Seah Lynn - Project Portfolio

1. Introduction

This project portfolio details my contributions to a Software Engineering group project, where my team and I created the Command Line Interface (CLI) desktop application, +Work.

1.1. About The Team

We are a group of 5 Year 2 NUS Computer Science students, who worked together to create the CLI application, +Work.

1.2. About The Project

+Work is a CLI application created by my team for NUS School of Computing's CS2103T Software Engineering module. In this module, each team is required to morph or enhance an existing CLI AddressBook application, to fit the needs of a target group of users.

Our application, +Work, is a project management tool aimed at NUS project leaders. It allows project leaders to manage their members, meeting times, task allocations as well as personal claims for items purchased. A user can use +Work to find the most popular meeting time among his project members when calling for a meeting, and even add project tasks and members that are assigned under these tasks, into +Work to better manage the project manpower.

1.3. Formatting of Document

Shown below are symbols that will appear in my Project Portfolio, and their significance.

IMPORTANT	Information listed in this note is essential information that users should be mindful of, in order for +Work to function as mentioned in the User Guide.
NOTE	Information listed here are things that users should take note of, so as to more efficiently use +Work.

2. Summary of Contributions

This section details all of my contributions to the project, and showcases how I worked on different aspects of the project.

2.1. Enhancements

- **Major enhancement**: I added the ability to retrieve the overall statistics of tasks and members in +Work.
    - What it does: This feature allows the user to retrieve statistics relating to tasks and members in +Work, including the number of tasks and inventory items allocated to each member, and the proportion of tasks of different statuses, along with the time spent on each task added.

    - Justification: This feature increases the functionality of +Work because it gives the user a broad overview of the project's progress and members' responsibility allocation at a glance, which is one of the main goals of effective project management.
    - Highlights: The statistics are displayed in the form of a Pie-Chart for easy comparison of data at a glance. Implementation of the statistics code also requires the use of a newly introduced mapping concept our team added to +Work, and required extensive consideration in the integration of the two implementations.
- **Major enhancement**: I added the ability for users to work with (add, delete, list, set image for) project members in +Work.
    - What it does: This feature allows both members and tasks to be managed in one application.
    - Justification: This feature improves the product significantly because it adds another aspect of project management (member management) into +Work, and allows different tasks and members to be assigned to one another. This allows users to be able to draw connections between tasks and members when managing the project.
    - Highlights: One of the unique features that `Member` objects has is the ability to be assigned profile pictures. Users can link an image on their laptop to a specific member in +Work, to set the image as the member's profile photo. This was challenging to implement because I had to ensure the User Interface (UI) would be able to accurately display the profile picture of each member.
- **Minor enhancement**:
    - Enhance the UI for members in +Work, which can include members' profile photos, as well as list tasks allocated to each member under individual member cards
    - Created the UI of statistics calculated, and made sure it was in an easy-to-read and aesthetic format
- **Code contributed**: [Functional code] [Test code]

2.2. Other Significant Contributions to Project

- Project management
    - Set up the project organisation and repository at the beginning of the project
    - Refactored sections of code to ensure the syntax of every command is consistent with +Work's User Guide (Pull request #85)
- Enhancements to existing features
    - Enhanced the UI display for tasks, to include information regarding member allocated to the different tasks in individual task cards. (Pull requests #85, #158)
- Documentation
    - Wrote parts of the +Work User Guide relating to Member and Statistics
    - Wrote parts of the +Work Developer Guide relating to Member and Statistics
- Community
    - Reported bugs and suggestions for another team in the cohort (reports)

3. Contributions to the User Guide

The following section displays excerpts of my additions to the **+Work User Guide** for the `Statistics` and `Member` features.

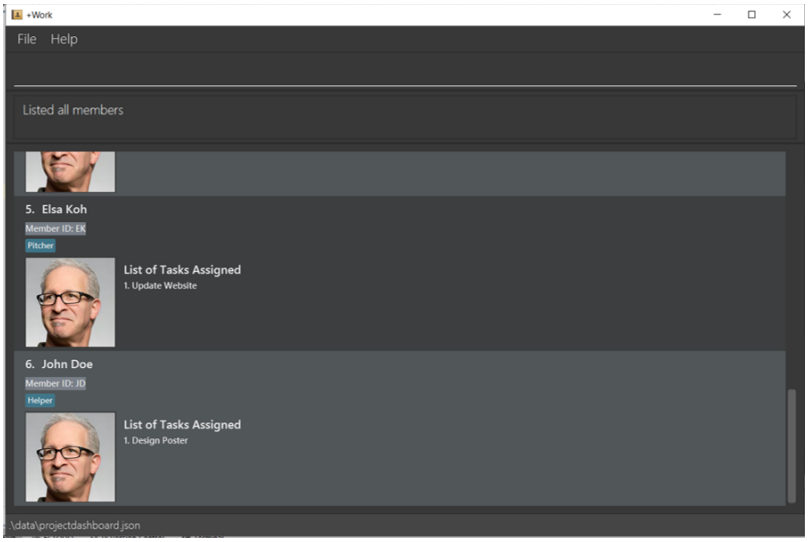
3.1. Member-related Commands

3.1.1. Adding a member: `add-member`

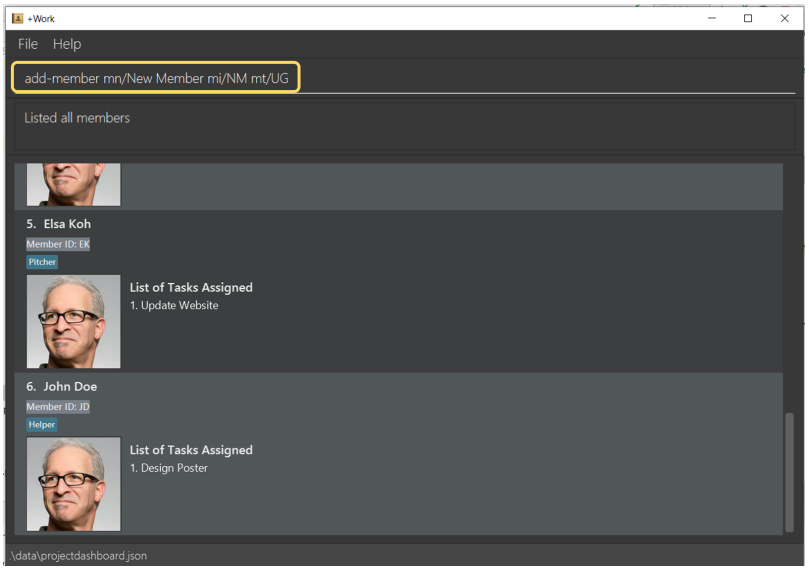
To add a member to the list of team members in +Work, use the command 'add-member' following the format: `add-member mn/MEMBER_NAME mi/MEMBER_ID mt/TAGS`

Example: `add-member mn/New Member mi/NM mt/UG` can be executed as follows:

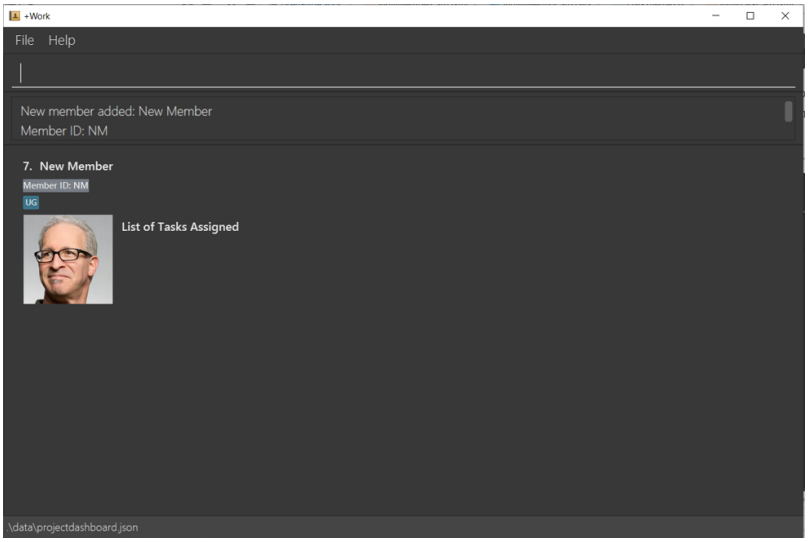
Step 1: +Work initially contains a list of 6 project members, as shown below.



Step 2: To add a new project member into +Work, you enter the command `add-member mn/New Member mi/NM mt/UG` into the command prompt box.



Step 3: After you hit `Enter`, the result box will display the message "New member added", and a new member with name 'New Member', member ID 'NM' and tag 'UG' is added to +Work.



IMPORTANT

Member ID is an alphanumeric ID set by you, and cannot be changed once the member is created.

NOTE

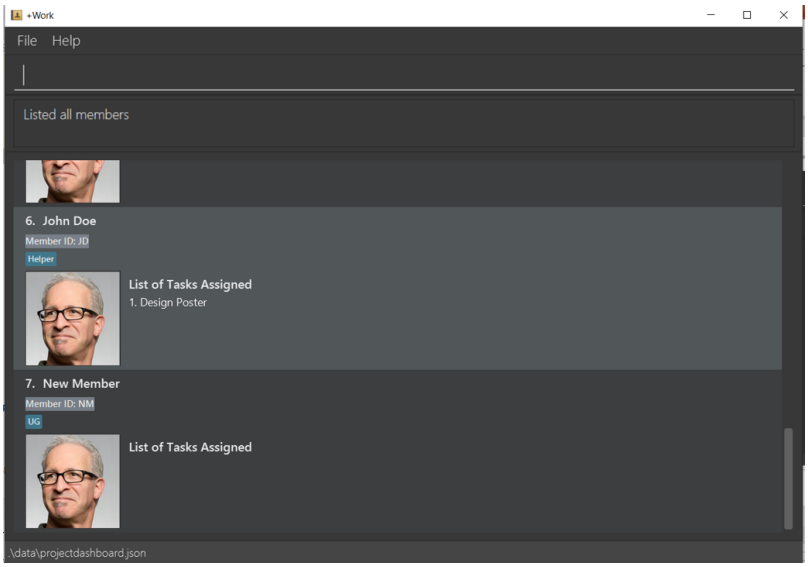
- Adding a member tag is optional in the adding of a new member.
- It is possible to add a member with multiple tags following this format: `add-member mn/New Member mi/NM mt/UG mt/DG mt/…`

3.1.2. Set image for member: `set-image`

To set a profile picture for a member in +Work, use the command `set-image` following the format: `set-image mi/MEMBER_ID im/IMAGE_PATH`

Example: `set-image mi/NM im/C:\Desktop\NewUserImage.png` can be executed as follows:

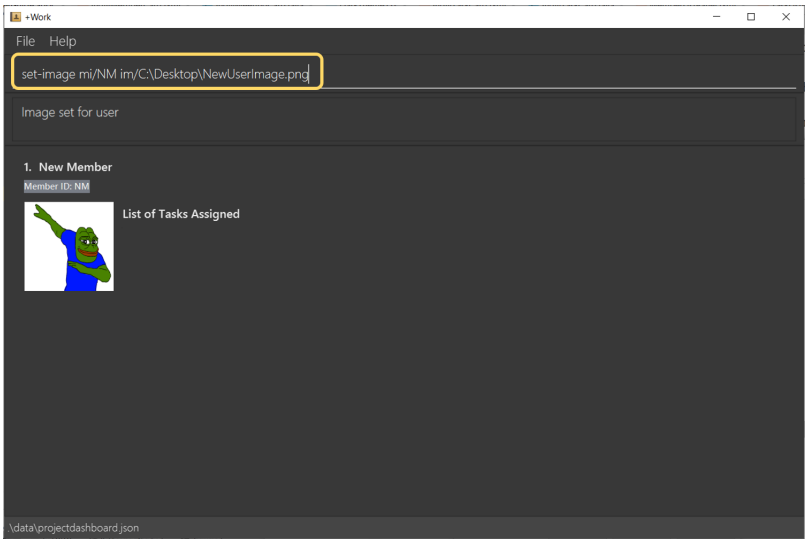
Step 1: +Work initially contains a list of project members with default profile pictures, as shown below.



Step 2: To update the profile picture of the project member with member ID 'NM' in +Work to a

specified image, you enter the command `set-image mi/NM im/C:\Desktop\NewUserImage.png` into the command prompt box.

Step 3: After you `Enter` the command, the member 'New Member' with member ID 'NM' has a new profile picture, specified by the image path you entered.

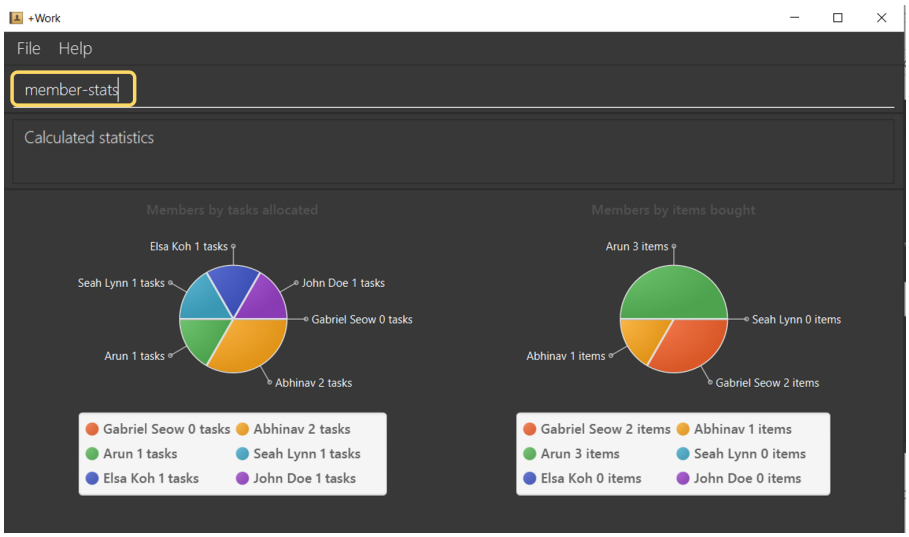


NOTE	Image Path refers to the folder path of the image stored in your computer, and should end with .png
IMPORTANT	If you shift the image's location in your computer, +Work will be unable to find the image to display, and will display a warning message, before displaying the default profile picture. It is recommended that you store all the images in a central folder to prevent this from happening.

3.2. Statistics Commands

3.2.1. Getting statistics of members: `member-stats`

To get statistics relating to the members in +Work, use the statistics command following the format: `member-stats`
Calling the `member-stats` command will result in the statistics being displayed as follows:



NOTE	The resultant statistics displayed shows the proportion and number of tasks and inventory items allocated to each project member in +Work.

4. Contribution to the Developer Guide

The following section displays my additions to the **+Work Developer Guide** for the `Statistics` and `Member` features. They showcase my ability to write technical documentation and the technical depth of my contributions to the project.

4.1. Statistics feature

The Statistics feature allows users to retrieve statistics relating to members and tasks in +Work, so that users can get a broad overview of the project and members' contribution to the project.

4.1.1. Implementation

The commands introduced by this statistics feature includes: `task-stats` and `member-stats`. These commands are facilitated by the class `Statistics` that resides within model. The `Statistics` class implements the following operations:

- `Statistics#doCalculations()` — Calculates the statistics needed using existing list of tasks, members and mappings.
- `Statistics#getPortionMembersByTasks()` — Retrieves statistics of all the members and number of tasks completed by the each individual member.
- `Statistics#getPortionMembersByItems()` — Retrieves statistics of all the members and number of items purchased by the each individual member.
- `Statistics#getPortionTasksByStatus()` — Retrieves statistics of all existing tasks and number of tasks of each status.

These operations are exposed in the `Model` interface as `Model#doCalculations`, and `Model#getStatistics`.

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

Step 1. The user launches the application for the first time. The `Statistics` object stored by ProjectDashboard is initialised based on the data previously saved.

NOTE	Data previously saved refers to the statistics calculation done based on list of members, tasks and mappings saved.

Step 2. The user executes the `task-stats` command to retrieve statistics related to the tasks in the application.

The `task-stats` command obtains lists of all the members, tasks and mappings saved in the application, and uses the lists to form a Statistics object. `Model#setStatistics` is then called to updated the statistics in ProjectDashboard.

The following sequence diagram shows how the `task-stats` operation works.

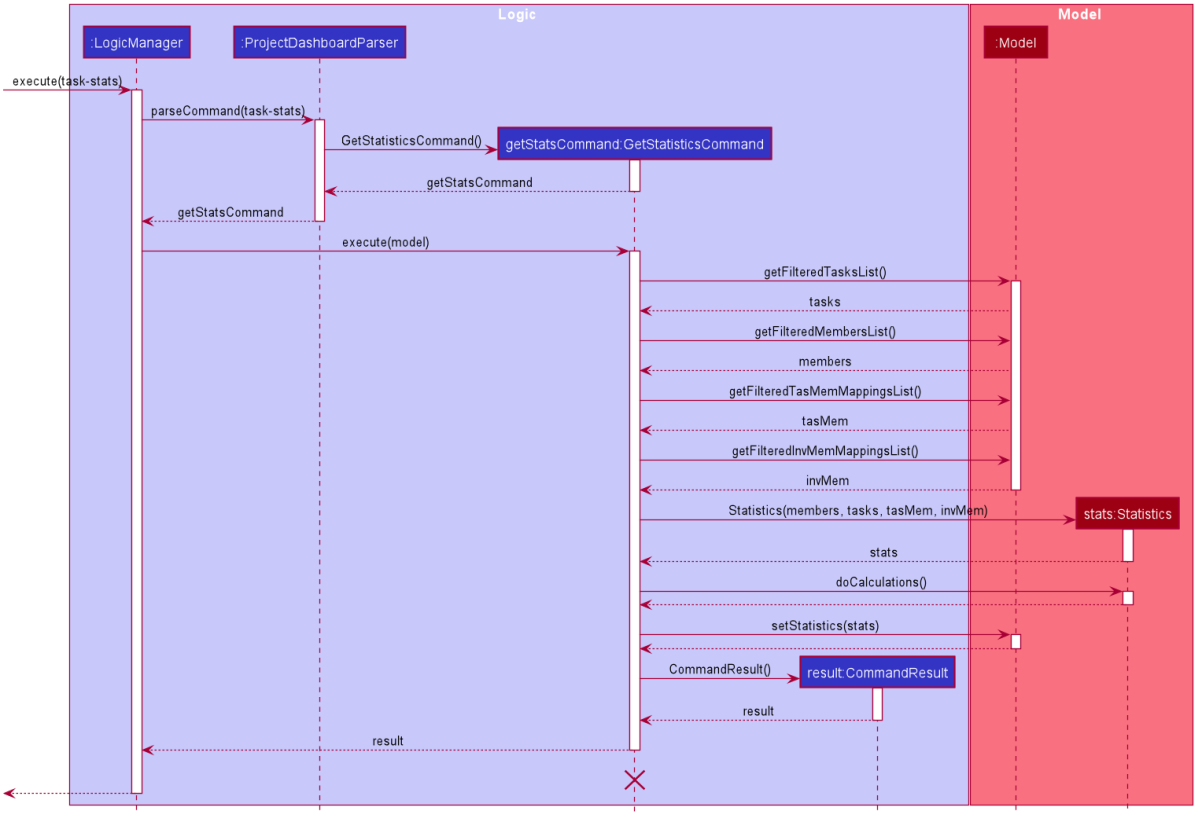


Figure 1. Operational flow of `GetStatisticsCommand`

Step 3. In order for task statistics to be displayed in a comprehensive manner, when the `task-stats` command is called, `TaskStatisticsView` class is also called to display the task stats. To allow the `UI` to be responsive, `getStatistics()` is similarly exposed in the `Logic`interface as `Logic#getStatistics()`.

The following sequence diagram shows how calling the `task-stats` operation leads to the comprehensive UI display of task statistics.

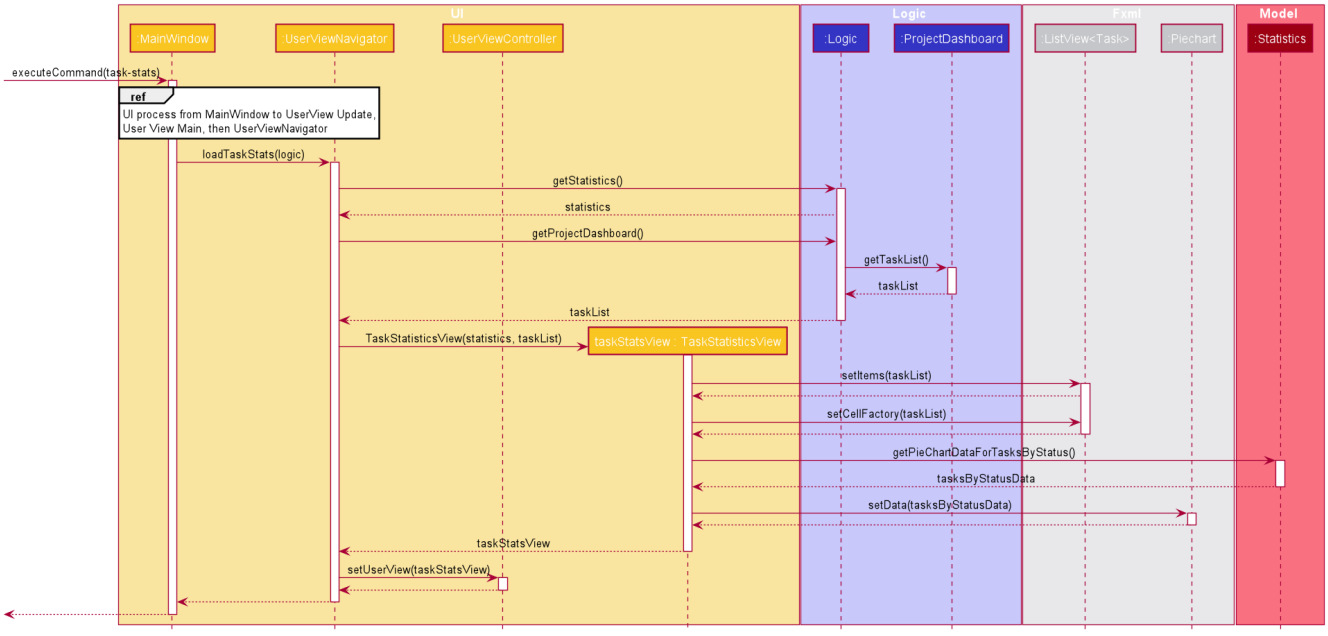


Figure 2. Operational flow of displaying statistics in +Work

4.1.2. Design Considerations

This section describes the pros and cons of the current and other alternative implementations of the Statistics class in +Work, as well as the display of statistics in +Work.

Aspect: Implementation of Statistics class

- **Alternative 1 (current choice):** One statistics object for the entire ProjectDashboard
  - Pros: Easy to implement, centralised class for all statistics
  - Cons: May have performance issues due to calculations involving large amounts of tasks and members.
- **Alternative 2:** Individual statistic objects for members and tasks.
  - Pros: Ensures faster performance, more detailed statistics can be included
  - Cons: Complicates the implementation of the statistics class, might not have enough time to implement it by v1.4

Alternative 1 was chosen given the time constraint in implementing the features in time for +Work Version 1.4.

Aspect: Display of Statistics for Project Dashboard

- **Alternative 1 (current choice):** Use a pie chart to represent information
  - Pros: Increases the ease of workload comparison
  - Cons: Decreases the amount of detail of individual tasks and members that are displayed
- **Alternative 2:** Use a list to represent information
  - Pros: Includes more details for individual elements
  - Cons: Decreases the ease of comparison between tasks and members

Because the team came to a consensus that the main objective of the Statistics feature in +Work is to provide the user with an overview of all the project tasks and members, for ease of comparison, **Alternative 1** was chosen as it fits the purpose more than Alternative 2 does.

4.2. Member Feature

The member feature introduces the ability for +Work to deal with project members, in the same way it deals with project tasks. This makes +Work a more comprehensive application because project tasks and members can be kept track of together.

4.2.1. Implementation

+Work's members and their related commands are supported by a `Member` class that resides within model. The following class diagram exposes the structure of the `Member` class, and shows how the different components relating to the Member class works together.

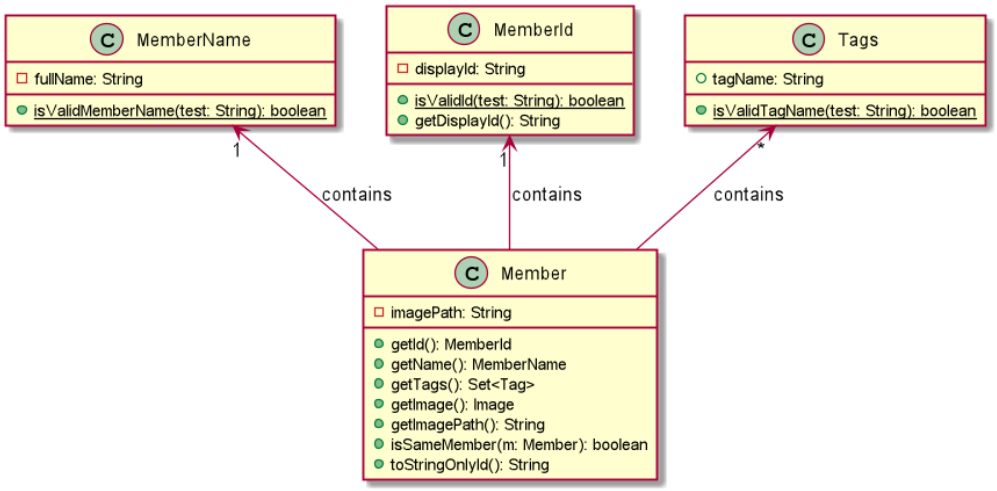


Figure 3. Class diagram of `Member` package

Apart from the typical commands (`add-member`, `delete-member`, `find-member`) involved in such a central class, the member feature also introduces a `set-image` command. The `set-image` command allows users to set an image in their computer as the profile picture of a member in +Work. To accommodate the `set-image` command, the `Member` class has an alternative constructor that takes in the image filepath as a parameter to save it as an attribute to the member object, when `set-image` command is called. Additionally, to support the command, the `Member` class implements the following operation:

- `Member#getImagePath()` — Retrieves the filepath of the image stored in the user's computer
- `Member#getImage()` — Retrieves the member's image using the image filepath

Given below is an example usage scenario and how the set-image mechanism behaves at each step.

Step 1. The user launches the application for the first time, and adds a team member into +Work. The member is displayed with a default profile picture.

Step 2. The user executes the `set-image` command to set an image in their computer as the profile picture of a member in +Work..

The `set-image` command calls `Model#getFilteredMembersList()` to retrieve the Member that is to be edited. A new member object is formed, with all the same parameters as the specified member object, and a new Image Filepath parameter. `Model#setMember` is called to replace the old member object with the new one in +Work.

The following sequence diagram shows how the `set-image` operation works.

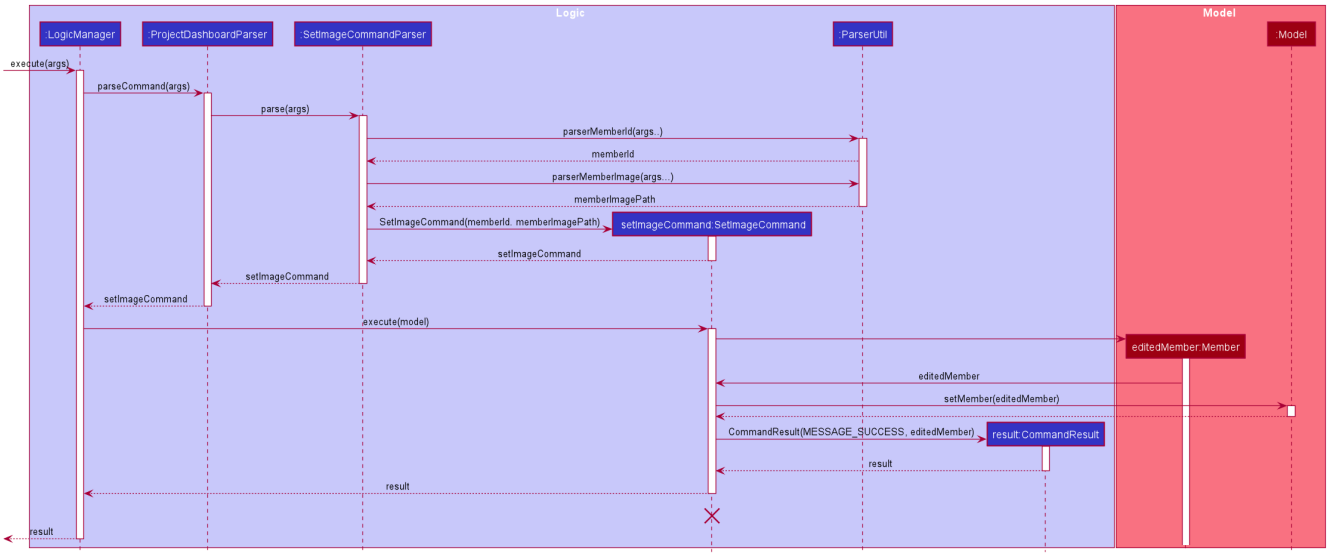


Figure 4. Operational flow of `SetImageCommand`

NOTE	The image's file path is stored in the Member object. If the image is shifted to another location, the file path stored becomes invalid, and the user has to call the `set-image` command again, with the new file path.

4.2.2. Design Considerations

This section describes the pros and cons of the current and other alternative implementations of the image attribute under members.

Aspect: Storage of image under member

- **Alternative 1:** Storing the image filepath as a changeable attribute
  - Pros: Editing a member's profile picture involves accessing the member and changing its file path attribute
  - Cons: The image file path attribute is exposed to the rest of the classes in +Work and may be unintentionally edited, causing the member's profile picture to be edited without the intention to.
- **Alternative 2 (current choice):** Storing the image filepath as a final attribute
  - Pros: Ensures the member's image filepath remains unchangeable and specific to the member
  - Cons: A new member object has to be created to replace the member being edited every time the member's profile picture is updated

Alternative 2 was chosen to keep in line with the original structure of the Person object in AB3, with all attributes being final and unchangeable.