Kerwin Lim - Project Portfolio

Introduction

[github] [linkedin]

Hello! My name is Kerwin Lim. I'm a Computer Science Major at the National University of Singapore (NUS). I'm currently in my second year of studies.

This portfolio page aims to document the contributions I have made in the development of TutorAid. TutorAid is a project that my team and I completed for our CS2103T module. We have put in our heart and soul creating TutorAid. It has been a tremendous learning journey for my team and I and we hope that users will enjoy and benefit from our product!

PROJECT: TutorAid

Overview

TutorAid is a useful application created for Tutors, by Tutors. The purpose of our application is to be a all encompassing solution to store information such as notes and reminders. Our product is made for tutors and teaching assistants in NUS. However, we have plans to expand to tutors in general.

TutorAid is a comprehensive solution with complete integration of its features. Some of its features include: tasks, reminders, notes, student profile and earnings. Our features have been tailored to suit the needs of our target audience. By storing their information in one central location, tutors' workspace can be made more organized. This can help improve productivity and enable a smooth workflow. A clear space is indeed a clear mind.

TutorAid is a Command Line Interface (CLI) based tool that caters to professionals adept at typing. We have recently added more Graphical User Interface (GUI) features to make TutorAid more user friendly and to increase efficiency..
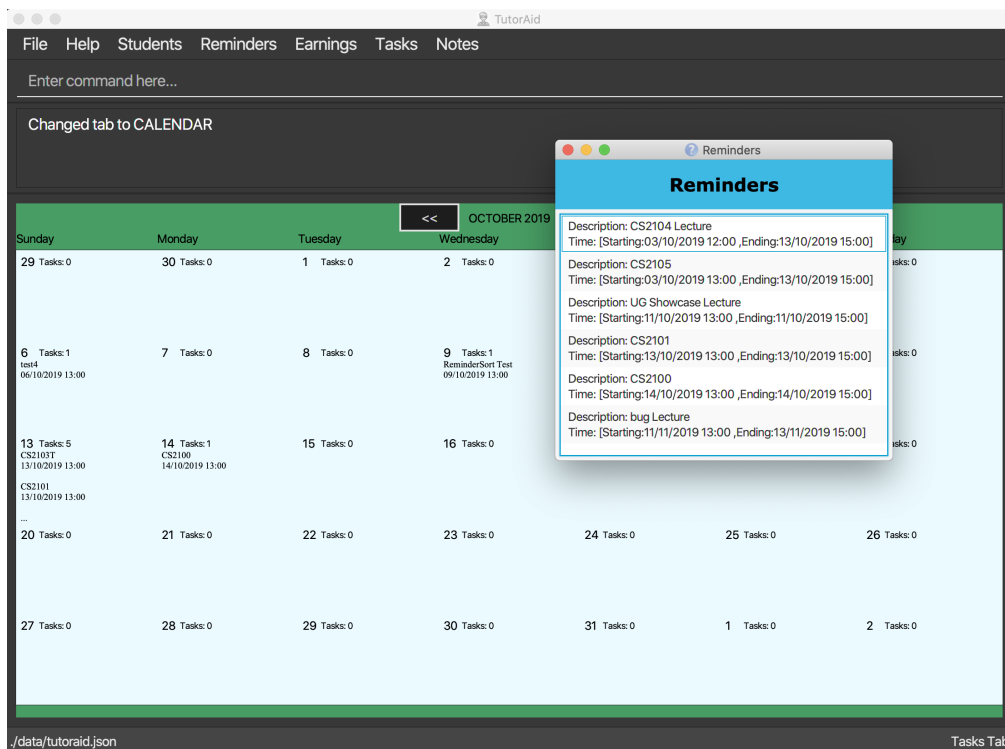
Figure 1. The Graphical User Interface of TutorAid

Role

My main role was to prototype and develop the Reminders feature, the Calendar feature and make enhancements to the Graphical User Interface. 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.

Summary of contributions

- **Major enhancement**: added **Reminders Feature**

  ◦ What it does: allows users to add Reminders. These will then be displayed in a reminders window that will pop up and remind users of urgent tasks.

  ◦ Justification: This feature improves the product significantly because users can be reminded of urgent and important tasks at hand. This will ensure that users have ample time to plan and complete their tasks punctually.

  ◦ Highlights: This feature allows users to create, remove and find Reminders. It required an in-depth analysis of design as it is also linked to the the Tasks feature. Users can create reminders for a specific task when creating said task. This highlights TutorAid's integration of all features with each other.

- **Major enhancement**: added **Calendar Graphical User Interface**

  ◦ What it does: allows users to view their Tasks chronologically. These will be reflected on the Calendar in Calendar View. Calendar View helps users keep track of tasks and plan their schedule better.

  ◦ Justification: This feature improves the product significantly because a user can view their

overall schedule. This can help users locate their busy periods and plan accordingly. It will also alert users of what tasks are due and when.

- ◦ Highlights: Calendars have a plethora of benefits: helping users prioritize, stay on task, increase productivity etc. Its implementation required integrating with the Tasks feature to be able to accurately display the Tasks Description and Start Date. Clicking on a date will also bring up the list of tasks due on that particular date.
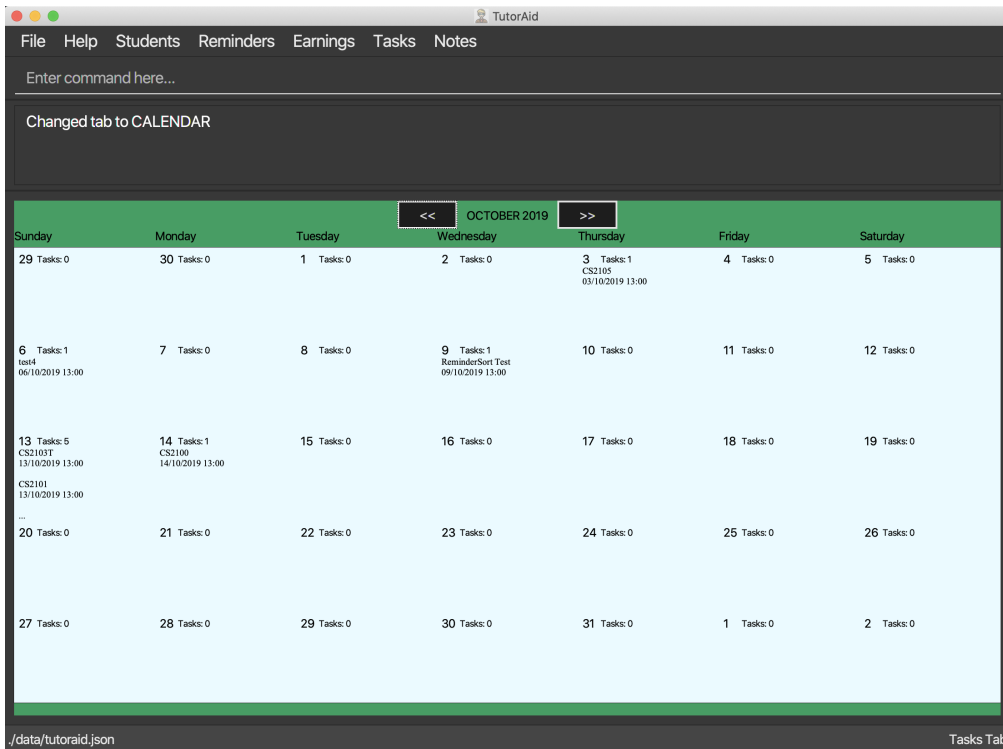

Figure 2. Calendar View

- **Minor enhancement**: added delete button for easy deletion of Reminders and Notes.
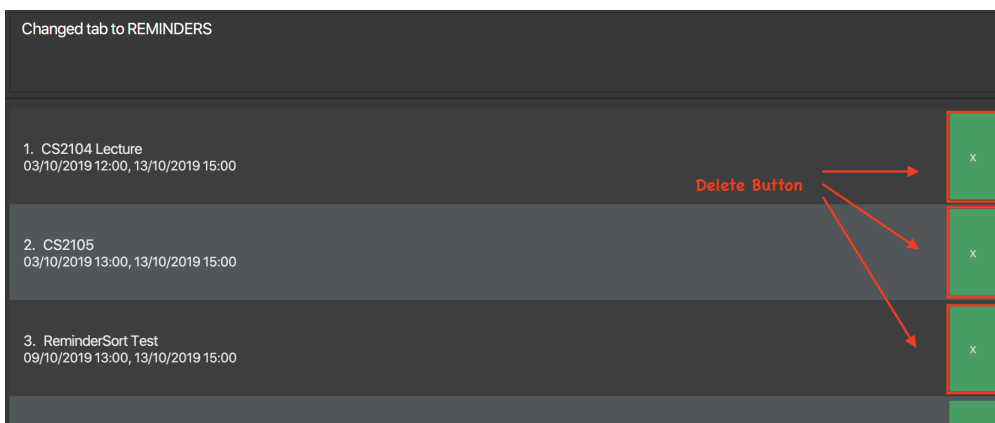

Figure 3. Delete Button

- **Minor enhancement**: added tabs in the Menu Bar for easy changing of tab.
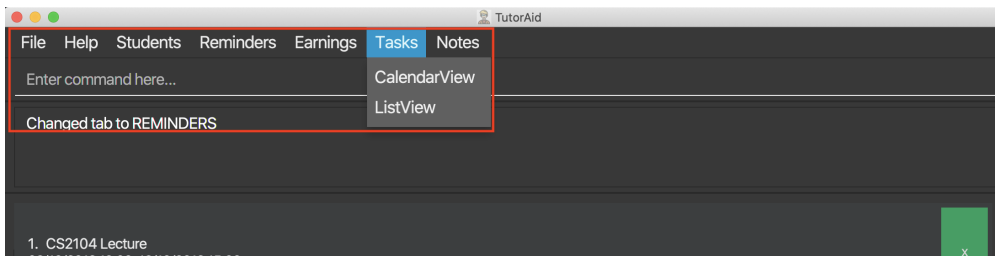
Figure 4. Change Tab

- **Minor enhancement**: added a tab status at the bottom right corner for users to know which tab they are viewing.

Figure 5. Tab status

- **Minor enhancement**: improved overall Graphical User Interface such as designing of the Login Page.
- **Code contributed**: [All commits] [Code Contributions]
- **Other contributions**:
  - Project management:
    - Managed and assigned these issues for the project: #177, #178, #229, #230
    - Managed bugs reported by other users in Practical Exam Dry Run: #268, #297
    - Managed releases `v1.1` - `v1.4` (4 releases) on GitHub
  - Enhancements to existing features:
    - Wrote multiple tests for existing features to increase code coverage incrementally (, , #)
    - Updated the GUI (Pull requests #222, #329, #337, #346)
  - Documentation:
    - Added detailed implementation documentation for the reminders, calendar and gui feature in User Guide
    - Did cosmetic tweaks to existing contents of the User Guide: #362
  - Community:
    - PRs reviewed (with non-trivial review comments): #161, #188
    - Reported bugs and suggestions for other teams in the class (examples: #109, #105)

Contributions to the User Guide

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

User Interface

Tab Status

The Tab that you are currently on will be shown on the bottom right corner

Change tab : `tab`

Change tab to any of the available ones.
Format: `change_tab tab/DESTINATION`

Examples (All available destinations listed):

- `change_tab tab/earnings`

- `change_tab tab/calendar`

- `change_tab tab/student_profile`

- `change_tab tab/reminders`

- `change_tab tab/notepad`

- `change_tab tab/task`

The user may alternatively choose to change tab by using the Items on the Menu Bar.

Delete Button

Delete Button for Reminders and Notes for quick and easy removal of Reminder or Note.

Reminders

Add Reminder

Adds reminders.
Format: `add_reminder rd/DESCRIPTION rt/START_TIME, END_TIME`

| TIP | A Reminder can have more than one time slots. |

- `START_TIME` and `END_TIME` must be in the format "dd/MM/YYYY HH:mm, dd/MM/YYYY HH:mm".

- If there are multiple task times, they will be automatically sorted based on their `START_TIME`.

Examples:

- `add_reminder rd/CS2103T Homework rt/13/10/2019 13:00, 13/10/2019 15:00`

Delete Reminder

Removes the reminder.
Format: `delete_reminder INDEX`

Examples:

- `delete_reminder 1`

  - `INDEX` must be a positive integer.

Reminders can also be deleted easily by clicking the delete button

Finding Reminders based on Description : `find_reminder_by_description`

Find specific reminders by description and list them.
Format: `find_reminder_by_description DESCRIPTION ⋯`

  - The `DESCRIPTION` is case insensitive. e.g `cs2100` will match `CS2100`
  - Only full words will be matched. e.g. `2100` will not match `CS2100`
  - Can find using more than one `DESCRIPTION` at a time.

Examples:

- `find_reminder_by_description CS2103T`
- `find_reminder_by_description CS2103T, cs2100`

Finding Reminders based on Date : `find_reminder_by_date`

Find specific reminders by date and list them.
Format: `find_reminders_by_date DATE ⋯`

  - The `DATE` should be in the format dd/MM/YYYY. e.g 12/10/2019

Examples:

- `find_reminders_by_date 13/10/2019`

Listing all reminders : `list_reminder`

List all reminders.
Format: `list_reminder`

  - Reminders are automatically sorted by Start `DATE` with the most upcoming being on top.

Reminder Window

The Reminder Window will pop up when Tutoraid is first loaded up.
It will list all the Reminders at hand.

Calendar View

Views the Task in Calendar View

TIP	Clicking on a date will show the user the Tasks with that date as its Start Time in normal list view. The Calendar will display the tasks starting on that date and the amount of tasks starting on that date. If there is a Task starting on that date, only the Task's `DESCRIPTION` and `START_TIME` will be displayed on the calendar.

- The maximum number of tasks that can be shown on each date is 2. If there are more than **2** tasks on a date, a ⋯ will be added at the bottom of that date to indicate that there are more tasks not shown.

- The Calendar will display the tasks starting on that date and the amount of tasks starting on that date.

- If there is a Task starting on that date, only the Task's `DESCRIPTION` and `START_TIME` will be displayed on the calendar.

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 contributions to the project.

Reminder Features

Add Reminder

The `add_reminder` command allows for tutors to add their reminders into TutorAid.

The format for the `add_reminder` command is as follows:

```
add_reminder rd/<DESCRIPTION rt/<TIME>
```

Overview

The add claim `add_reminder` mechanism is facilitated by `addReminderCommand` and `addReminderParser`, taking in the following input from the user: `Description`, `Time`, which will construct `Reminder` objects.
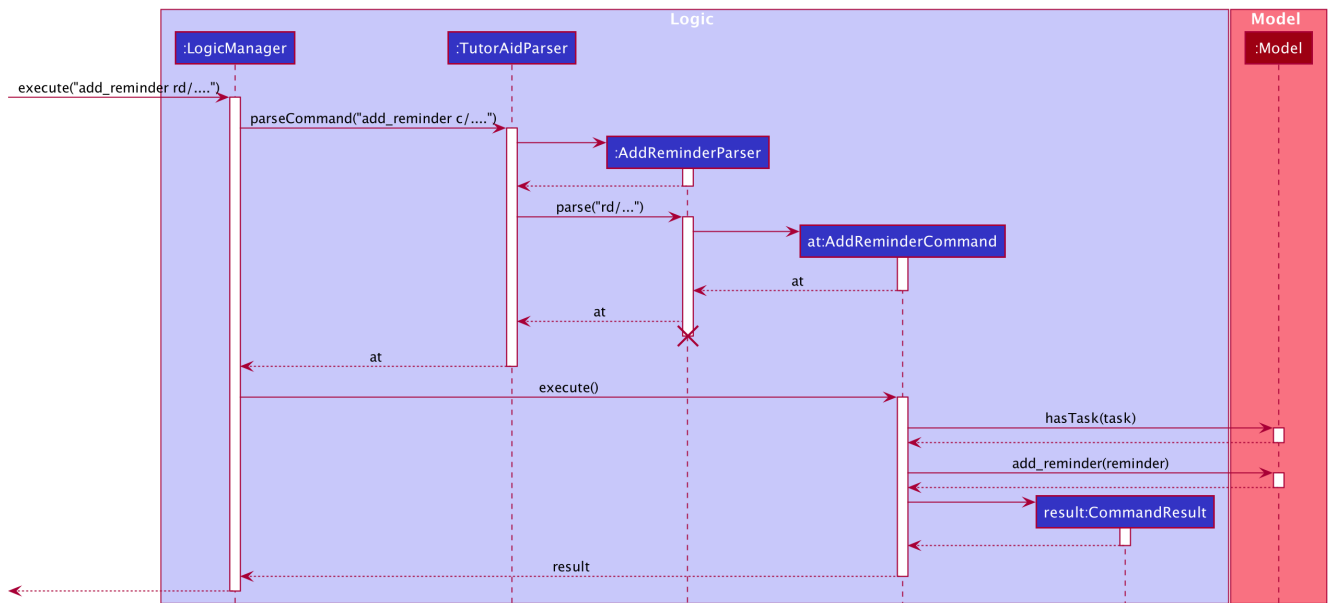
Figure 6. Add Reminder Command Sequence Diagram

NOTE The lifeline for `AddReminderParser` should end at the destroy marker (X) but due to a limitation of PlantUML, the lifeline reaches the end of diagram.

The `addReminderCommand` implements `Parser` with the following operation:

- `addReminderParser#parse()` - This operation will take in a `String` input from the user and create individual objects based on the prefixes `rd/` and `rt/`. The `String` value after the respective prefixes will create the respective objects. A validation check will be done to ensure that the strings that are entered by the user is entered correctly. If any of the strings entered are invalid, an error will be shown to the user to enter the correct format of the respective objects.

  - `description` would use `ParserUtil#parseReminderDescription()` to ensure that the description typed by the user is in the correct format.

  - `time` would use `ParserUtil#parseReminderTime()` to ensure that the content is in the correct format.

- After validation of the individual objects, a `Reminder` object would be created with the parameters `description` and `time`.

- `addReminderParser` would then return a `addReminderCommand` object with the parameter, `Reminder` object.

The following activity diagram summarizes what happens when a user executes a new command.
Example Scenerio

- Step 1: The user enters `add_reminder rd/teach rt/23/01/2019 13:00, 23/01/2019 15:00` to add a reminder for teaching classes.

- Step 2: `LogicManager` would use `TutorAidParser#parse()` to parse input from the user.

- Step 3: `TutorAidParser` would match the command word given by the user with the correct command. In this example, the given command is `add_reminder`, thus, `addReminderParser` object would be created with the user's input.

- Step 4: `addReminderParser` performs a validation check on each of the respective objects through

`addReminderParser#parse()`. In this case, it would use `ParserUtil#parseReminderDescription()` and `ParserUtil#parseReminderTime()`. It would then return a `addReminderCommand` object with a `Reminder` object.

- Step 5: `LogicManager` would execute `addReminderCommand#execute`. In this particular method, the `Reminder` object will be check with the rest of the prior `Reminder` object, to ensure that there is no duplicate `Reminder` object. If there are no similar `Reminder` object with the same parameters created previously, it would then be added into the reminder list.

- Step 6: `addReminderCommand` would then return a `CommandResult` to `LogicManager`, which would show the user that the new `Reminder` object have been successfully added.

Delete Reminders

This command allows the user to delete reminders from the existing reminders list through its index.

The delete reminders command is facilitated by the `DeleteReminderCommand` and `DeleteReminderParser`.

The delete reminder command is part of the logic component of our application. It interacts with the model component of our software application.

Overview

The `DeleteReminderParser` implements `Parser` with the following operation:

- `DeleteReminderParser#parse()` - This operation will take in an `int` input from the user, that the user wants to delete, which will correspond to the index shown on the reminders list on the application. This will delete the reminders in the reminders list. If an invalid format is entered or an invalid index is entered, there will be a prompt from the commmand parser.

Current Implementation

The delete earnings command is executed by the `DeleteReminderCommand`. Currently, the deletion of any earnings is done based on the `INDEX` of the reminders entry.

During the design of our delete reminders function, we considered between two alternatives.
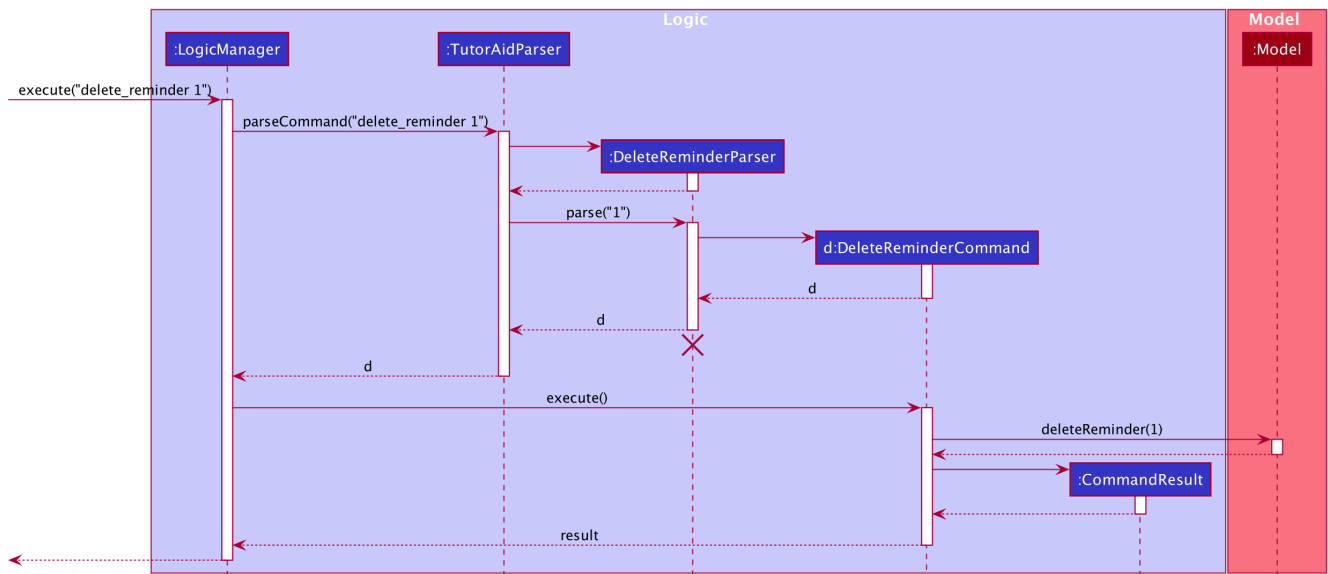
Figure 7. Delete Reminder Sequence Diagram

Example Scenerio

- Step 1: The user enters `delete_reminder 1` to delete a reminder with index 1.

- Step 2: `LogicManager` would use `TutorAidParser#parse()` to parse input from the user.

- Step 3: `TutorAidParser` would match the command word given by the user with the correct command. In this example, the given command is `delete_reminder`, thus, `deleteReminderParser` object would be created with the user's input.

- Step 4: `deleteReminderParser` performs a validation check on each of the respective objects through `deleteReminderParser#parse()`.

- Step 5: `LogicManager` would execute `deleteReminderCommand#execute`. In this particular method, the `Reminder` object will be of that index will be deleted from the current unique filtered reminder list.

- Step 6: `deleteReminderCommand` would then return a `CommandResult` to `LogicManager`, which would show the user that the new reminder list.

Tab Change feature

Tab Change feature is a type of Command that allows users to change to respective windows for using different features in the system. Tab change function is implemented in using both CLI and GUI. User can execute tab change by typing command in the command box or by interaction with the GUI component (Menu Bar)