Wong Chuan Kai - Project Portfolio for EzWatchList

PROJECT: Ezwatchlist

1. Introduction

The purpose of the Project Portfolio is to document and showcase my contributions to the software project, EzWatchList.

Our team was initially tasked with enhancing a basic command line interface(CLI) desktop application for our Software Engineering project. We are also allowed to morph it to other application that uses command line interface. Thus, we chose to morph it into a movie records management system called EzWatchList. This enhanced version provides a unique, clean and simple way of organizing and keeping track of movie or TV show watch list.

1.1 Overview

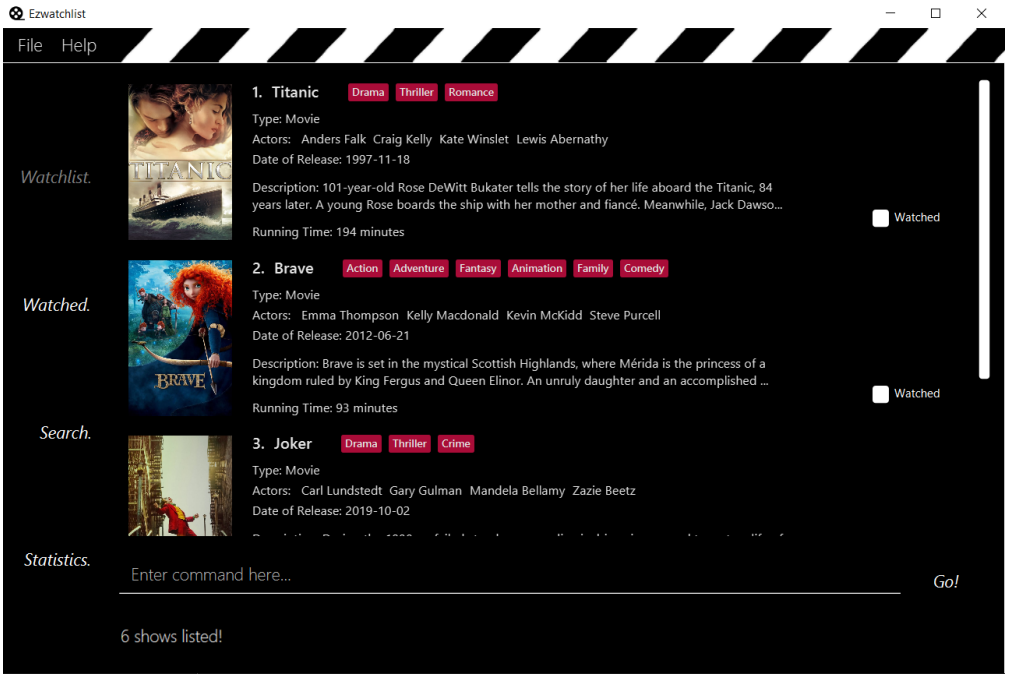


Figure 1. The graphical interface for Ezwatchlist.

EzWatchList helps users to keep track and organise movie or TV show in a watch list. User can simply interact with the application by inputting commands into the interface.

Main features of EzWatchList:

- Keep track of shows that users plan to watch

- Allows users to edit and mark down shows that they have watched
- Allows users to search for show online using `search` online function or search within user's watch list.
- Gives users statistics about their watching habits and recommend shows to them.

> Note the following symbols and formatting used in this document: This symbol indicates important information. API - Application programming interface. A sort of interaction with applications through software intermediaries. undo A grey highlight (called a mark-up) indicates that this is a command that can be inputted into the command line and executed by the application. VersionedAddressBook Blue text with grey highlight indicates a component, class or object in the architecture

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 coding, documentation, and other helpful contributions to the team project.

My role was to design and implement the `Add` and `Sync` features. These commands are essential to the application as they are the building block of the application. Before I can design these features, I have to understand the application requirement first such as functional and non-function requirements. I also have to know who my target audience is and study the use-cases properly to plan out and design these features.

Major enhancement:

Added 'add' and 'sync' commands.

Add command

What it does:

There are two functions for `Add` command. Firstly, it allows the user to add movies or TV shows into the watch list. User has to input several information of the show such as name, description, and name of actors. Beside this core function, the add command also allows user to add a movie found in search result page after user has used the `search` online feature. Search page displays the search result of the movies the user is interested to watch. The `search` feature is implemented by my team mate, Michelle.

Justification:

The purpose of this application is to allow user to track movie list. Thus, adding shows into watch list is a fundamental and core function that runs the application. If the user wants to know more about a certain movie, he can simply use the `search` function to search for it. The `search` function is

able to gather information about movies online and user can choose to ‘add’ a certain movie from the search list into his own watch list.

Highlights:

Credits:

Since we are retrieving information on shows online, we decided to use The Movie Database (TMDB) api.

Sync command

What it does:

There are two functions for `Add` command. Firstly, it allows the user to add movies or TV shows into the watch list. User has to input several information of the show such as name, description, and name of actors. Beside this core function, the add command also allows user to add a movie found in search result page after user has used the `search` online feature. Search page displays the search result of the movies the user is interested to watch. The `search` feature is implemented by my team mate.

Justification:

The purpose of this application is to allow user to track movie list. Thus, adding shows into watch list is a fundamental and core function that runs the application. If the user wants to know more about a certain movie, he can simply use the `search` function to search for it. The `search` function is able to gather information about movies online and user can choose to ‘add’ a certain movie from the search list into his own watch list.

Highlights:

Credits:

Since we are retrieving information on shows online, we decided to use The Movie Database (TMDB) api.

Minor enhancement:

Added shortcut keys to move about pages easily.

What it does:

Users are able to press a single shortcut key button to move about the panels such as ‘Watchlist’, ‘Watched’, ‘Search’ and ‘Statistics’ without moving and clicking mouse.

Justification:

Our target user is one who prefers to type fast and complete tasks quickly. One of the non-functional requirement is to reduce the usage of the mouse as this is mainly a command line application. Adding shortcut keys will reduce the usage of the mouse.

Highlights:

Code contributed:

My contributions to EzWatchList can be found in the following link: [My Code Contribution]

Other contributions:

Project management:

Enhancements to existing features :

Documentation:

Community:

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

This section is an excerpt from our EzWatchList User Guide, showing additions that I have made for the add and sync features. They showcase my ability to write documentation targeting end-users.

Adding a show from search result page: `add`

This is an extension to the `add` feature. After user has searched from the show, user can add a show found in the search result page into their watchlist.

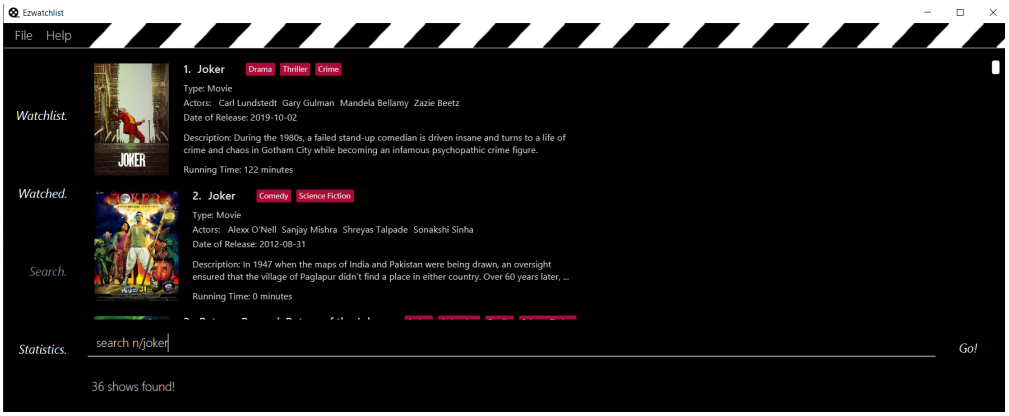
Format: `add INDEX`

INDEX is a positive integer and is limited to the number of shows found in search result page.

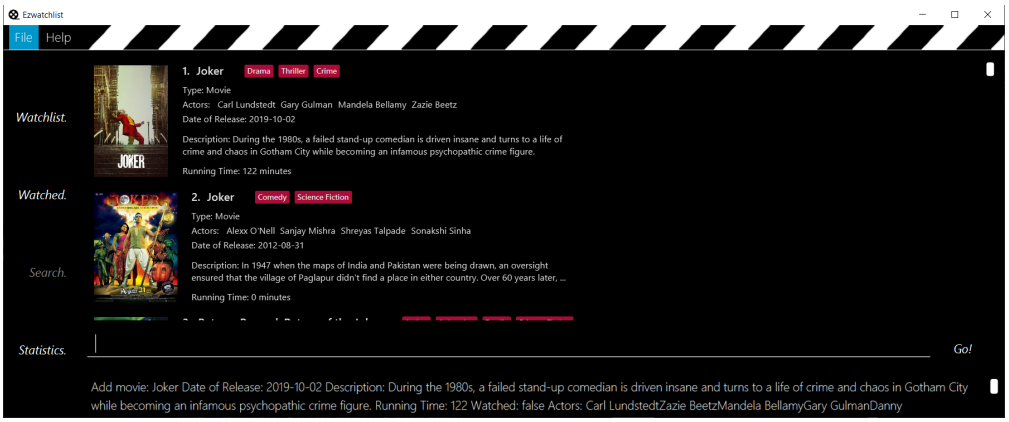
TIP	This add command can only be used if user is currently at `search page`, and has already searched for show using the search online command.

Example Usage:

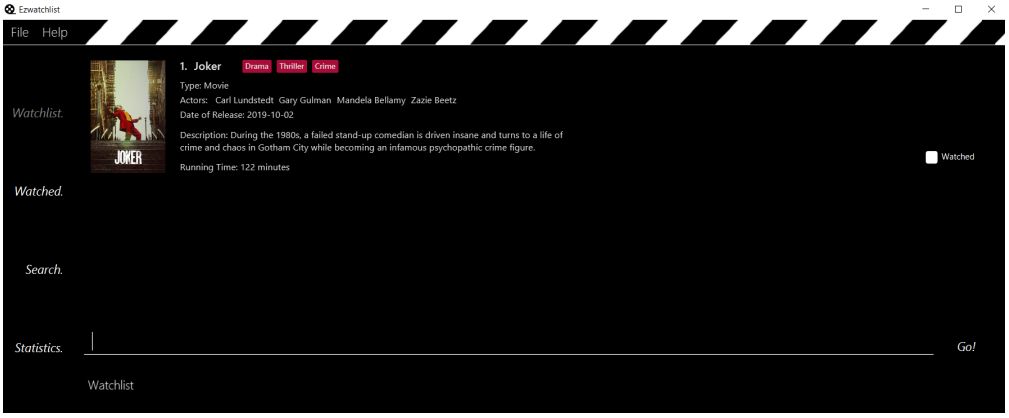
Pre-condition: User has already searched for a show using search online command.



Step 1. User click (or use keyboard `3` key) on search page. User then input `add 1` on command box.



Step 2. User click (or press keyboard `1` key) on watch list page. User will see `Joker` movie added to watch list.



Synchronise user's show data: `sync`

If user has lack of information about a certain show in their watch list, User can use `sync` command. Synchronise, `sync`, command will transfer all the information about a certain show (for example: show A) found in search result page with a show (for example: show A') that has the same name as Show A found in watch list.

Note:

1. The show in the watch list must have at least a name and type.
2. Names are not-case sensitive.
3. `Sync` will **WRITE OVER** all the information of show with same name found in watchlist.

Format: `sync INDEX`

INDEX is a positive integer and is limited to the number of shows found in search result page.

Example Usage

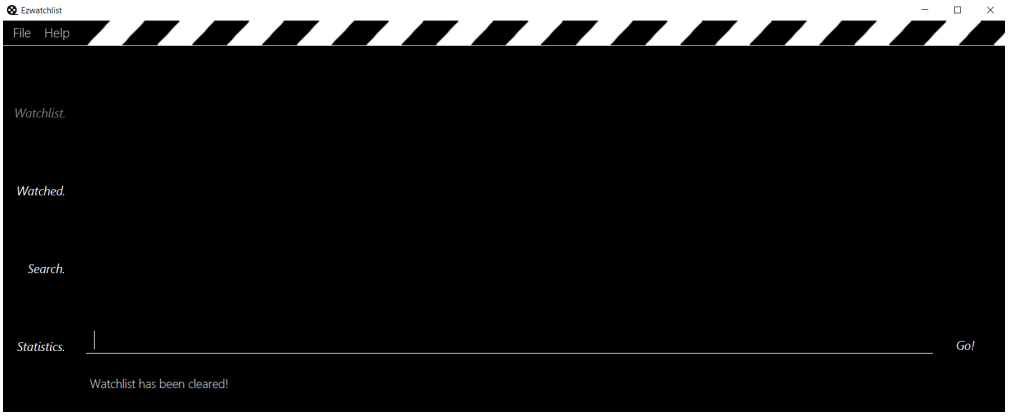
Scenario 1: User has already input 'Titanic' show into watchlist manually.

Step 1. User searches for `Titanic` show in search page.

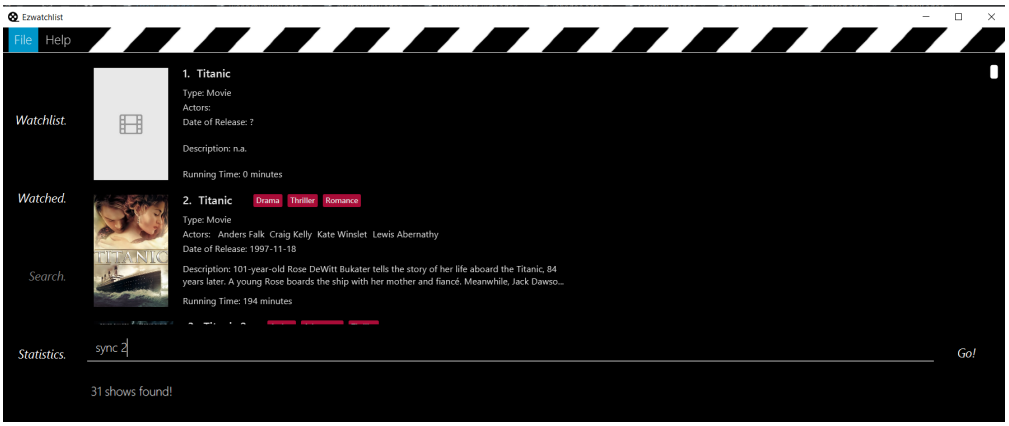
Step 2. `Titanic` result page will be displayed. User input `sync 2` to synchronise movie at index 2 of the list with a movie of same name found in watchlist.

Step 3. Go to watchlist. New information of `Titanic` in watchlist will be displayed.

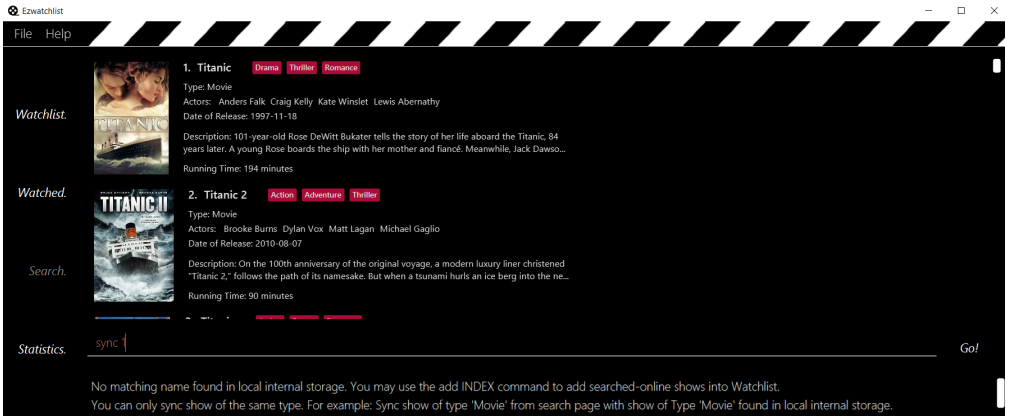
Scenario 2. User has not input Titanic show into watchlist manually.



Step 1. Similar to scenario 1, user searches for `Titanic` show in search page.



Step 2. `Titanic` result page will be displayed. If user were to sync any index, error message will be displayed because there is no show of similar name found in watch list.



Step 3. User can choose to use `add 1` command to add show of index 1 found in search result page.

Contributions to the Developer Guide

Given below are sections I contributed to the Developer Guide. They showcase my ability to write technical documentation and the technical depth of my software engineering skills to the project.

[Feature] Add feature

Implementation

The `AddCommand` extends `Command` and uses `AddCommandParser` to process the command entered by the user.

There are two different ways of using add feature and both have different outcome.

Scenario 1: Adding show in the WatchList.

Scenario 2. Adding show found from online search.

Given below is an example usage of scenario 1 and how the add mechanism behaves at each step.

Step 1. The user launches the application and executes `add n/Joker…` command to add a show in the WatchList with the name "Joker".

Step 2. Entering the command calls AddCommandParser#parse().

Step 3. A new `AddCommand` is created, with the show to be added in AddCommand.

Step 4. The `AddCommand#execute()` method is called, referencing the current model and add the show given by user to the `filteredShowList` found in model.

The following activity diagram summarises the workflow of Add:

[addactivitydiagram] | *addactivitydiagram.png*

Figure 3: Activity Diagram of `AddCommand`

In Figure 3, the user first launches the app. After the user input a add command, the program runs and add the show input by user into WatchList.

Design Considerations

Aspect: How `AddCommand` executes

- Current choice: Create a show object and add it to a `filteredShowList` found in ModelManager.
  - Pros: Easy to implement and make use of.
  - Cons: May have performance issues in terms of memory usage.

Given below is an example usage of scenario 2 and how the add mechanism behaves at each step.

Step 1. The user uses the search(Online) command and executes `add INDEX` command to add a show from search result page of INDEX in the WatchList.

Step 2. Entering the command calls AddCommandParser#parse().

Step 3. A new `AddCommand` is created, with the show to be added in AddCommand.

Step 4. The `AddCommand#execute()` method is called, referencing the current model and add the show given by user to the `searchList` of INDEX found in model.

The following activity diagram summarises the workflow of Add:

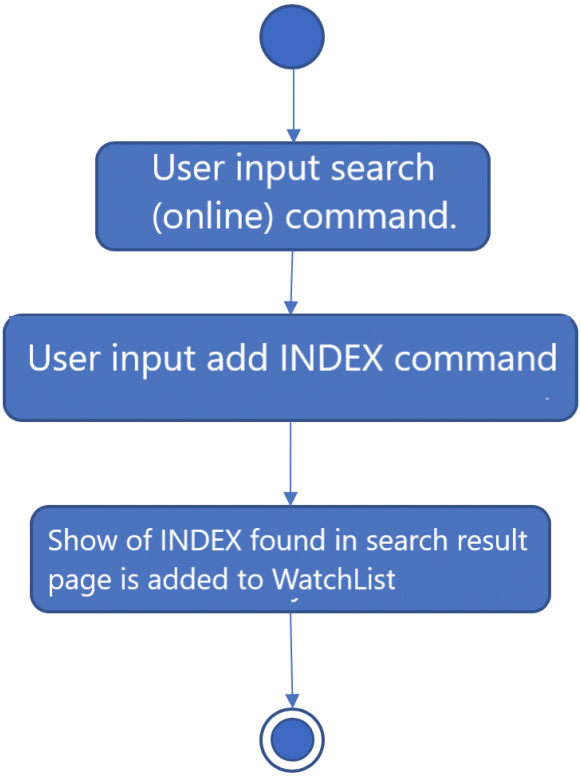


Figure 4: Activity Diagram of `AddCommand`

In Figure 4, User input search(online) command. User then input `add INDEX` command. Show of INDEX found in search result page is added to WatchList.

Design Considerations

Aspect: How `AddCommand` executes

- Current choice: Retrieve the show object found in `searchList` of INDEX from ModelManager and add it to `filteredShowList`.
- Pros: Enables for greater cohesion since there is a specific command for adding information of a show in watchlist.
- Cons: Requires longer code, and the code is also repetitive since its implementation is similar to that of the add and edit command
    - Cons: May have performance issues in terms of memory usage.

[Feature] Synchronise user's show data

The synchronise feature allows user to sync a show found in watchlist with online searched show data. It modifies all of the parameters/information is user selected show with online searched show data.
User may have added their show with their own information. However, user might not know some of the parameters such as actors. Thus, user can use the search online command `search n/` to look up information regarding that show.
Then, Synchronise command `sync` can be used to update information/modify on that show.

Implementation

The Synchronise feature is facilitated by `SyncCommand` object which can be found under the commands package. It extends `Command` and uses the `SyncCommandParser` to process the command entered by the user.

Given below is an example usage scenario and how the Synchronise command work as Sync mechanism works at each step.

Pre-Condition: User has already added a certain show into watchlist manually. That show must have at least `name` and `type` parameters. Example of Pre-Condition: User has added Titanic movie into watchlist.

Step 1. The user launches the application, go to Search page and execute `search n/titanic`.

Step 2. The user execute `sync 1` command to synchronise index 1 of result page with a show in watchlist with same name (case-insensitive).

Step 3. Entering the command calls `SyncCommandParser#parseCommand()`, which in turn returns a new `SyncCommandParser` and the `SyncCommandParser#parse()` command is called.

Step 4. A new `SyncCommand` is created, with the index of the show being parsed as a field of the `SyncCommand`.

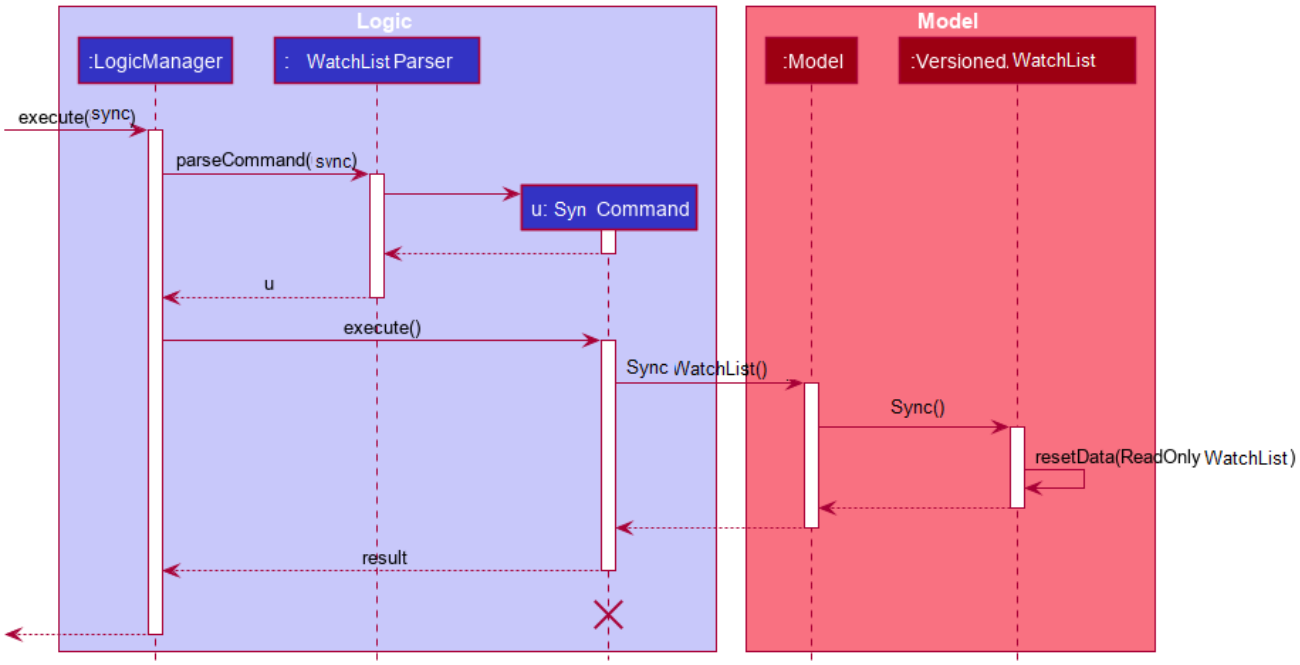
Step 5. The `SyncCommand#execute()` method is called, referencing the current `model`, and the show that is in the current `FilteredShowList` is referenced based off the current `model`.

NOTE If the `index` is out of bounds, a new `CommandException` is thrown.

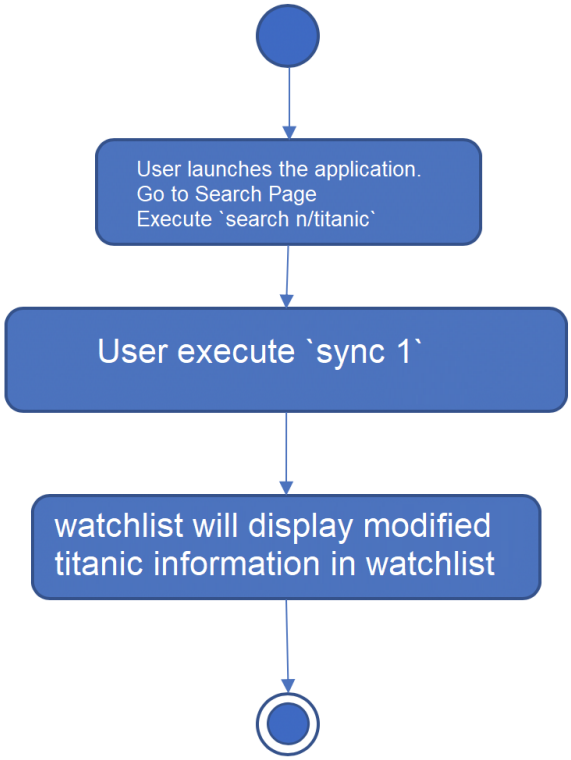
Step 6. A list of shows found in search page and watchlist are retrieved from `model`. The show according to the Index of the `searchpagelist` are retrieved as well. Then, the list of show in watchlist will be checked through to match the name of the index show.

Step 7. If a show in watchlist matched with the name of the index show, `model.setShow` will be called to replace the show found in watchlist with index show. `CommandResult` will be return which contains information regarding the feedback result. Else, `CommandException` is thrown to notify user no similar show name is found in watchlist as index show.

The following sequence diagram shows how the sync operation works:



The following activity diagram summarises the workflow of Sync:



Design Considerations

Aspect: Creating a new Synchronise instead of an altered EditCommand and AddCommand

- **Alternative 1 (current choice):** Creating a new Synchronise class for replace information of a certain show found in search page with one in watchlist.
    - Pros: Enables for greater cohesion since there is a specific command for replacing/modifying information of a show in watchlist.
    - Cons: Requires longer code, and the code is also repetitive since its implementation is similar to that of the add and edit command
- **Alternative 2:** Use the SyncCommandParser to create a new EditCommand object that edits the information of a certain show found in search page with one in watchlist.
    - Pros: Less code repetition and shorter code in general.
    - Cons: This will mean that there is less cohesion of the code and perhaps greater dependencies since more classes depend on the EditCommand class.
