Yap Jia Aun - Project Portfolio for Alfred

About the project

My team and I were tasked with enhancing a command line interface addressbook application. After careful consideration, we morphed the application into a Hackathon personnel management system called Alfred. With human resource managers of startups and small- and medium- sized enterprise in mind, we created this application to help streamline the management of Hackathon competitions. This improved application enables users to manage different parties(like participants, teams and mentors). Additionally, it automatically establishes relationships between these parties, delegating the tedious task of maintaining complex relationships to the system itself.

This is what our project looks like:

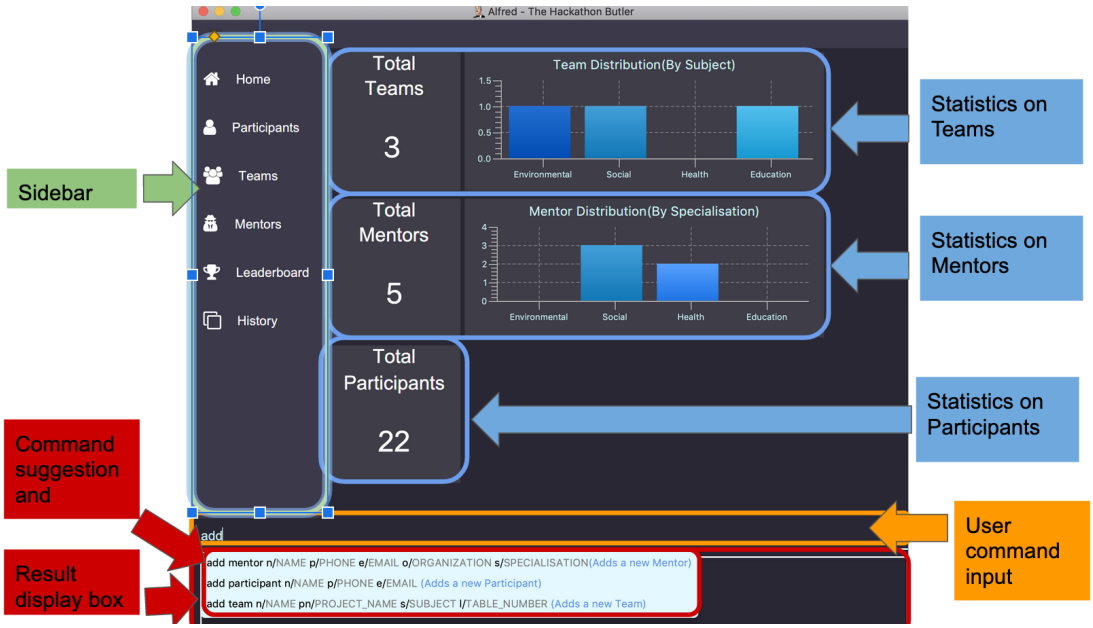


Figure 1. The graphical user interface for the homepage of Alfred.

My main role was to design and write the code for the Graphical User Interface(GUI), to ensure that information about each party is displayed in an organised and readable manner. As part of my duty, I have implemented the command suggestion prompt. I have also implemented assign and remove commands to assign participants or mentors to a team. These are the two main feature I have implemented, The following sections illustrate these enhancements in further detail, including the relevant documentation I have included in the user and developer guide with reference to these features. Further details on the `home` command feature and layout of the GUI (that I have also implemented) can be found in the user guide and developer guide.

Note the following terms used in this document:

Entity: Entity refers to different parties that could be stored in the system. This includes participants, teams and mentors that are involved in the Hackathon competition.

Note the following symbols and formatting used in this document:

NOTE	This symbol indicates important information.

`remove mentor` 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.

`Team` An underlined text with grey highlight indicates a component, class or object in the architecture of the application. Clicking these text will redirect user to the implementation or interface of the component, class or object.

Summary of contributions

This section shows a summary of my coding, documentation, and other contributions to the project.

Enhancement added: I added the ability to choose from different suggested command templates as a user types into the user input box

- What it does: This feature predicts the command that a user is going to enter as they are typing, and provides suggestions. The user can then navigate and choose from these suggestions. Doing so will provide a command template for the user to work it.
- Justification: Managing different entities in the system requires a variety of commands(in which some are more complex in nature). Thus, this feature allows users to conveniently summon commands. Furthermore, it takes the memory work of remembering numerous commands out of the equation.
- Highlights: This enhancement works with existing as well as future commands. Addition of new commands would only require minimal changes to the code base of this feature. The fulfilment of this feature was more demanding as it requires a deeper understanding on the working of the third party library JavaFX. On top of that, its implementation was constantly met with errors, before vital concepts regarding the Node components of JavaFX came into light.
- Credits: This was inspired by the command suggestion box in Telegram bots. The inspiration of how it can be implemented comes from the following webpage: https://stackoverflow.com/questions/50495430/how-to-customize-auto-complete-text-field-suggestion-in-javafx

Enhancement added: I added the ability to assign participants or a mentor to a team

- What it does: The assign participant and assign mentor command assigns a participant or mentor to a team. The remove participant or remove mentor command removes said entity from the team.
- Justification: Hackathons involves teaming participants up and allocating a mentor to guide teams. This feature helps to establish this relation.

Code contributed: Please click these links to see a sample of my code: Functional code | Test code

Other contributions:

- Project management:

- There were a total of 5 releases, from version 1.1 to 1.5. I managed releases versions 1.2.1 on GitHub.

- Enhancements to existing features:
  - Wrote additional tests for existing features to increase coverage (Pull requests #347)
  - Added and modified constraints of different fields in the commands(Pull requests #15)
  - Implemented sidebar and dark mode for readability and aesthetic purposes (Pull requests #158,https://github.com/AY1920S1-CS2103T-F11-1/main/pull/284[#284])
- Documentation:
  - Added user stories and use cases to Developer Guide(Pull requests #149, #153, #215, #293)
- Community:
  - Reviewed Pull Requests (with non-trivial review comments): #200, #104 #173 #127
  - Integrated a third party library (FontAwesomeFX and JFeonix) to the project (#159)

Contributions to the User Guide

We had to amend the Addressbook User Guide with instructions on how to utilise the enhancement we added. Below is an excerpt of the Alfred User Guide, showcasing the homepage statistics, command suggestion prompt and assign command features I have added.

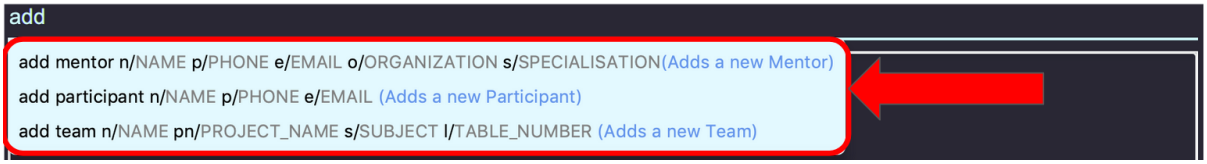
Command Suggestion Prompt

As you type, there will be a popup box predicting the type of commands you are going to type. You can navigate through these suggestions and choose the template that suits you.

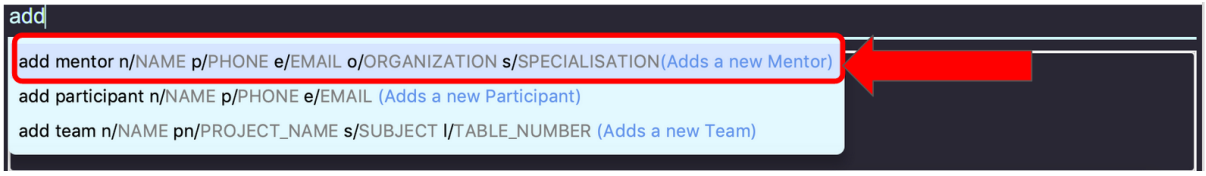
Example: Let us suppose that you want to type the command `add mentor` and you forgot the fields that are required in the command.

Instead of going through the user guide to look for the command, a popup box will appear as you type. The content in this box will change as you type, such that the commands suggested will start with the words or letters that you have already entered.

As you type: 1. Type ‘add’ into the command box, and `add participant`, `add mentor` and `add team` commands will be suggested to you. The grey text are meant as guides and blue text are meant as usage instructions . These text will not appear if you choose the command.

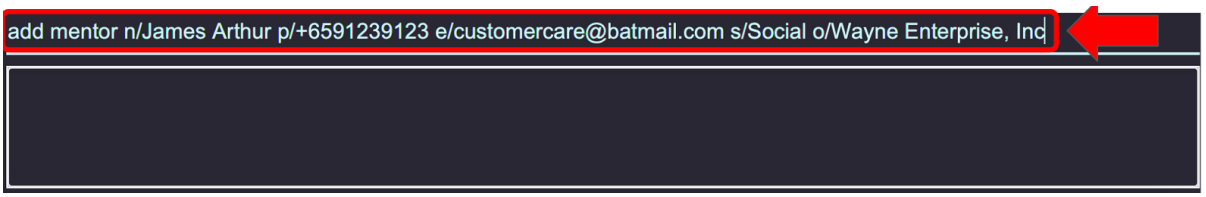


1. Press the `up or `down` arrow keys to navigate up and down the popup box.



1. Press `Enter` to choose the command of your choice. The command will then appear on the user input box.

1. Press 'left' or 'right' keys to navigate the cursor and fill in the respective fields. Press Enter to execute the command.



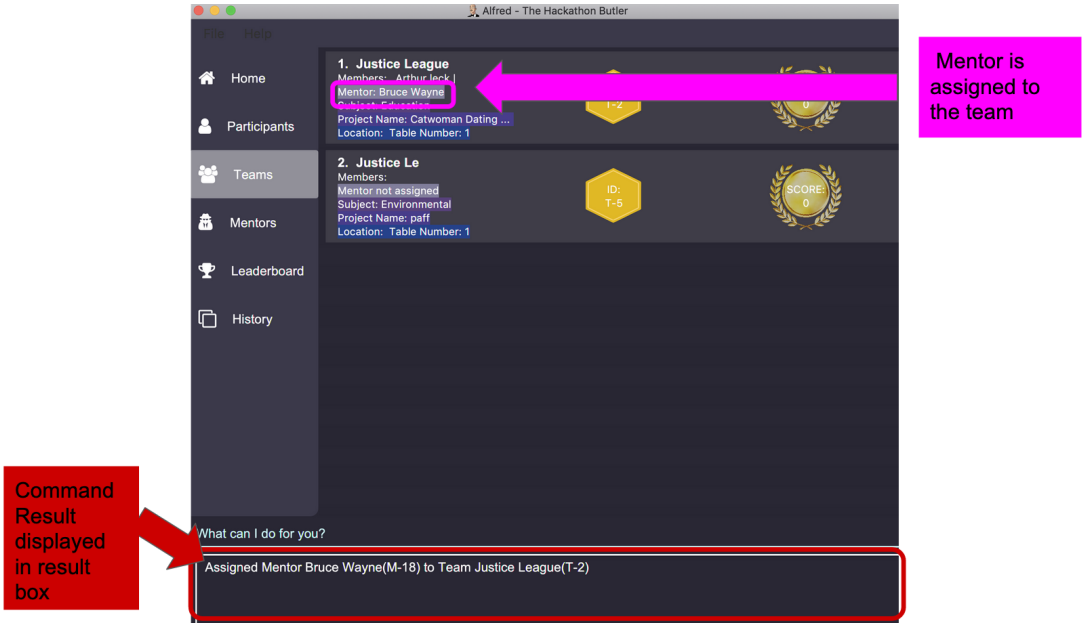
Assigning a Entity to a Team: `assign {mentor/participant} ID TEAM_ID …`

Assigns Mentor or Participant Entity by their ID to a team identified by TEAM_ID.

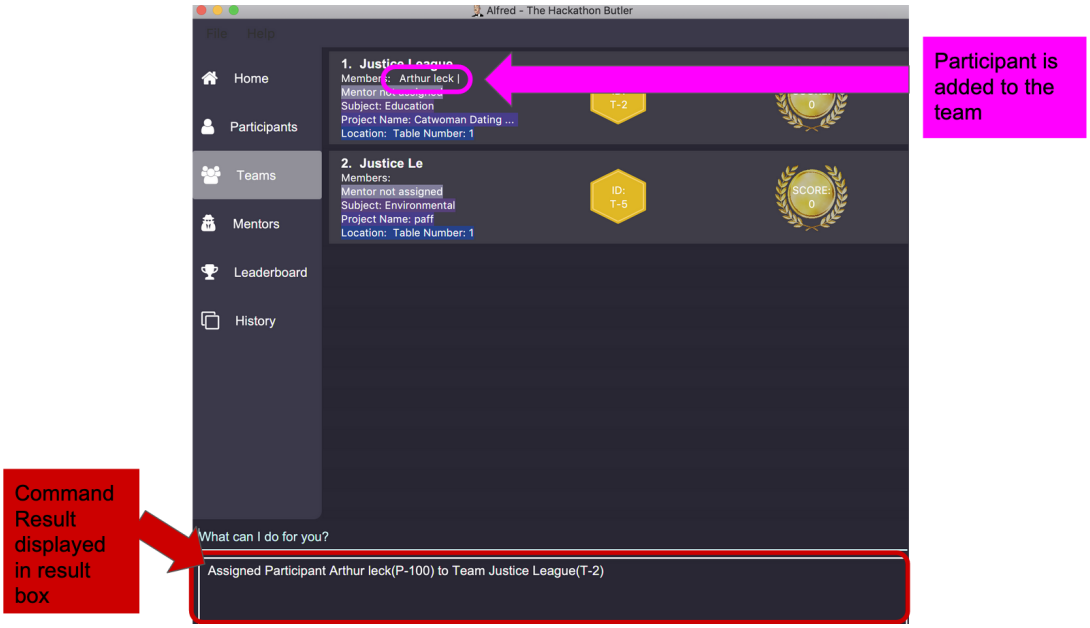
> - Take note that the `assign` command can only be used to assign Participants and Mentors to a Team.
> - An error will be shown is the Team already has a Mentor.
> - An error will be shown if the Team already has said Participant.

Examples:

- `assign mentor M-18 T-2` will assign Mentor with ID M-18 to Team with ID T-2. Running the command will show you the following output in the 'Team' section of the GUI:



- `assign participant P-100 T-2` will assign Participant with ID P-100 to Team with ID T-2. Running the command will show you the following output in the 'Team' section of the GUI:



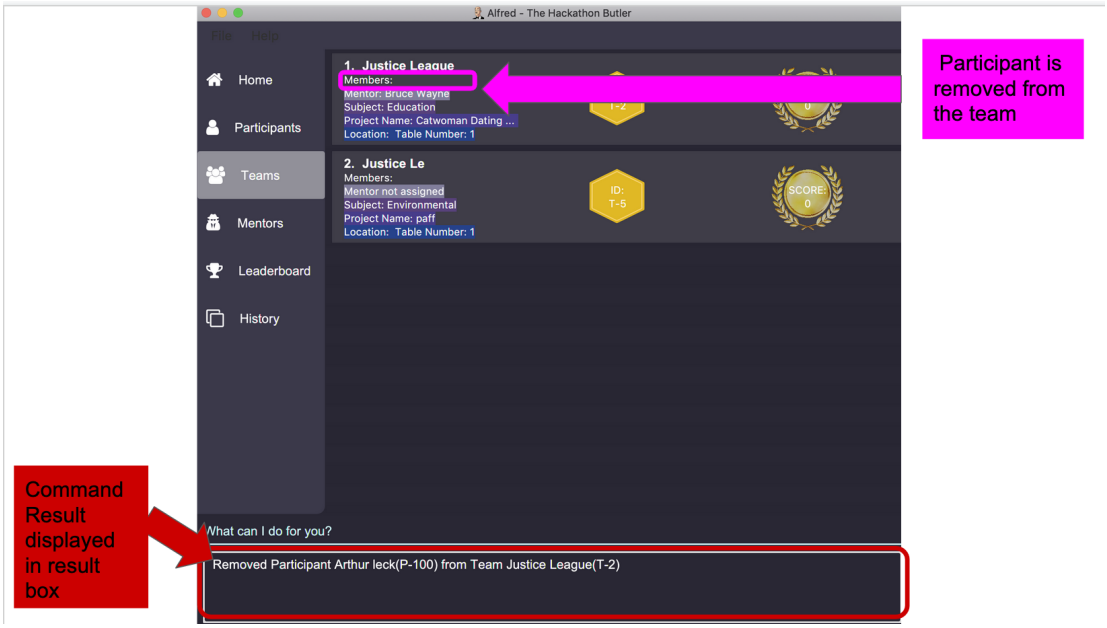
Removing an Entity from a Team: `remove {mentor/participant} ID TEAM_ID ...`

Removes Mentor or Participant Entity by their ID from a team identified by TEAM_ID.

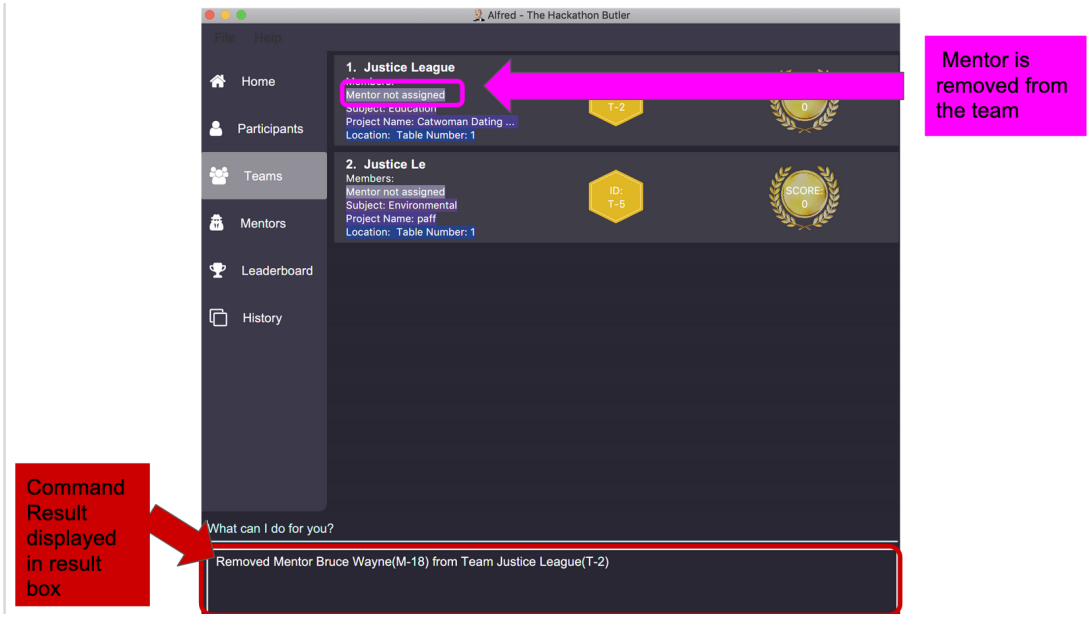
- Take note that the `remove` command can only be used to remove Participant or Mentor from a Team.
- An error will be shown if the Participant or Mentor is not in the specific Team.
- Deleting a Participant or Mentor will also delete all their connections with a team or teams.

Examples:

- `remove mentor M-18 T-8` will remove Mentor with ID M-18 to Team with ID T-8. Running the command will show you the text 'Mentor not assigned' in the respective team. The following will be shown in the 'Team' section of the GUI:



- `remove participant P-100 T-2` will remove Participant with ID P-100 from Team with ID T-2.Running the command will show you the following output in the 'Team' section of the GUI



Contributions to the Developer Guide

The following sections shows my contribution to the Alfred Developer Guide for the homepage statistics feature, command suggestion and assign feature.

Command suggestion feature

This feature provides suggestions by predicting the commands that a user intends to enter(as the user types).

NOTE	Only suggestions that start with the same alphabets or spaces as those entered by user will be suggested.

Implementation

The main class responsible for remembering and providing the previously used command input strings is the `AutoCompleteCommandBox` class.Typing into the `AutoCompleteCommandBox` will prompt the attached `Listener` to be activated. Activation of the `Listener` will prompt it to filter through the set of predefined command suggestions. The commands that start with the same text entered by user will be filtered through. The first five result will then be mapped to their respective `TextFlow` object and added to the `ContextMenu`. This `ContextMenu` will then appear as a pop up box. ser input box, the method. Pressing `up` and `down` arrow keys will enable the user to choose a command suggestion. Additionally, pressing `enter` will filter out different `Text` from the `TextFlow` object. The `setText` method will then be called to set the `JFXTextField` to the said `Text` object.

`assign` / `remove` feature

The Class Diagram below showing the high level representation of the Object Oriented solution devised to implement the `assign` feature.

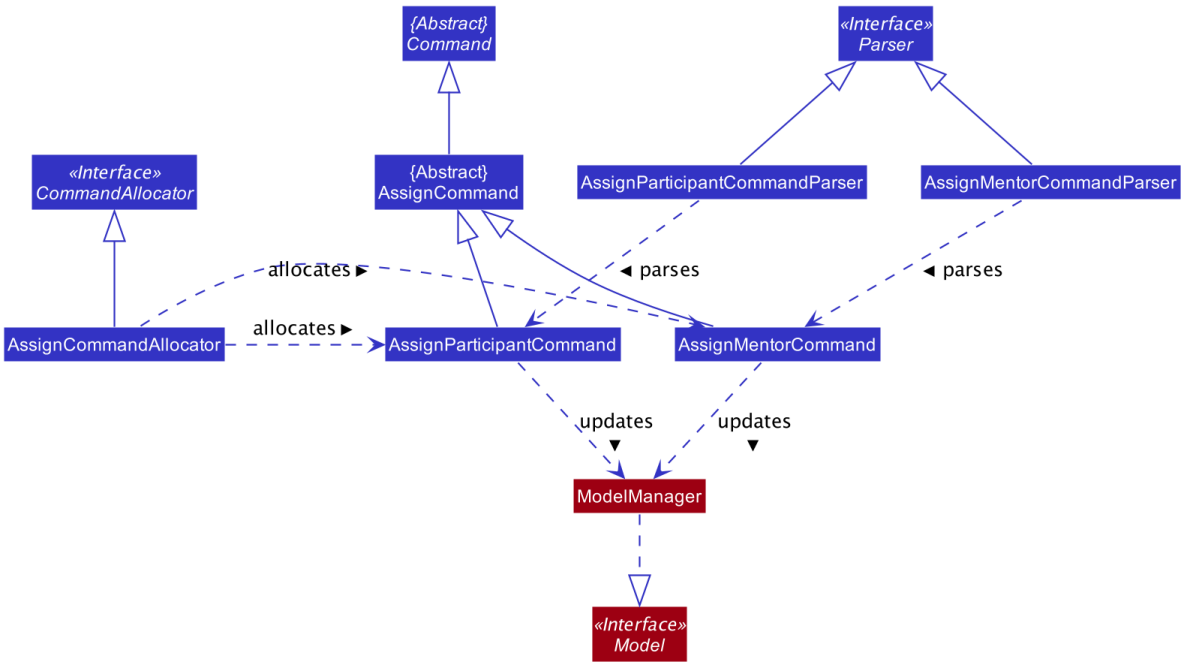


Figure 2. Assign Class Diagram

The Class Diagram below showing the high level representation of the Object Oriented solution devised to implement the `remove` feature.

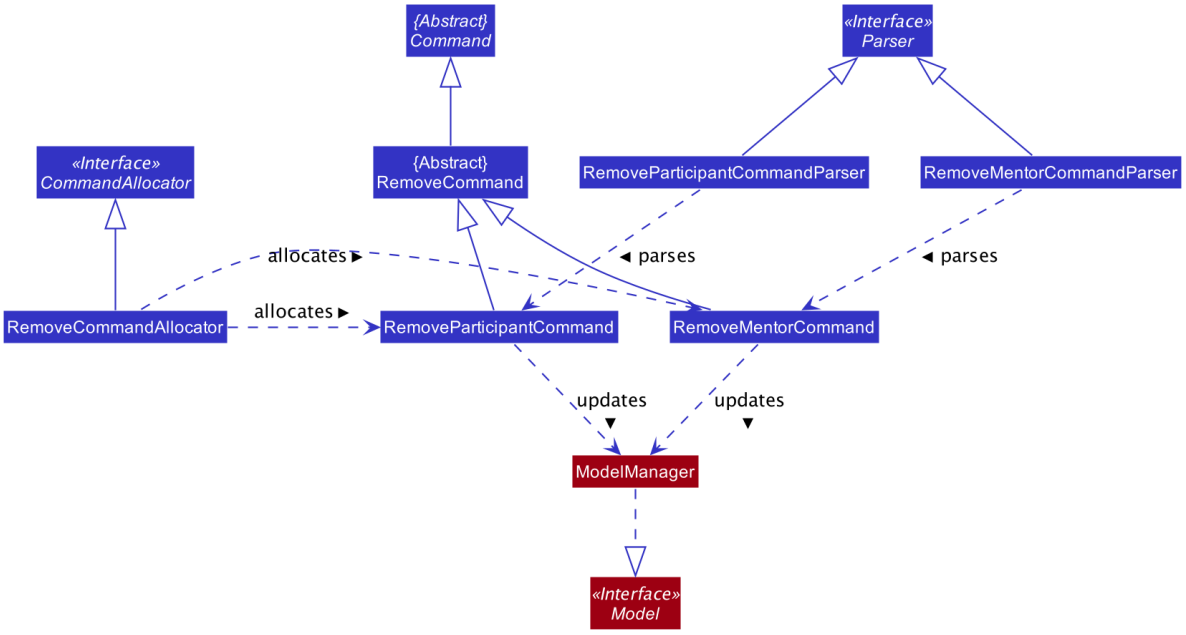


Figure 3. Remove Class Diagram

Upon successful assignation, the new participant or mentor will be stored internally in the list of participant or optional mentor field in the Team object. Upon successful removal of Participant or Team, the specified participant or mentor will be removed from the Team object. Additionally, it calls the following operations:

- `ModelManager#addMentorToTeam` – adds mentor to a specified team
- `ModelManager#addParticipantToTeam` – adds participant to team
- `ModelManager#removeParticipantFromTeam` – removes participant from team
- `ModelManager#removeMentorFromTeam` – removes mentor from team

`assign` feature

1. The `assign participant` command will add the new participant under the list of participant in the specified `Team` object. This is provided that the number of members in the `Team` object(size of list of participants) is less than 5.

The following sequence diagram shows how the `assign participant` operation works:

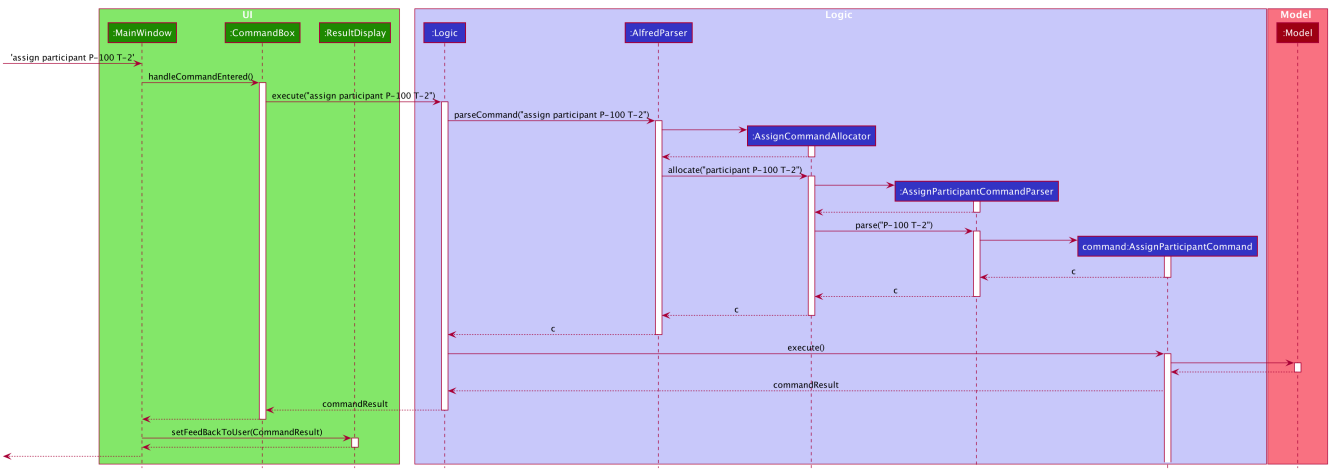


Figure 4. Sequence Diagram for an example of `assign participant` command

1. The `assign mentor` command will add the new mentor under the `Optional<Mentor>` field in the specified `Team` object. This is provided that there is no existing mentor in the team. The sequence diagram of `assign mentor` is similar to that of `assign participant`.

`remove` feature

1. The remove participant will first search through the list of participant under the specified `Team` object. This checks whether the specified participant is a member of the team in the first place. If it is not a member, an error will be thrown. Whereas if it is a member, the specified participant will be removed from the list of participant.

The following sequence diagram shows how the `remove participant` operation works:

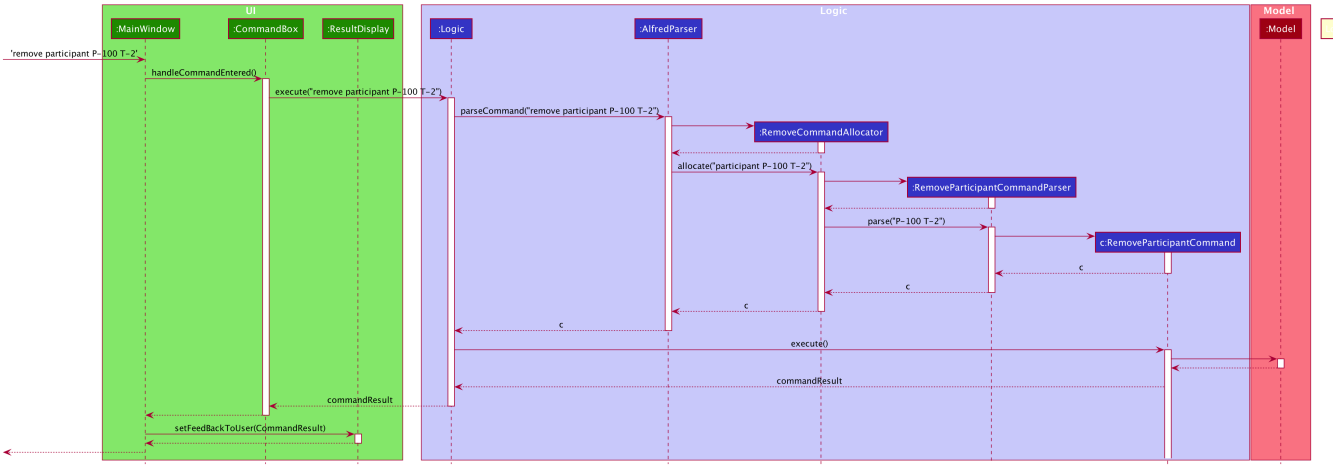


Figure 5. Sequence Diagram for an example of `remove participant` command

1. The `remove mentor` command will first check whether the `Optional<Mentor>` field under the specified `Team` object is not empty and corresponds to the specified mentor. This checks whether the team have not been assigned a mentor, or they have been assigned to a different mentor. Under any of these cases, and error will be thrown. Whereas if the team is assigned the specific mentor, the specified mentor will be removed from the `Optional<Mentor>` field.The sequence diagram of `remove mentor` is similar to that of `remove participant`.

Design Considerations

When designing the GUI, homepage statistics, command suggestion and assign/remove functions, careful deliberation was needed on optimum data structure to use and design principle to employ to implement these features. Below is a brief summary of the thought process I put in before coming to a decision.

Aspect	Alternative 1	Alternative 2

How to store a set of correct commands and search through it as user types: In order to provide suggestions as a user types, there needs to be a way to store the set of correct commands as strings in the system, and search through it to check if it contains some parts of user input. The results will be provided to the user as suggestions.	Use an `ArrayList` and go through every element in the array list one by one to look for commands that matches the user input. **Pros**: Easy to implement **Pros**: When new commands are implemented, they can easily be added into the `ArrayList` **Cons**: Allows duplicates, if duplicate commands are added into the `ArrayList`, duplicate suggestions may be provided. **Cons**: Searching through the `ArrayList` is inefficient as `Stream` is not used.	Store the commands in a `Set` and convert it to a `Stream` to search through the commands. **Pros**: Prevent duplicates, so that duplicate commands will not be entered accidentally. **Pros**: More efficient. **Pros**: Easier to search through the set of commands when it is stored as `Stream`. I have decided to proceed with this option as preventing duplicates enables a more defensive style of programming. It is also more efficient.
How to fill up user input text field when a command suggestion is chosen: upon choosing a command suggestion, it template(the command suggestion excluding blue-colored instructions and grey-colored guides) will be used to occupy the user input text field.	Map each command suggestion to their respective templates in `String` form **Pros**: Easy to implement. **Cons**: Tedious to implement(requires many lines of code). **Cons**: Duplicate logic will be implemented, as the templates could be extracted from the command suggestion itself.	Filter out the relevant text template from the command suggestion that is choosen. **Pros**: No duplicate logic is implemented as the templates are extracted directly from the choosen command suggestion. **Pros**: Lesser code needs to be written in order to extract the template from the choosen command suggestion. **Pros**: Use of regular expression may be harder to implement.