picture shows the recommendation function, and the library has a built-in sorting feature. Clicking on "title" will automatically sort the content.

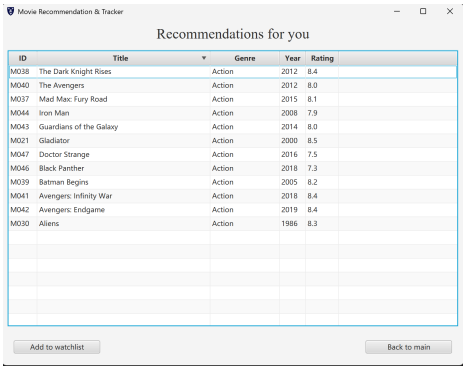

Figure 10: recommendation

This picture shows the function of changing passwords. The pop-up window indicates that only students with a grade of 2+ have the right to modify it.

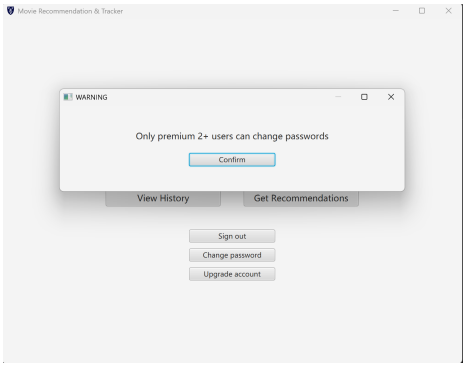

Figure 11: change password

This picture shows the function of upgrading the account. Users need to answer the questions in the interface to qualify for the upgrade.

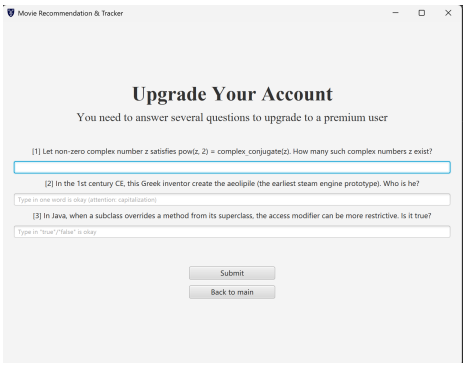

Figure 12: upgrade account

4.4 File I/O Errors

This system relies on CSV files to load user data and movie information. Therefore, any errors related to the files will affect the stability of the system. During operation, the FileHandler class is responsible for reading and writing these files and includes basic exception handling mechanisms to prevent the program from crashing when the files are inaccessible or missing. If an exception occurs during file loading, the system will capture the exception and prevent the program from suddenly stopping. This enables the program to continue running even if some data cannot be retrieved. Although the current implementation protects the system from immediate failures, specific recovery options are not included, such as restoring damaged files. However, the current response mechanism provides the most basic guarantee for handling unexpected I/O issues.

4.5 Invalid User Input

The system also considers another situation during execution: invalid user input. The JavaFX responsible for interacting with the user, including the login, movie and so on , all contain relevant checks to ensure that the user's operations are within the manageable range. For example, when logging in, it checks whether the input username and password match the records. Similarly, if the user wants to add a film to the watch list, the system will confirm whether the film exists in the movie list loaded from the CSV file. These checks can prevent errors that may occur due to referencing invalid or non-existent items. Although the system currently does not provide detailed warning information, the current mechanism can prevent the most common misuse cases. When users attempt to perform restricted functions such as changing passwords, the system will also verify the level of permissions to ensure that only authorized users can carry out the operation.

4.6 Preventing Crashes & Recovery Mechanisms

Throughout the program structure, the system uses exception handling to maintain stability and prevent the application from crashing due to unhandled errors. The FileHandler class includes try-catch blocks around all major file operations to ensure that operations such as reading and writing CSV files do not cause the system to unexpectedly stop. Although this system does not include advanced recovery mechanisms, such as backup recovery or user notifications, this basic level of protection enables the system to operate reliably under normal conditions.

5 Team Management Practices

5.1 Work Allocation

For the allocation of responsibilities and tasks, we made it based on the strengths of each team member and their familiarity with different development areas. Xu Jiayu was responsible for implementing the registration function and password modification function, ensuring that the system could effectively manage user account updates.Ma Shang is responsible for testing optimization and adding comments to the code. Pan Yichen focused on implementing the front-end components of the movie recommendation tracker, ensuring that the Java FX interface was both aesthetically pleasing and practical, and guaranteeing a good user experience. Qian Siyu was responsible for the core business logic and developing the back-end functions supporting the four main modules of the system. Li Zhuoran was responsible for writing the project report, ensuring that the documentation accurately reflected the design and implementation of the system. This task division ensured that each member contributed to the specific aspects of the system, enabling the team to work efficiently and avoid redundant work.

5.2 Collaboration Tools

To maintain continuous communication throughout the development process, we used online communication tools such as WeChat for daily communication, as well as to show the technical progress of individual tasks. When discussing task allocation or technical implementation, we used Tencent Meeting. During the integration of code, the team also adopted version sharing or file sharing methods, which were conducive to maintaining the uniformity of project files. These tools ensured the smooth progress of the development work and reduced inefficiency caused by improper communication.

5.3 Difficulties & Resolutions

During the entire project development, some problems were encountered, such as misunderstandings about the requirements of their own tasks or prob-

lems in technical implementation. To solve the problems, team members used real-time communication tools to promptly and candidly explain the situation that occurred, and solved the current problems after discussion. When facing technical challenges such as synchronizing front-end and back-end behaviors, team members would test the components together and adjust the implementation method to ensure that all components were smoothly integrated into the final system.

6 Ethical Assessment

6.1 Privacy Concerns

The system generates recommended content by tracking and analyzing user viewing records. Although this may improve the accuracy of recommendations, these preferences also reflect users' personal habits and other privacy information. If not protected, some information may be leaked or analyzed without the users' knowledge.

Furthermore, the FileHandler class does not implement any form of access control or encryption when loading and updating user data, which increases the risk of privacy leakage. The system does not include permission checks, meaning that any internal or external personnel who can access the CSV file can freely modify or extract user information. These privacy issues highlight the necessity of improving data protection measures.

6.2 Possible System Abuse

Although this system is used to provide recommendation information, without appropriate protection measures, the information is prone to being leaked or abused. A potential problem is that user data can be manipulated. Since the system relies on plain CSV files, and these files have no access restrictions, anyone who obtains access to these files can deliberately tamper with the information, such as changing viewing history or modifying favorite lists. This unauthorized modification not only infringes on user privacy but also makes the recommendation results unreliable.

Another possible form of abuse is to exploit the recommendation mechanism. Since the algorithm relies on the frequency of type occurrences, external parties may repeatedly insert specific movies into the user's history file, affecting the recommendation output. This may be used to promote certain movies or make the system biased towards certain types, thereby undermining the fairness of the recommendation process.

In addition, apart from simple username and password matching, there is a lack of other authentication methods, which means that if someone obtains access to the CSV file, they can directly modify the content of the file that records user information to impersonate other users.

6.3 Solutions and Mitigations

To address the privacy risks and potential abuse issues mentioned above, several measures can be taken to enhance the security and reliability of the system.Access to the CSV files should be restricted to prevent unauthorized reading or modification. Additionally, migrating the files to a protected storage environment will reduce the risks of data leakage and tampering.

Secondly, to prevent manipulation of the recommendation results, the system should include verification mechanisms to ensure that changes to the user's history or viewing list are only made by authorized users within the application.

Overall, these technical improvements and enhancements can jointly enhance the security, fairness, and anti-manipulation capabilities of the system.

7 Conclusion

This project successfully completed the implementation of the movie recommendation and tracking system, supporting core functions such as user login, browsing, managing lists, recording viewing history, and receiving recommendations. The entire project was developed using Java technology and combined CSV for data storage and update. During the establishment of the entire system, we applied the object-oriented programming learned in the course and combined technologies such as file input/output, exception handling, and structured design. These knowledge helped us build a more complete and running smoothly system. Through the construction of this movie recommendation and tracking system, we have gained a deeper understanding of the design and implementation of recommendation systems, and also recognized the importance of recommendation algorithms in real-world applications.