Lim Kang Yee – Project Portfolio for treasurerPro (tP)

1. Introduction

This project portfolio briefly introduces the project, treasurerPro and outlines my contributions to the project and showcases the key feature I implemented.

1.1. About the team

My team of 5 members consists of 4 year 2 Computer Science Undergraduate students, including me, and another year 4 Computer Engineering Undergraduate student.

1.2. About the project

This project is part of the module 'Software Engineering Project CS2103T' where we were tasked to develop a basic command line interface desktop application by morphing or enhancing an existing AddressBook desktop application. Our team decided to incorporate and morph the AddressBook application as part of our all-in-one application which enables treasurers or members of Co-Curricular Activities (CCA) Clubs and Societies to manage their club finances, reimbursements, inventory and member's contact details easily. The duration of our project was 13 weeks.

1.3. Key to the icons and formatting used in the document

This symbol indicates extra information or definition.

`Model` : Text with grey highlight indicates a component, class or object in the architecture of the application.

`command` : Text with blue font and grey highlight indicates a command that can be inputted by the user.

1.4. Introduction of treasurerPro

This desktop application consists of 6 tabs, a command box for users to input their commands and a response box for Leo, our lion mascot. Each tab serves a different purpose that helps treasurers and members better manage their club or Society's finances.

This is what our application looks like when it is first opened. (graphical user interface for treasurerPro):

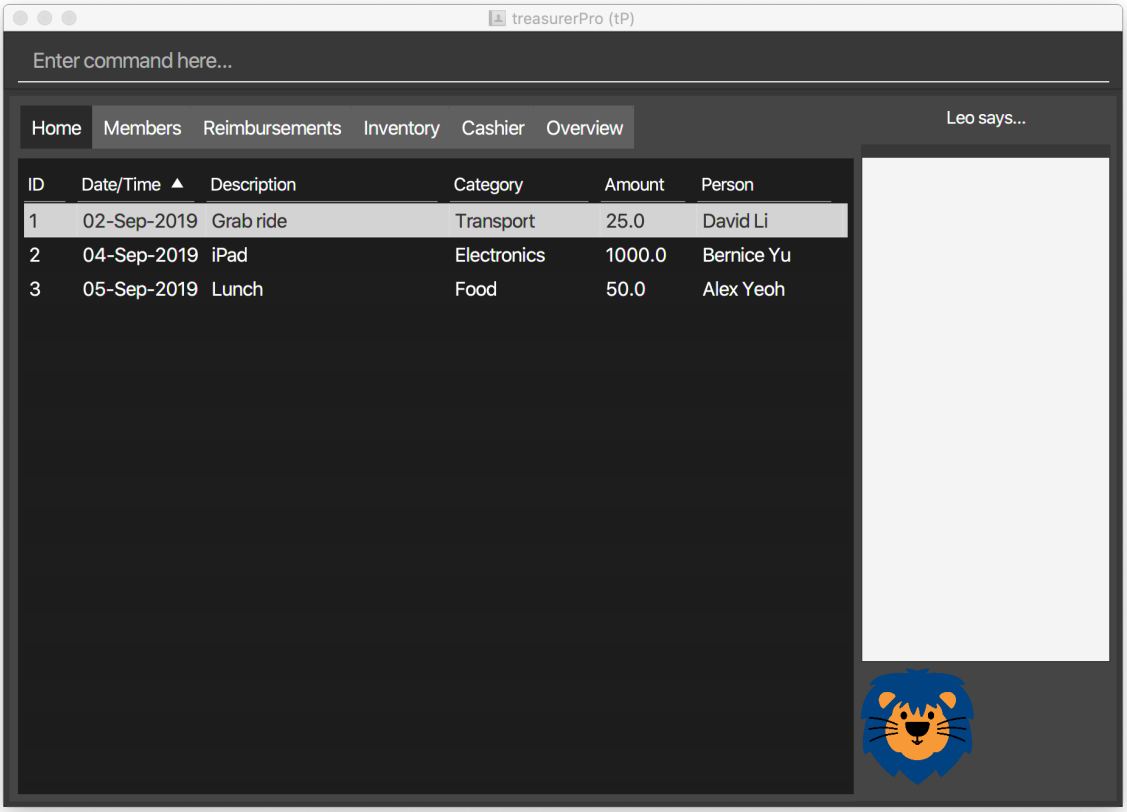


Figure 1. Graphical User Interface of treasurerPro

2. Summary of contributions

My role was to design and write the code for the features of the Home Tab. The following sections shows a summary of these features, as well as the relevant documentation I have added to the user and developer guides in relation to these features.

Enhancements

- Deleting Transactions of a Specific Member
    - What it does: The **delete** command allows all transactions linked to the specified member in the command to be deleted at once.
    - Justification: This feature allows the user to clear all transaction records linked to a specific member if the member has left the CCA since a member's details can only be deleted in the Members Tab if the member does not have existing transaction records.
    - Highlights: This feature is a faster way to delete transactions of a specific member, especially if the member has many transaction records. This works well with the deletion of member feature in Members Tab.
- Sorting Transactions in a Specific Order
    - What it does: Each transaction records consists of a date, description, category, amount of money and person accountable for the transaction. Thus, the **sort** command helps to sort the transactions by the alphabetical order of the person's name, by the date (from oldest to most recent) or amount (from smallest to largest).

    - Justification: It is useful for users to keep track of transactions and view the transaction records according to different priority.
    - Highlights: This command can be extended easily to allow for sorting of transactions in the reverse order or with other different orders.

Code contributed

Please click these links to view the code I have contributed for Home Tab: Functional Code , Test Code

Other contributions

- Community
    - Reviewed Pull Requests (with non-trivial review comments): (PR #98)
    - Added Detailed Explanation of Added Code in Pull Requests to Help Understanding By Other Members: (PR #42), (PR #201)
    - Helped to Debug Code in Other Member's package: (PR #126)
- Integration
    - Integrated `Transaction Tab` with `Reimbursement Tab`: (PR #49)
    - Integrated the Original `AddressBook` into the `Members Tab` in Graphical User Interface: (PR #42)
    - Integrated the Edit and Delete Command of `AddressBook` with the Logic of `Transaction Tab`: (PR #49), (PR #85)
- Documentation
    - Added implementation details for Home Tab, Model Component and Storage Component to the Developer Guide: (PR #154), (PR #199)
    - Added to User Stories and Guide to Use the Home Tab to the User Guide: (PR#22), (PR #197)

3. Contributions to the User Guide

The following section shows my additions to the treasurerPro User Guide for the `Home Tab` features.

3.1. Current enhancement

{start of extract from User Guide}

5.1.2 Delete Transaction(s)

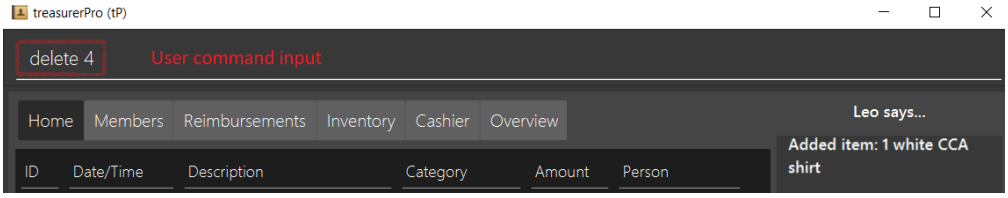
This command deletes either all transactions of a person or a single transaction of a specific ID from the table.

- Command: `delete ID` or `delete p/PERSON`
- Examples:
    - `delete 1`

    - `delete p/Alex Yeo`

- Steps for Deleting by ID:

    1. Type the command with the ID of the transaction to be deleted as shown in the screenshot below:

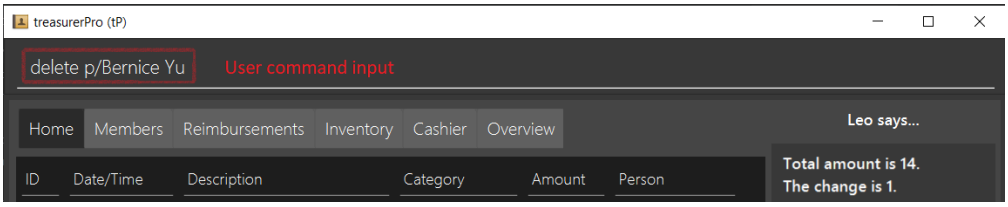


2. Hit `Enter`.

Leo will respond with a success message and the transaction will be removed from the table as shown below:

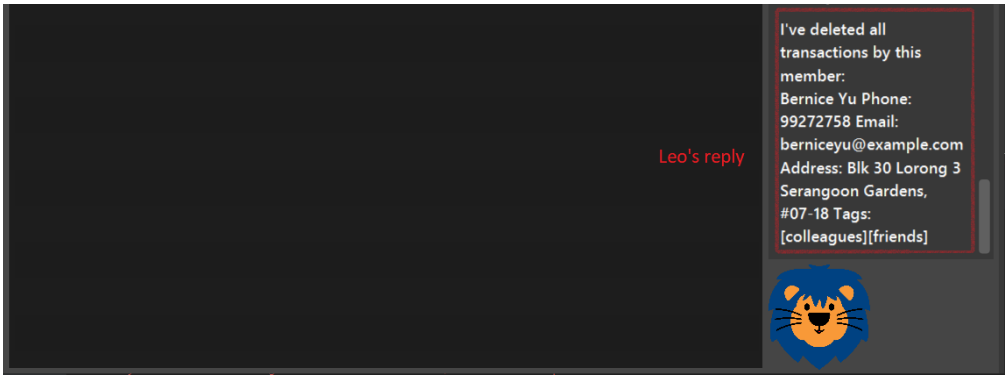
- Steps for Deleting by Person:

    1. Type the command with the person's name to delete all transactions related to that person, as shown in the screenshot below:



2. Hit `Enter`.

The transaction(s) will be removed from the table. Leo will also respond with the success message and as shown below:



If the transaction(s) deleted was part of a pending reimbursement record, it will also be removed

from that reimbursement record. On the other hand, if the person entered is not part of our data base shown in the Members Tab, Leo will respond with a message to inform you as shown in the screenshot below:

In addition, if the person does not have any transactions, Leo will also respond with a message to inform you. {end of extract}

{start of extract from User Guide} 5.1.4 Sort Transactions in the Table

This command sorts the table of transactions into a specified order for viewing and carrying out of subsequent commands.

- To sort:
  - By date (from oldest to most recent): `sort date`
  - By name (from alphabetical order of name): `sort person`
  - By amount (from largest to smallest in amount): `sort amount`
  - Undo sort: `sort reset`

The undo sort command allows you to view the table of transactions in the order originally shown when the application was initially opened.

{end of extract}

4. Contributions to the Developer Guide

The following section shows my additions to the treasurerPro Developer Guide for the `Home Tab` features.

4.1. Current enhancement

{start of extract from Developer Guide}

3.1.2 Delete Feature

This feature allows for 2 types of deletion, by the index shown in the table or by the person's name. Inputting the person's name will cause all transactions linked to that person to be deleted.

The following sequence diagram shows how the delete by name command works:

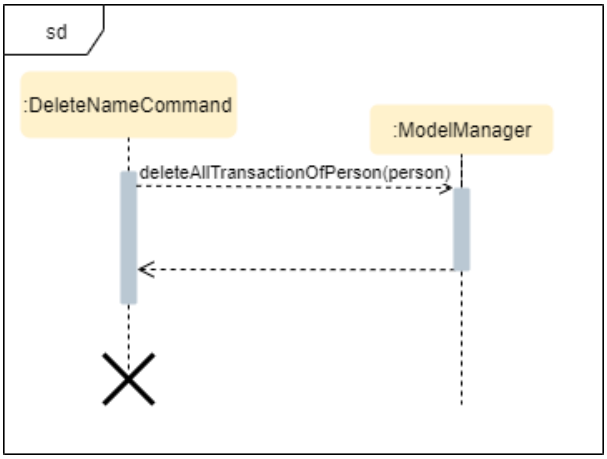


Figure 14. Sequence Diagram of Delete Command in Home Tab (transaction package)

In addition, the `resetPredicate()` method in `ModelManager` is not called in the `DeleteNameCommand`. Thus, the UI table will continue to show the filtered transaction list. If the prior input is a Find Command and the list at the start of the activity diagram shows a filtered list, the table will continue to show the filtered list at the end of the Delete Command. To view the full transaction list, the user would be required to input the Back Command where `BackCommand` calls `resetPredicate()`. The sequence diagram for the `BackCommand` is shown in the following section 3.1.3 BackCommand

After this, the list of transactions and reimbursement tab is updated as shown in Figure 11 and Figure 12 respectively. The delete by index implementation would be similar but does not require interaction with the `Model` from the `AddressBook` in the person package.

The following activity diagram shows the steps needed to delete a new transaction:

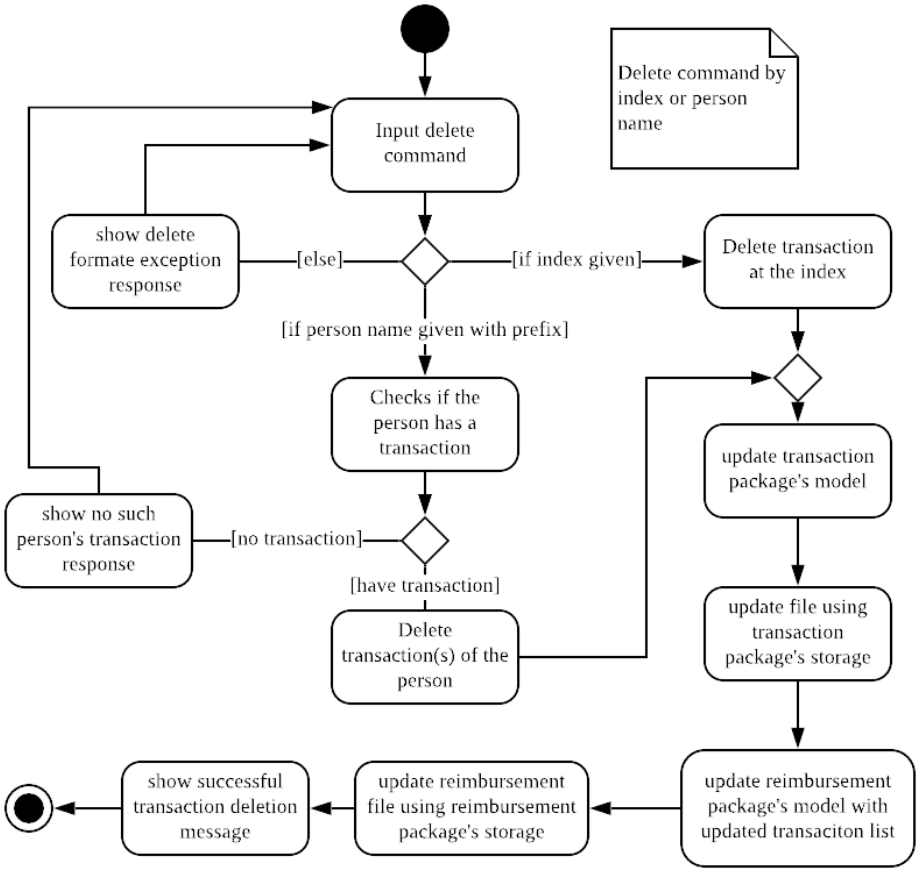


Figure 15. Activity Diagram of Delete Command in Home Tab (transaction package)

The above activity diagram assumes the index to be within the bounds of the table but if it is not, a response will be shown about the incorrect input. Also, as shown above, responses will be shown to indicate if an input is incorrect or when a successful deletion is done.

3.1.3 Back Command Feature

The `BackCommand` is not initialised by a specific command parser as shown in as shown in Interactions Inside the Logic Component for a Command but initialised by the `TransactionTabParser` instead. The following detailed sequence diagram shows how the back command works:

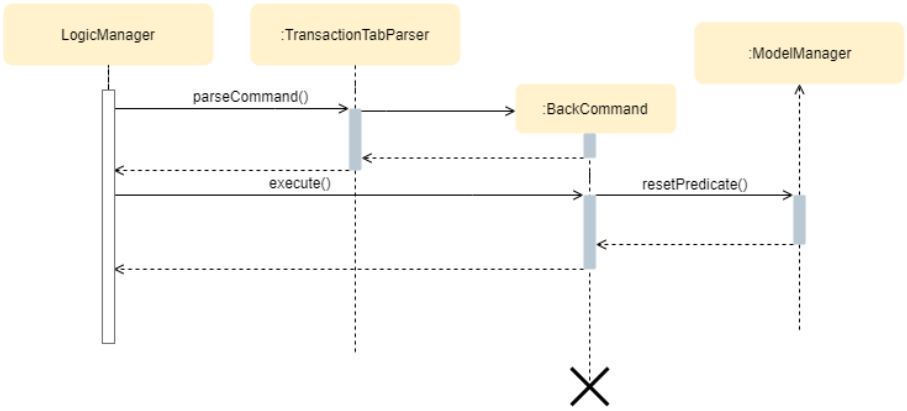


Figure 16. Sequence Diagram of Back Command in Home Tab (transaction package)

3.1.4 Sort Command Feature

The `SortCommand` allows for 3 types of sort, by name in alphabetical order, by amount (from least to most) and by date (from oldest to most recent).

The following sequence diagram shows how the sort command works:

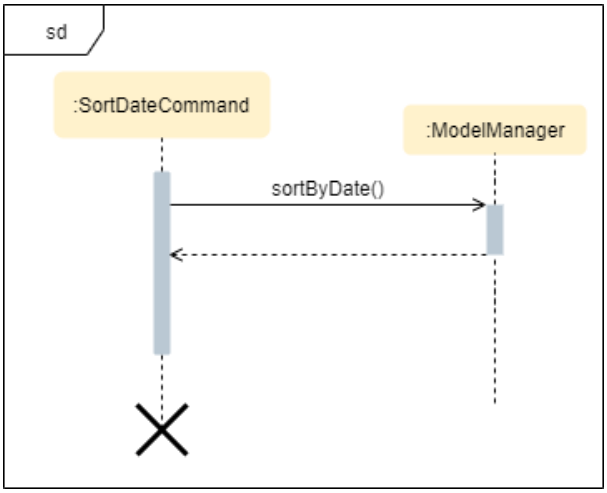


Figure 17. Sequence Diagram of Sort Command in Home Tab (transaction package)

When a user inputs the sort command, it is only checked that it is one of the 3 types or it will show a response about the incorrect user input. When it is successfully sorted, there will also be a response message shown.

Similar to the Delete Command, the `resetPredicate()` method in `ModelManager` is not called.

3.1.5 This section's table explains the design considerations for some implementations in the Home Tab.

Alternative 1	**Alternative 2**	**Conclusion and Explanation**
The Reimbursement Tab's `Model` and `Storage` interface is passed as parameters into Transaction Tab's `Logic` to call the methods in `Model` and `Storage` to update all the reimbursement data after a `Command` is executed in Home Tab.	The methods in Reimbursement Tab's `Model` and `Storage` is called to update all the reimbursement data in `MainWindow` after a `Command` is executed in Home Tab.	Alternative 2 was implemented. This limits access to all public methods in Reimbursement Tab's `Model` and `Storage` in Transaction Tab's `Logic` to prevent unwanted modifications to any of the reimbursement data.
The Members Tab's `Model` interface is passed as parameters into Transaction Tab's `Logic` to give `Logic` access to all public methods of `ModelManager`.	A new interface is made to allow the only used method of Members Tab's `ModelManager` to be accessed in Transaction Tab's `Logic`.	Alternative 2 was implemented. The new interface acts as a facade for `ModelManager` which prevent unwanted modifications to `AddressBook`.
An `ArrayList` is used to store `Transaction` objects in `TransactionList`.	A `LinkedList` is used to store `Transaction` objects in `TransactionList`.	Alternative 1 was implemented. An `ArrayList` has better performance for the set and get methods than a `LinkedList` which would beuse frequently in `ModelManager`.

{end of extract}