Caleb Goh - Project Portfolio

PROJECT: `EzWatchlist`

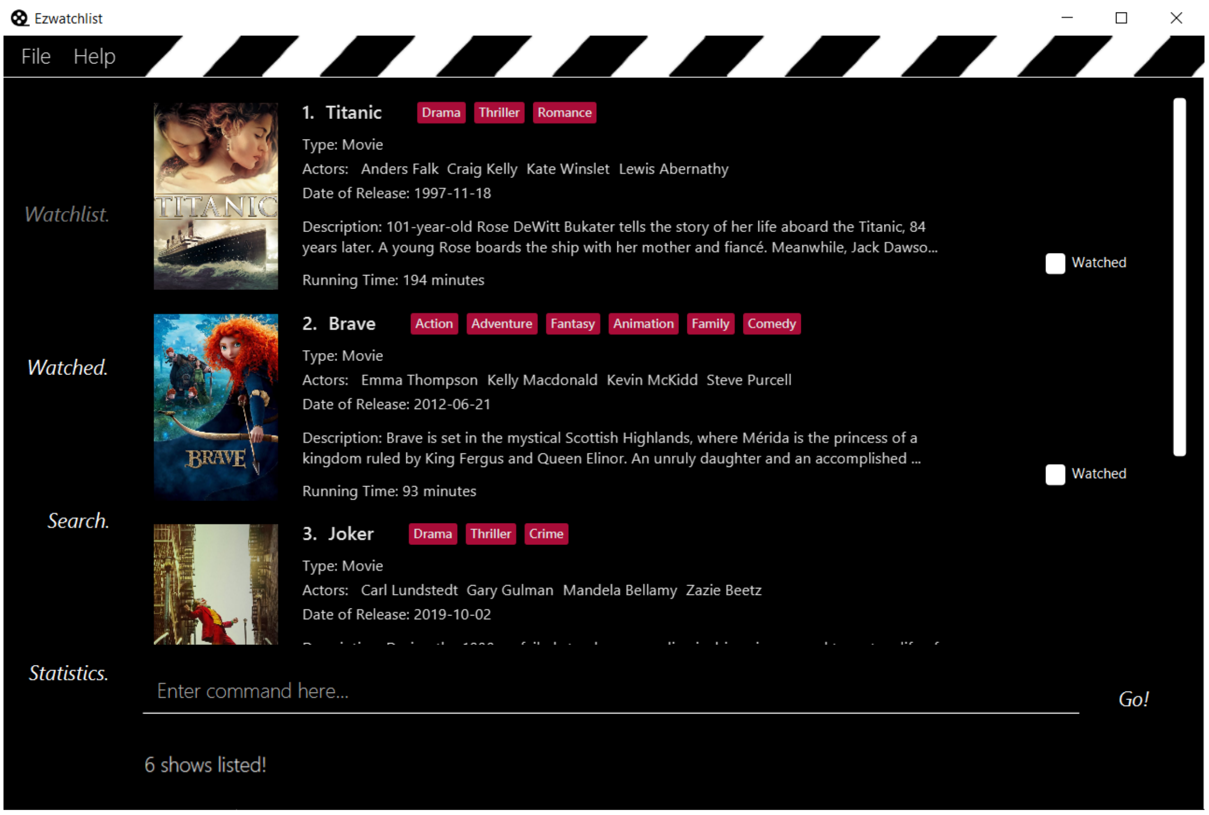
1. Introduction

The following Project Portfolio documents my contributions to the software project, `EzWatchlist`. Which was a project my team of software engineering students and I decided to pursue for our Software Engineering project.

Our team were initially tasked with enhancing a basic command line interface (CLI) desktop application that functioned as an Adressbook. What this means is that the application focused on typing as the main mode of use, and had an existing rudimentary base of code already functioning.

Ultimately, we made the decision to modify it into a management application for Movies and Tv Shows named `EzWatchlist`.

1.1 Overview



1. Titanic Drama Thriller Romance
Type: Movie
Actors: Anders Falk Craig Kelly Kate Winslet Lewis Abernathy
Date of Release: 1997-11-18
Description: 101-year-old Rose DeWitt Bukater tells the story of her life aboard the Titanic, 84 years later. A young Rose boards the ship with her mother and fiancé. Meanwhile, Jack Dawso...
Running Time: 194 minutes

2. Brave Action Adventure Fantasy Animation Family Comedy
Type: Movie
Actors: Emma Thompson Kelly Macdonald Kevin McKidd Steve Purcell
Date of Release: 2012-06-21
Description: Brave is set in the mystical Scottish Highlands, where Mérida is the princess of a kingdom ruled by King Fergus and Queen Elinor. An unruly daughter and an accomplished ...
Running Time: 93 minutes

3. Joker Drama Thriller Crime
Type: Movie
Actors: Carl Lundstedt Gary Gulman Mandela Bellamy Zazie Beetz
Date of Release: 2019-10-02

6 shows listed!

Figure 1. The graphical interface for `EzWatchlist`.

`EzWatchlist` is an application that saves movies and television shows the user adds into a watchlist. The user interacts with `EzWatchlist` through typing commands or the graphical interface seen in Figure 1. This enhanced application boasts the following features:

1. Keeps track of the movies and television shows that they planed to watch.
2. Keeps track of the movies and television shows that they have watched.
3. Allows them to update the episodes they are at each television show.
4. Gets information about their watching habits.
5. Gets recommendations for movies and television shows based on their personal taste.

NOTE The following symbols, abbreviations and formatting used in this document:

API - Application Programming Interface	An interface as a intermediary to use other programs or software.
`Highlighted words`	Our application related components such as commands, classes, methods, e.t.c.
Keyboard characters	Keyboard characters on the desktop.
#number	Pull request links based on their serial number.

The following sections illustrate these enhancements in more detail, as well as the relevant documentation I have added to the user and developer guides in relation to these enhancements.

2. Summary of contributions

This section shows a summary of my role in the project, my technical ability to code, document, and other beneficial contributions to the team project.

Role

My main technical role was to design and write the codes for the integration with the application with an online database of movies and tv shows. This was highly integral due to the nature and vision of the application where information about movies and tv shows should be readily available to the user.

Moreover, I took the supervising role of team lead in this project. I made sure that the team was well coordinated and that we were all clear on the overall vision of the application at every point in the development. Ensuring clear communication throughout the team, so that we all knew what to do. And that issues that arrived were handled smoothly and cleanly.

Major enhancement:

Added the ability for the application to integrate with an **online database of movies and tv shows**.

What it does:

1. Allows the user to see images in the application.
2. Retrieve information about movies and tv shows.

3. Search for movies and tv shows from a large up-to-date database.
4. Access reviews, ratings, recommendations, collections of movies and tv shows, genre information, and much more.

Justification:

This feature improves the product significantly because a user should be encouraged to use the application by providing appealing images, comprehensive information and up-to-date data.

Highlights:

This feature required a robust knowledge on how to set up third party libraries in a Java application. Json conversion (Online data retrieval) is also another key element when considering the API used.

Moreover, I had to integrate this external data with our own internal systems. Meaning my teammates had to use my implementations easily. Hence, data handling and conversion had to be designed in an understandable manner.

Credits:

We are using The Movie Database (TMDB) API to retrieve information on movies and tv shows. As well as using a java wrapper for the TMDB api implemented by Holger Brandl.

Minor enhancement:

Added ability to **store and retrieve images** in the application.

- Justification: This feature improves the visual interface, and provides movies and tv shows a more potent identity compared to just words.
- Highlights: A lot of consideration had to be placed in the retrieval and presentation of images. I had to integrate the image portrayal with the `JavaFX`-based interface, consider how to store images and retrieve them when needed.

Minor enhancement:

Added ability to get **recommendations** based on the user’s personal movies and tv shows.

- Justification: This feature improves the product because a user can gain the added benefit of getting more suggestions for Movies and Tv Shows, which makes the application more robust. Moreover, it improves the value of the application encouraging usage of the application.
- Highlights: This design for this enhancement had considerable thought into the efficiency of this feature due to the large amount of processing required from parsing the user’s data. There is also the subjective manner of knowing what to recommend from a large list.
- Credits: The Movie Database (TMDB) API

Code contributed:

All of my contributions to the application can be found in the following link: [My Code Contribution]

Other contributions:

Project management:

- I am the Project lead, thus my job consisted of making sure the team coordinated, the overall vision of the application was reached, and handling all the issues that arose in the development.
- Managed releases `v1.1` - `v1.4` (3 releases) on GitHub
- Managed and assigned the issues, milestones and project board on Github.

Enhancements to existing features:

- Updated the application’s model to integrate seamlessly with the external library’s data. This meant that each internal model class had to be error-friendly due to the unpredictable manner of the online database.
- Added more internal classes such as `Poster` and `Genre` to be used in the application. (Commits Poster and Genre)
- Wrote additional tests to increase coverage.
- Helped to integrate the API code with the rest of the application.

Documentation:

- Did cosmetic tweaks to our website. Pull request #224

Community:

- Reviewed Pull Request : #107, #30, #73, #51, #78
- Reported bugs and suggestions for other teams in the class: T11-3 team, F12-4 team

Tools:

- Integrated a third party library (TMDB) to the project (TMDB Api)
- Integrated a new Github plugin (Java wrapper) to the team repo.
- Added a successfully merged pull request to the Java Wrapper we are using in our application to fix their issue of not supporting recommendations. (Pull request merged)

3. Contributions to the User Guide

Our `EzWatchlist` user guide consists of instructions to the user on how to use our application. It displays my ability to document in an engaging and readable format.

I wrote the introduction to ease the user into the main features of the application, breaking it down to navigation and interaction with our program.

The following is an excerpt from our `EzWatchlist` User Guide, showing additions that I have made for the Introduction. The full introduction can be found here:

Introduction

Tired of using multiple sources on the internet to keep track of all your movies and tv shows that you want to watch? EzWatchlist is a solution to your entertainment problems!

`EzWatchList` is an application is for cinephiles who **prefer to use a desktop for managing movies and tv shows**. More importantly, `EzWatchList` is **optimized for those who prefer to work with a Command Line Interface (CLI)** while still having the benefits of a Graphical User Interface (GUI).

1. If you can type fast, `EzWatchList` works faster than traditional GUI apps.
2. Keep track of movies and tv shows that you plan to watch or have watched.
3. Access quick information about your favourite shows through our online database.
4. Discover your personal movie habits, with our statistics page.
5. Can't find any good shows to watch? We will give you recommendations on what to watch based on your `EzWatchlist` usage.

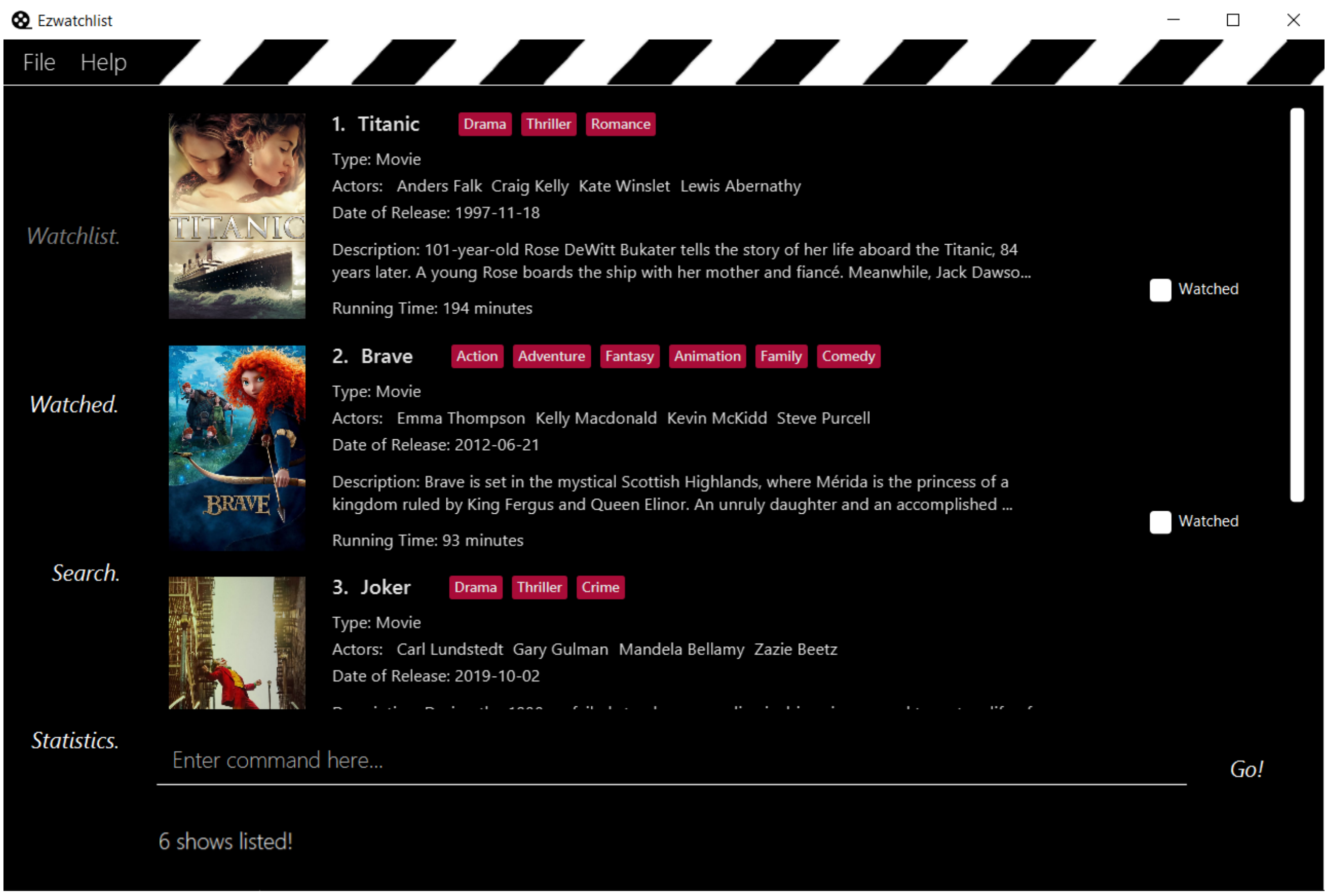
Already interested? Jump to [Quick Start] to quickly get started. Enjoy!

What else does EzWatchlist offer you?

- Integration with an online database of shows, allowing you to search and get information about shows.
- Functions and works without internet.
- No installations required.
- Auto-saves all data.
- Dual-purpose interface, command line interface for those who prefer typing and graphical for those who prefer using a mouse.

How do I use EzWatchlist?

EzWatchlist was built with ease in mind hence our interface reflects this design philosophy.


Ezwatchlist

File Help

Watchlist.

Watched.

Search.

Statistics.

1. Titanic Drama Thriller Romance

Type: Movie

Actors: Anders Falk Craig Kelly Kate Winslet Lewis Abernathy

Date of Release: 1997-11-18

Description: 101-year-old Rose DeWitt Bukater tells the story of her life aboard the Titanic, 84 years later. A young Rose boards the ship with her mother and fiancé. Meanwhile, Jack Dawso...

Running Time: 194 minutes

Watched

2. Brave Action Adventure Fantasy Animation Family Comedy

Type: Movie

Actors: Emma Thompson Kelly Macdonald Kevin McKidd Steve Purcell

Date of Release: 2012-06-21

Description: Brave is set in the mystical Scottish Highlands, where Mérida is the princess of a kingdom ruled by King Fergus and Queen Elinor. An unruly daughter and an accomplished ...

Running Time: 93 minutes

Watched

3. Joker Drama Thriller Crime

Type: Movie

Actors: Carl Lundstedt Gary Gulman Mandela Bellamy Zazie Beetz

Date of Release: 2019-10-02

Enter command here...

Go!

6 shows listed!

Figure 1. EzWatchlist's graphical interface

Navigation

EzWatchlist is split into four different pages:

1. **Watchlist**, where shows you want to watch are added.
2. **Watchedlist**, where shows you have watched are located.
3. **Search**, where you search for shows both online and offline.
4. **Statistics**, where information about your viewing habits are shown.

This pages are represented by the **sidebar panel** shown visibly in Figure 1. Navigate between pages by using keyboard shortcuts kbd:[1], kbd:[2], kbd:[3], kbd:[4] respectively or through a button press on the graphical interface.

Interaction

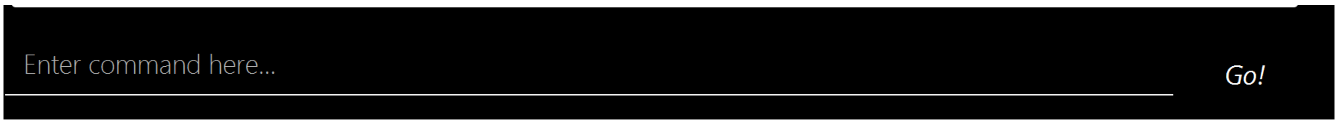


Figure 2. EzWatchlist's command line interface.

EzWatchlist uses Commands entered through the command line interface in figure 2 to interact with the application. Typing a command into the interface and entering it (by pressing kbd:[Enter] or clicking `Go!`) is the main way of interaction in the application. The following is an example of adding a movie into your watch list:

Moreover I wrote the Advanced Features portion of the user guide, available here.

Contributions to the Developer Guide

This sections showcases my contribution to the `EzWatchlist` Developer Guide, which serve to document the implementation of our features to technical users. It demonstrates my capability to use diagrams and illustrate technical details.

The following are sections for the Api Component I wrote to introduce the design architecture in the application.

API component

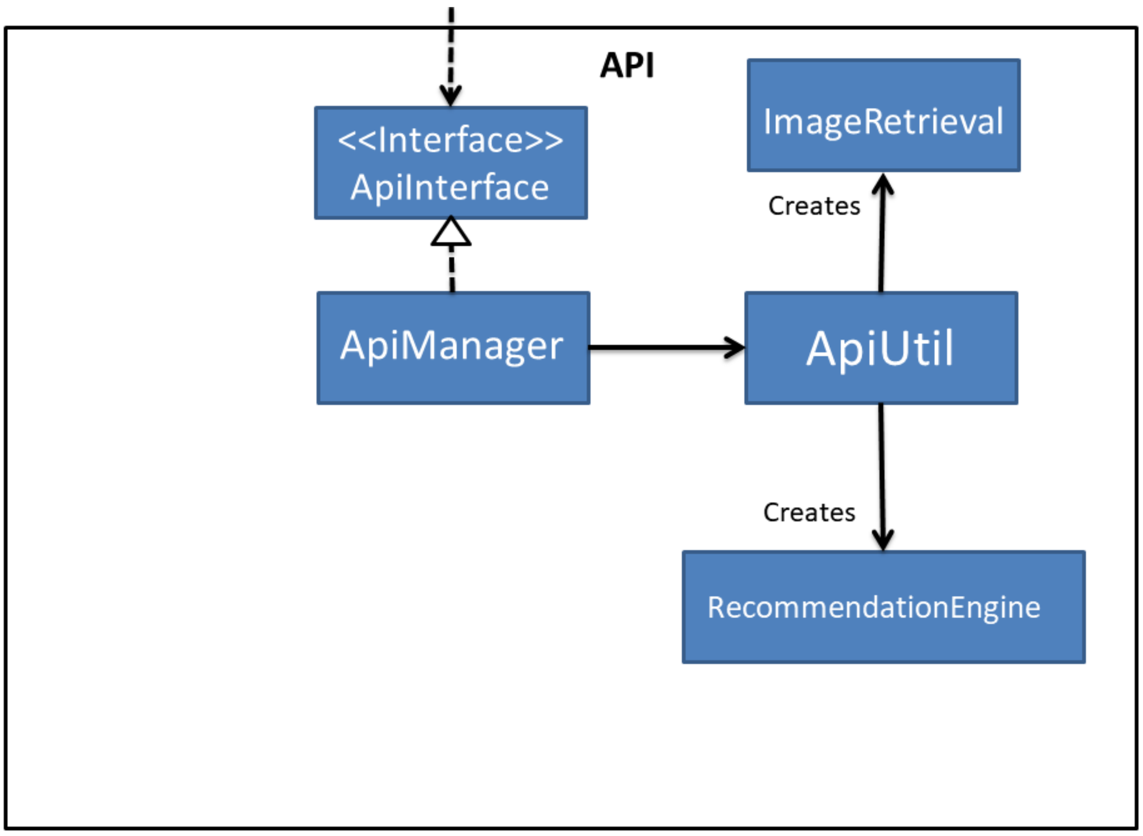


Figure 3. Structure of the Api Component

API : `ApiManager.java`

In Figure 9, we see the structure of the API component centered around `ApiManager`. Moreover:

1. `ApiManager` uses the `ApiUtil` class for static methods for data handling.
2. The `ApiUtil` class creates an `ImageRetrieval` object for retrieving images over the network and a `RecommendationEngine` object to generate recommendations.
3. `ApiManager` object encapsulated by the interface `ApiInterface` can be created at any point in the application to access the online database.

4. If no network connection can be established, an `OnlineConnectionException` is thrown.
5. `ApiManager` will not affect any of the internal logic and model in the application.

Moreover, I also contributed the description of my implementation of the Api Model classes, Image Retrieval, and generation of Recommendations. The following is only an excerpt of my design considerations. The full portion can be found here

Design Considerations

Why this implementation was chosen

- **All API interactions would be in the API package and go through ApiInterface.**
  - Pros:
    - The application wouldn't have to seek access to the database on their own. The interface should provide all the functionality needed.
    - Follows the Single Responsibility Principle (SRP) that a module in the program should be encapsulated and have one repsponsibility.

Aspect: Image retrieval implementation

Images are retrieved through the `ImageRetrieval` class. Which downloads the image into the computer, wrapping a `Poster` class with the local path of the image, for the application to access images.

Design Considerations

- **Alternative 1 (current choice):** Download the image into the computer then access it locally on the computer.
  - Pros:
    - Easier to keep track of images.
    - All images are retrieved the same way making it easier to implement showing the images.
  - Cons:
    - All images shown are currently downloaded without being deleted, hence the amount of images downloaded can get unwieldy quickly, increasing amount of memory used.
- **Alternative 2:** Parse the online url to the application for them to access the image online when needed.
  - Pros:
    - No need for the images to be downloaded allows less memory to be used by the application.
  - Cons:
    - Everytime the image is viewed, the application has to retrieve it from online making it more intensive on the network.

- If the internet connection fails, the image can no longer be viewed.

Aspect: Recommendations

Recommendations are generated through the `RecommendationEngine` class in the API package. Currently, recommendations are retrieved through the `ApiInterface` implemented in the API package.

Implementation

Given below in Figure 14 is a *sequence diagram* of how the recommendations are generated when called from the `ApiManager`.

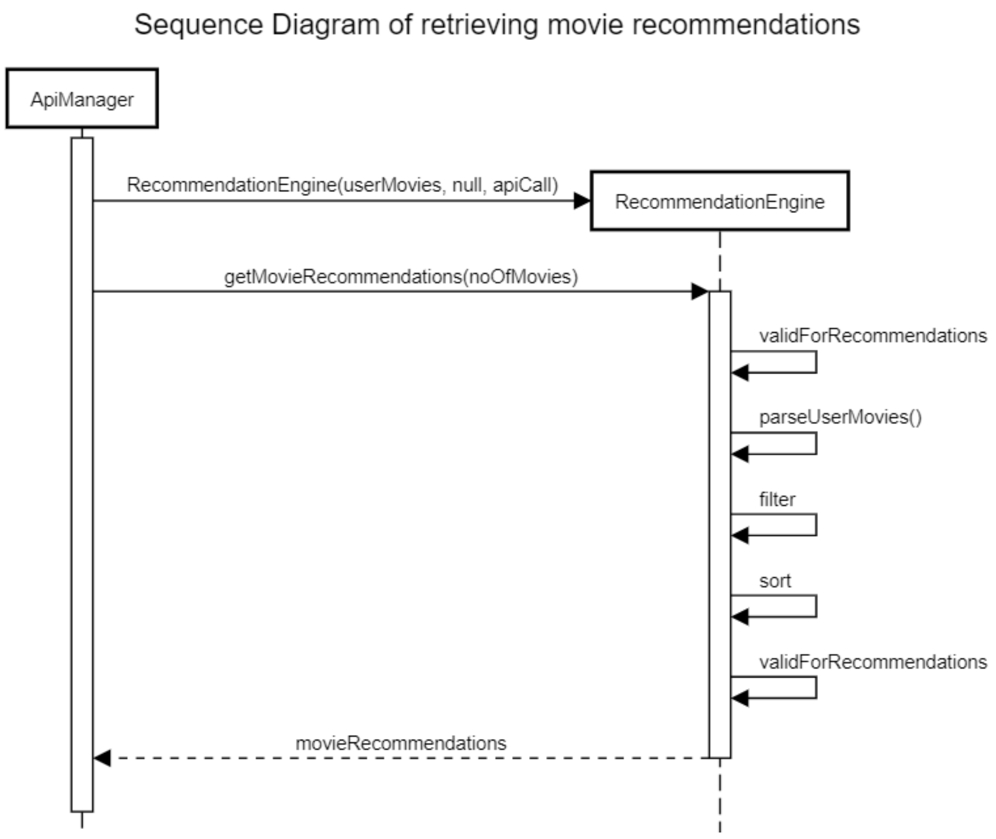

Sequence Diagram of retrieving movie recommendations
ApiManager
RecommendationEngine(userMovies, null, apiCall)
RecommendationEngine
getMovieRecommendations(noOfMovies)
validForRecommendations
parseUserMovies()
filter
sort
validForRecommendations
movieRecommendations

Figure 4. Sequence Diagram of Movie Recommendations retrieval.

Step 1. First an instance of `RecommendationEngine` is created by passing in the list of movies the user has and the **api call object** generated by the instance of the `ApiManager` object.

Step 2. The method `getMovieRecommendations(noOfRecommendations)` is called in the `RecommendationEngine` object and it will:

- Check if the list is valid to generate recommendations.
- Parse the list to get the online entries in the database, and their recommendations from the database.
- For each recommendation, store it in a *HashMap* and if there are duplicates increase the value. This counts the amount of occurrences each recommendation occurs.
- Filter the entries to remove all entries that the user already has.

Step 3. The recommendations are then sorted based on the amount of occurrences in appears in the *HashMap*.

Step 4. The recommendations are then returned in a list in which the length depends on the amount of recommendations requested.

Step 5. With the list of movies returned, the application can then display the movie recommendations back to the user.

NOTE If no recommendations can be generated, a `NoRecommendationsExceptions` is thrown. Moreover, an `OnlineConnectionException` is still thrown when not connected to the internet.