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) application, +Work. It also serves to document the key features I implemented in the application.

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 exiting CLI AddressBook application, to fit the needs of a target group of users.

Our application, +Work, is aproject management tool aimed at NUS project leaders who are in charge of a small project group. It allows project leaders to successfully achieve the objectives of their project, by helping them manage their members, meeting times, task allocations as well as personal claims for equipment purchased. A user can use +Work to find the most popular meeting time among his project members, when calling for a meeting. He can add project tasks, and members that are assigned under these tasks, into +Work to better manage the project manpower. Additionally, he can add inventory items into +Work to keep track of the purchases made over the course of the project.

1.3. Formatting of Document

Show down below are symbols that appear in my Project Portfolio, and their significance.

NOTE	Information here exposes a small bit of +Work’s inner workings so that the user understands how a command can affect +Work.
TIP	Information here helps users use +Work more efficiently, but is not necessary in using +Work in the most basic sense.
IMPORTANT	Information listed in under this note is essential information that users should take note of, in order for +Work to function as stated.

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. For project members, the number of tasks and inventory items allocated to each member is retrieved. For project tasks, the proportion of tasks that are undone, in progress and completed, as well as the time spent on each task inputted into +Work, is retrieved.
  - Justification: This feature improves the product significantly because it gives the user a broad overview of the project’s progress and members' responsibility allocation at a glance. This makes it easier for the user to plan out future tasks and allocate them accordingly to members so that everyone has close to equal responsibility, which is one of the main goals of effective project management.
  - Highlights: This enhancement affects existing commands and commands to be added in future. It required an in-depth analysis of design alternatives. The implementation too was challenging as it required changes to existing commands.
- **Major enhancement**: I added **the ability for users to work with (add, delete, list) project members on +Work**
  - What it does: This feature allows the user to retrieve statistics relating to tasks and members in +Work. For project members, the number of tasks and inventory items allocated to each member is retrieved. For project tasks, the proportion of tasks that are undone, in progress and completed, as well as the time spent on each task inputted into +Work, is retrieved.
  - Justification: This feature improves the product significantly because it gives the user a

  broad overview of the project’s progress and members' responsibility allocation at a glance. This makes it easier for the user to plan out future tasks and allocate them accordingly to members so that everyone has close to equal responsibility, which is one of the main goals of effective project management.

  - Highlights: This enhancement affects existing commands and commands to be added in future. It required an in-depth analysis of design alternatives. The implementation too was challenging as it required changes to existing commands.

- **Major enhancement**: I added **the ability for users to add profile pictures for each project member in +Work**
  - What it does: This feature allows the user to retrieve statistics relating to tasks and members in +Work. For project members, the number of tasks and inventory items allocated to each member is retrieved. For project tasks, the proportion of tasks that are undone, in progress and completed, as well as the time spent on each task inputted into +Work, is retrieved.
  - Justification: This feature improves the product significantly because it gives the user a broad overview of the project’s progress and members' responsibility allocation at a glance. This makes it easier for the user to plan out future tasks and allocate them accordingly to members so that everyone has close to equal responsibility, which is one of the main goals of effective project management.
  - Highlights: This enhancement affects existing commands and commands to be added in future. It required an in-depth analysis of design alternatives. The implementation too was challenging as it required changes to existing commands.
- **Minor enhancement**:
  - Enhanced the UI of Members to include the uploading of members' profile photos, as well as listing of tasks allocated to each member under individual member cards.
  - Enhanced the UI of statistics calculated in easy-to-read format.
- **Code contributed**: [Functional code] [Test code] *{give links to collated code files}*

2.2. Other Significant Contribution to Project

- Project management
  - Managed releases `v1.3` - `v1.5rc` (3 releases) on GitHub
- Enhancements to existing features
  - Updated the GUI color scheme (Pull requests #33, #34)
  - Wrote additional tests for existing features to increase coverage from 88% to 92% (Pull requests #36, #38)
- Documentation
  - Did cosmetic tweaks to existing contents of the User Guide: #14
- Community
  - Reviewed PRs (with non-trivial review comments): #12, #32, #19, #42
  - Contributed to forum discussions (examples: 1, 2, 3, 4)

    - Reported bugs and suggestions for other teams in the class (examples: 1, 2, 3)
    - Some parts of the history feature I added was adopted by several other class mates (1, 2)
- Tools
    - Integrated a new Github plugin (CircleCI) to the team repo

3. Contributions to the User Guide

3.1. Current Enhancement

3.2. Proposed Enhancement for Version 2.0

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

3.3. Universal Commands

3.3.1. Accessing project dashboard page: `home`

This command brings you to the project dashboard page, where tasks are displayed.

Format: `home`

Calling the `home` command will bring you to the following page:

[Home] | *Home.png*

3.3.2. Accessing time management page: `calendar`

This command brings you to the time management page where calendar and meeting times are displayed

Format: `calendar`

Calling the `calendar` command will bring you to the following page:

3.3.3. Accessing settings page: `settings` (Coming in v1.3)

This command brings you to the settings page

Format: `settings`

Calling the `settings` command will bring you to the following page:

[Settings] | *Settings.png*

NOTE You can refer to section 3.7 on specific settings-related commands to apply.

3.3.4. Viewing help: `help`

Displays a list of possible commands for the user
You can toggle through the command list (either through up down keys or mouse) and it will paste the correct syntax into the command line.
Format: `help`

[Help] | *Help.png*

3.4. Member-related Commands

3.4.1. 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 below.

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.

[BeforeAdd] | *BeforeAdd.png*

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.

[DuringAdd] | *DuringAdd.png*

Step 3: After you hit kbd:[Enter], a new member with name 'New Member', member ID 'NM' and tag 'UG' is added to +Work.

[DoneAdd] | *DoneAdd.png*

The addition of a new member can also be seen from the list of members down below:

[ListAdd] | *ListAdd.png*

NOTE 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/…`

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

To set a profile picture for a member in +Work, use the command `set-image` following the format below.

Format: `set-image mi/MEMBER_ID im/IMAGE_PATH`

Example: `set-image mi/NM im/C:\Users\Lynn\Desktop\Y2S1\CS2103T\tP\NewUserImage.png` can be executed as follows:

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

[BeforeSet] | *BeforeSet.png*

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:\Users\Lynn\Desktop\Y2S1\CS2103T\tP\NewUserImage.png` into the command prompt box.

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

[SetImage] | *SetImage.png*

NOTE	Image Path refers to the folder path of the image stored in your computer, and should end with .png

NOTE	If you shift the image’s location in your computer, +Work will be unable to find the image to display. It is recommended that you store all the images in a central folder to prevent this from happening.

3.5. Editing a member : `edit-member`

To edit a member in +Work, using the command `edit-member` following the format below.

Format: `edit-member mi/MEMBER_ID mn/MEMBER_NAME mt/MEMBER_TAG`

- Edits the member at the specified `mi/MEMBER_ID`.
- In this command, all the fields apart from `mi/MEMBER_ID` is optional. However, at least one of the optional fields must be provided.
- Existing values will be updated to the input values.
- When editing tags, the existing tags of the member will be removed i.e adding of tags is not cumulative.

Example: `edit-member mi/NM mn/No Longer New mt/edited` can be executed as follows:

Step 1: +Work initially contains a list of 7 project members, as shown below.

[BeforeEdit] | *BeforeEdit.png*

Step 2: To edit the member name and tag of 'New Member' with member ID 'NM', you enter the command `edit-member mi/NM mn/No Longer New mt/edited` into the command prompt box.

Step 3: After you kbd:[Enter] the command, the member with member ID 'NM' and tag 'UG' is edited, with a new member name 'No Longer New' and new tag 'edited'.

[AfterEdit] | *AfterEdit.png*

3.5.1. List existing members: `list-members`

To get a list of all members added to +Work, used the command `list-members` following the format below.
Format: `list-members`

Example: Entering `list-members` into the command prompt will result in the following:

[ListMembers] | *ListMembers.png*

As seen from the above, all existing project members in +Work will be listed.

3.5.2. Removing a member: `remove-member`

To remove a member from the project, and subsequently remove him from associated tasks, use the 'remove-member' command in the format below.
Format: `remove-member [mi/MEMBER_ID]`

Example: `remove-member mi/GS` can be executed as follows:

Step 1: +Work now contains a list of 7 project members, as shown below.

[BeforeRemove] | *BeforeRemove.png*

Step 2: To remove project member 'No Longer New', with member ID 'NM' from +Work, you enter the command `remove-member mi/NM` into the command prompt box.

Step 3: After you kbd:[Enter] the command, the member 'No Longer New' is no longer a project member in +Work, as seen from the list of members below.

[AfterRemove] | *AfterRemove.png*

3.5.3. Assign a task to a member: `assign`

To assign a task to a specific team member, use the `assign` command in the format below.

Format: `assign ti/TASK_ID mi/MEMBER_ID`

Example: `assign ti/1 mi/GS` can be executed as follows:

Step 1: From the list of tasks shown below, you decide to assign the task 'Review Budget' to project member 'Gabriel Seow' with member ID 'GS'. The task 'Review Budget' has task ID 1, prompting you to enter the command `assign ti/1 mi/GS`.

[BeforeAssign] | *BeforeAssign.png*

Step 2: After you kbd:[Enter] the command, the task 'Review Budget' with task ID '1' is added under member 'Gabriel Seow' with member id 'GS', as seen from the image below.

[AfterAssign] | *AfterAssign.png*

3.5.4. Removing a task from a member: `fire`

To remove a task from a specific team member, use the 'fire' command in the format below.

Format: `fire ti/TASK_ID mi/MEMBER_ID`

Example: `fire ti/1 mi/GS` can be executed as follows:

Step 1: From the list of tasks shown below, you decide to remove project member 'Gabriel Seow' with member ID 'GS' from being assigned to task 'Review Budget'. The task 'Review Budget' has task ID 1, prompting you to enter the command `fire ti/1 mi/GS`.

[BeforeFire] | *BeforeFire.png*

Step 2: After you kbd:[Enter] the command, the task 'Review Budget' with task id 1 is removed from member 'Gabriel Seow' with member id 'GS' as seen from the image below.

[AfterFireMember] | *AfterFireMember.png*

---

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

4.1.1. Implementation

The commands introduced by this statistics feature includes: `get-task-stats` and `get-member-stats`. These commands are faciliatated 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'.

NOTE To allow the `UI` to be responsive, one of the operations is similarly exposed in the `Logic` interface as `Logic#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' will be 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 `get-task-stats` command to retrieve statistics related to the tasks in the application.

The `get-task-stats` command calls `Model#getFilteredTasksList()`, `Model#getFilteredMembersList()` and `Model#getFilteredMappingsList()` after updating all the lists to show all tasks/members/mappings, to obtain lists of all the members, tasks and mappings saved in the application. Using the lists, a Statistics object is formed. 'Model#setStatistics' is called to updated the statistics in ProjectDashboard.

The following sequence diagram shows how the 'get-member-stats' operation works.

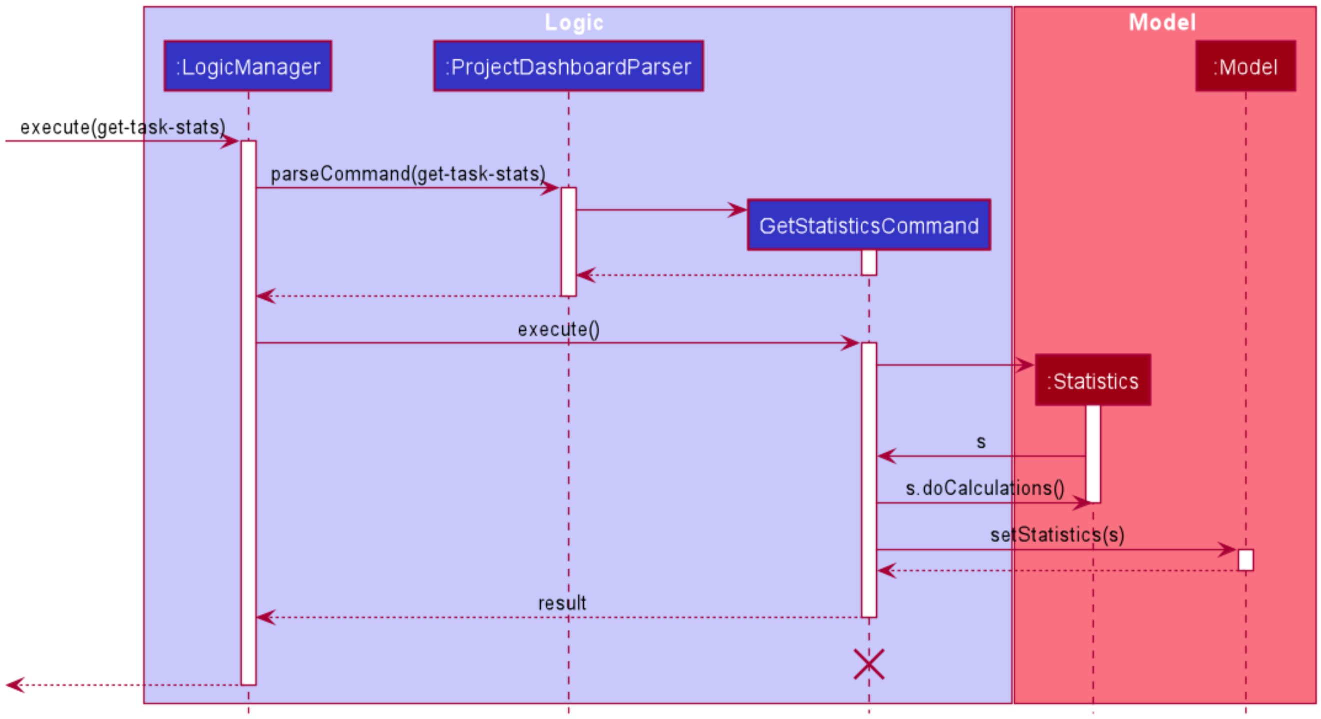
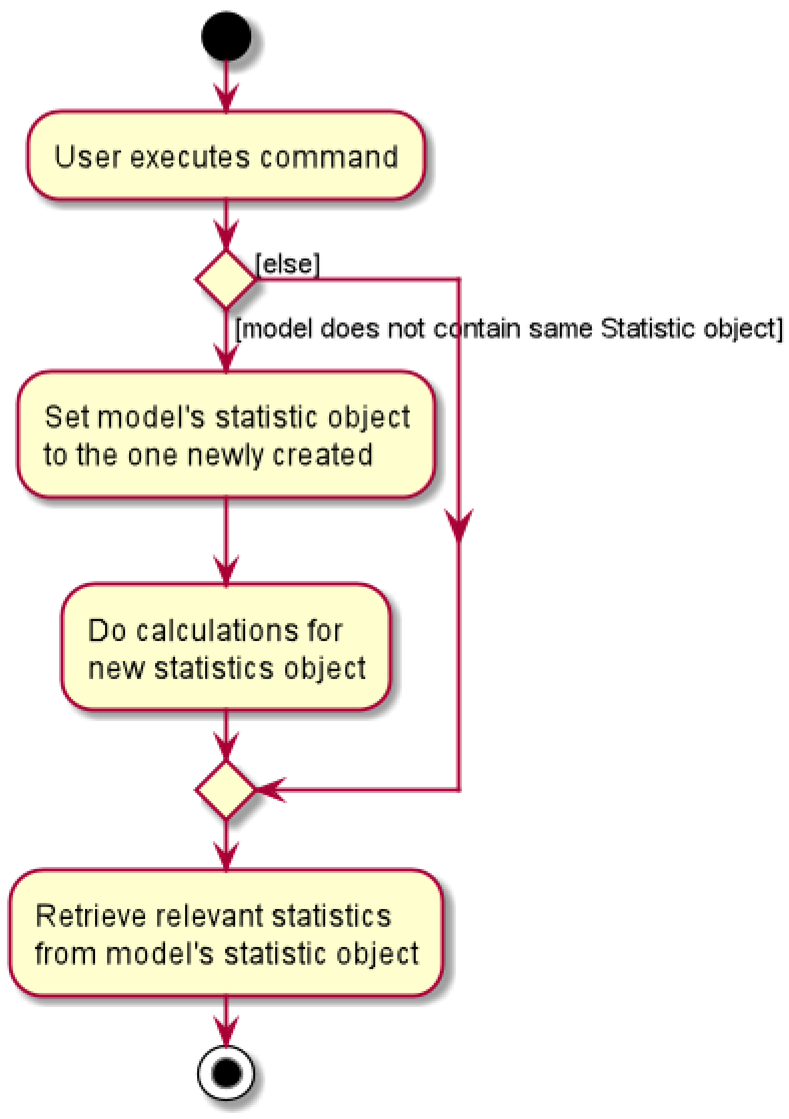


Figure 10. Operational flow of 'GetStatisticsCommand'

NOTE The 'get-member-stats' operation is similar to the one shown in figure 10.

The following activity diagram summarizes what happens when a user executes a new command:



User executes command
[else]
[model does not contain same Statistic object]
Set model's statistic object to the one newly created
Do calculations for new statistics object
Retrieve relevant statistics from model's statistic object

4.1.2. Design Considerations

Aspect: How to implement 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: Will ensure faster performance,
  - Cons: Design of statistics object has to be very well-implemented, might not have enough time to implement it by v1.3

Aspect: Display of Statistics for Project Dashboard

- **Alternative 1 (current choice):** Use a pie chart to represent information
    - Pros: Easy for workload comparison
    - Cons: Less details of individual taks and members exposed in the statistiscs page.
- **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

4.2. Member Feature

4.2.1. Implementation

The member feature introduces the ability for +Work to deal with project members, in the same way it deals with project tasks. Apart from the typical commands ('add-member', 'delete-member', 'find-member') involved in such a central class, the member features also introduce 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 accomodate the set-image command, the 'Member' class has an alternative constructor that takes in the image url as a parameter to save it as an attribute to the member object, when 'set-image' command is called. Additionally, to support the 'set-image' command, the 'Member' class implements the following operation:

- `Member#getImageUrl()` — Retrieves the URL of the image stored in the user’s computer
- `Member#getImage()` — Retrieves the member’s image using the image url

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 `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 retrive 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 URL parameter. `Model#setMember` to replace the old member object with the new one in +Work.

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

[SetImageSequenceDiagram] | *SetImageSequenceDiagram.png*

Figure 10. Operational flow of 'SetImageCommand'

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

Step 3. When an operation is called to display a member, `Member#getImage` is called to display the image using Javafx's ImageView.

The following sequence diagram shows how the image is called up and subsequently displayed in the UI for members.

[DisplayImage] | *DisplayImage.png*

Figure 10. Operational flow of displaying a member with image.

The following activity diagram summarizes what happens when a user executes a new command:

[SetImageActivityDiagram] | *SetImageActivityDiagram.png*

4.2.2. Design Considerations

Aspect: How to save image under Member (url or image object)

- **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: Will ensure faster performance,
  - Cons: Design of statistics object has to be very well-implemented, might not have enough time to implement it by v1.3

Aspect: Store image as (final or changeable) (basically repalcing the whole member or not)

- **Alternative 1 (current choice):** Use a pie chart to represent information
  - Pros: Easy for workload comparison
  - Cons: Less details of individual taks and members exposed in the statistiscs page.
- **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

Aspect: Ui Display of members

- **Alternative 1 (current choice):** Use a pie chart to represent information
  - Pros: Easy for workload comparison
  - Cons: Less details of individual taks and members exposed in the statistiscs page.
- **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