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 explains the feature I implemented.

1.1. About the team

My team consists of 5 software engineering students under the module CS2103T. 4 of us, including me, are year 2 Computer Science students and the other is a year 4 Computer Engineering 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 [1. CLI] 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 to members, inventory and member's contact details easily.

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

This symbol indicates extra information or definition.

This symbol indicates important information to take note of.

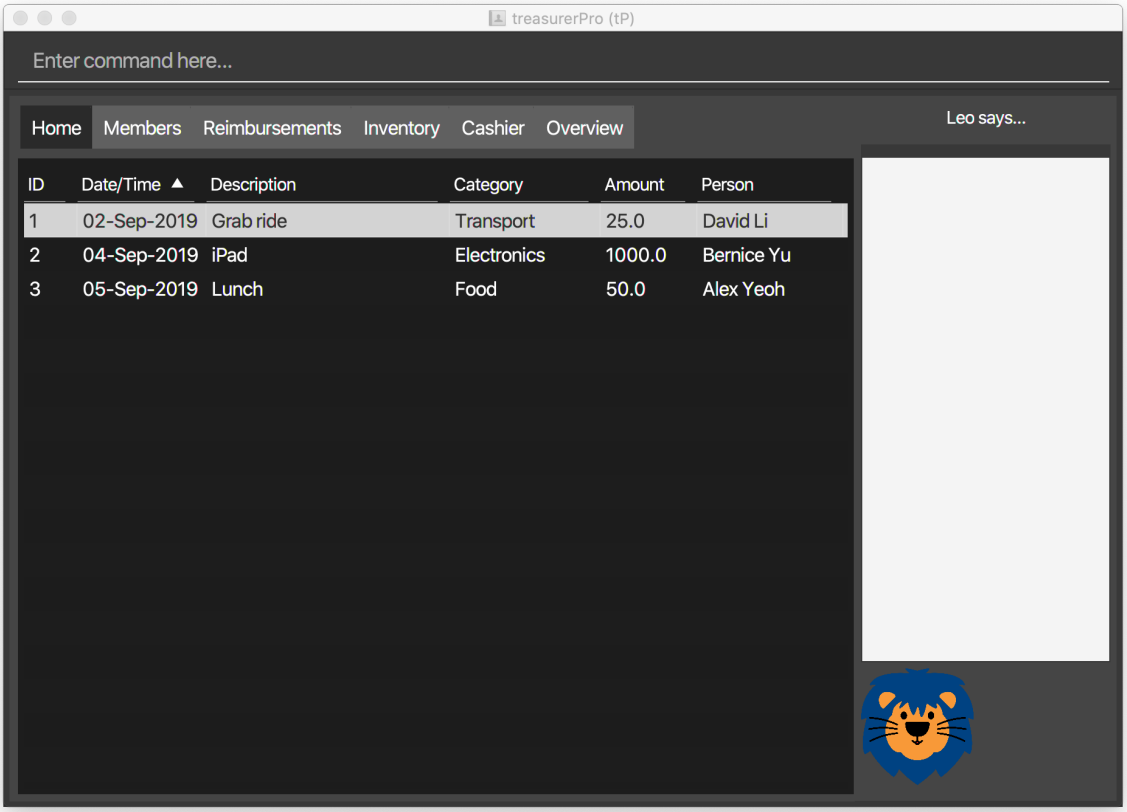
`Model` : Text with grey highlight indicates a component, class or object in the architecture of the application.

`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 [2.GUI] for treasurerPro):



2. Summary of contributions

My role was to design and write the codes for the features of the `Home Tab`. 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.

2.1. Enhancements

I added the enhancements of the different commands for the Home Tab.

2.1.1. Add, Delete and Edit Commands

The basic create, read, update and delete (CRUD) commands were added such that the user can `add`, `delete` and `edit` transactions to keep track of them.

- What it does: It allows the user to create, delete and update transactions. It also allows the application to restore data from previous usage of the application by reading and saving data in a background text file.
- Justification: It forms the fundamental features required for the `Home Tab` to be useful for users to keep track of transactions.
- Highlights: These commands help to store and keep track of transactions in the `Home Tab` but also help keep track of sales transactions from the `Cashier Tab`. The transactions for each member's spending in this tab are also tabulated in the `Reimbursement Tab` to help the treasurer keep track of pending reimbursements.

2.1.2. Sort Command

A `sort` command was also implemented for the transactions in the table to sort the commands to a certain order.

- What it does: The addition of transactions requires the user to input the date, description, category, amount of money and person accountable for each transaction. Thus, this 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 largest to smallest).
- 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.

2.2. Code contributed

The code that I have contributed can be found in the following links:

- Functional Code
- Test Code

The links lead to our team's GitHub repository

2.3. Other contributions

The following section leads to the GitHub pull requests [3. PR] related to the contribution.

- Integration of `Transaction Tab` with other newly implemented tabs
  - Integrated `Transaction Tab` with `Reimbursement Tab`: pull request #49

Our workflow requires us to push to our own forked repository before making a pull request to our group repository which allows our members to review the new code before it can be merged with the current code in the group repository.

- Integration of existing `AddressBook` into our application
  - Integrated the original `AddressBook` into the `Members Tab` in graphical user interface [2. GUI]: pull request #42
  - Integrated the Edit and Delete command of `AddressBook` with the logic of `Transaction Tab`: pull request #49, pull request #85
- Tools
  - Added Travis CI [4. TravisCI] for the repository.

By version 1.2, our team ensures that we only merge pull requests that pass all the tests to ensure better code quality.

- Documentation
  - Added to user stories to the User Guide: pull request #22
  - Edited the READ.ME cover of our repository: pull request #19

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)

5.1.1 Add a Transaction

This command helps to add a transaction record into the table and to be saved into the system.

- Command format: `add dt/DATE d/DESCRIPTION c/CATEGORY a/AMOUNT p/PERSON`

Examples: `add dt/24-Aug-2019, 07:00PM d/Printer ink c/Miscellaneous a/3.50 p/Janelle`

The format of the date has to be in dd-MMM-yyyy format. (Eg. 24-Aug-2019 or 03-Sep-2019)

- Steps:
  1. Type the command with all the parameters filled in as shown in the screenshot below:

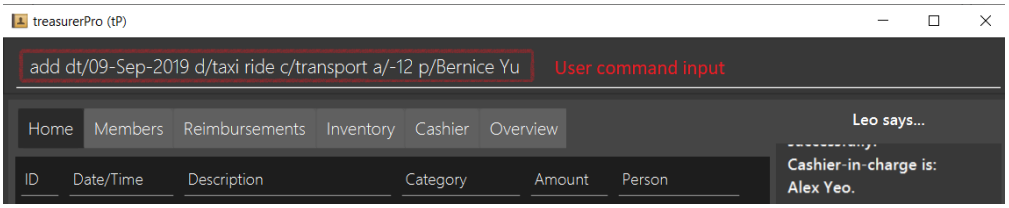


Figure 1. Screenshot of user input into command box for add command in Home tab

The format of the date has to be in dd-MMM-yyyy format. (Eg. 24-Aug-2019 or 03-Sep-2019)

  2. Hit `Enter`.

If the command is successfully added, Leo will respond with a success message and the transaction will be shown in the table. This is shown in the screenshot below:

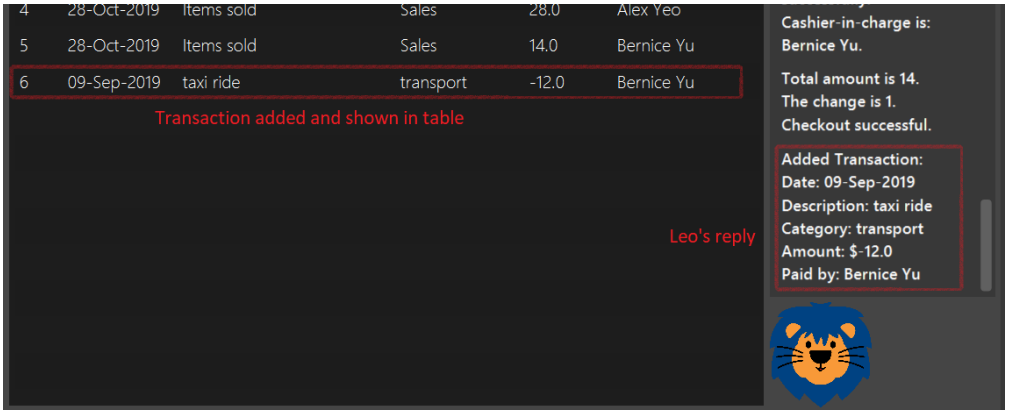


Figure 2. Screenshot of a successful user input for Add Command in Home Tab

If the person's name does not exactly match a current member in the Members Tab, Leo will respond with an error and the transaction record will not be added. This is shown in the screenshot below:

Figure 3. Screenshot of an unsuccessful user input for Add Command in the Home Tab due to an invalid name

If the added transaction contains a negative amount (indicating an expenditure), a corresponding entry will automatically be shown in the Reimbursement Tab, tagged to the member who spent it.

Since reimbursements are aggregated by member, if the member already has other outstanding reimbursements, it will simply be added to his existing row.

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:

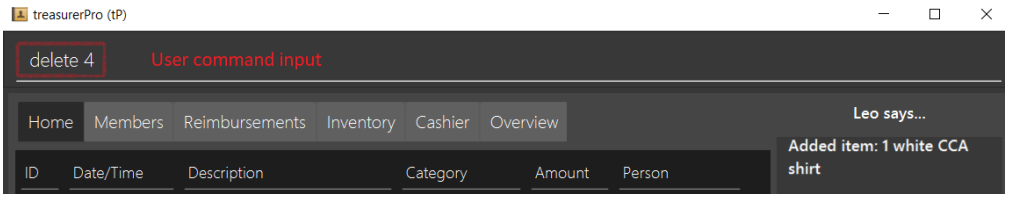


Figure 4. Screenshot of the user input into Command Box for Delete by ID command in Home tab

2. Hit `Enter`.

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

Figure 5. Screenshot of a successful user input for Delete by ID Command in Home Tab

- 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:

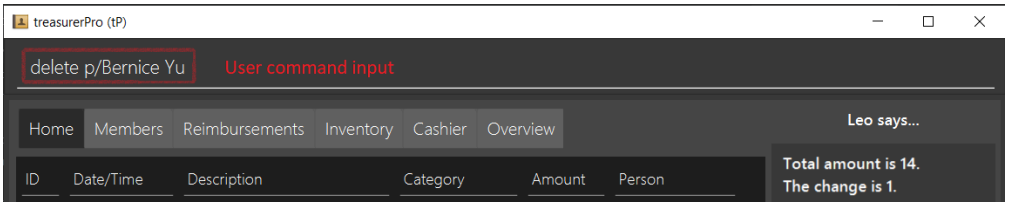


Figure 6. Screenshot of a user input into Command Box for Delete by ID command in Home Tab

2. Hit `Enter`.

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

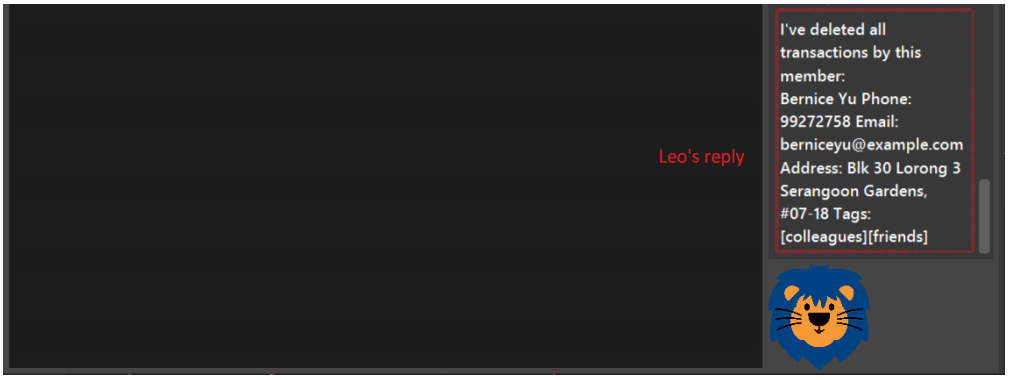


Figure 7. Screenshot of a successful user input for Delete by Person Command in Home Tab

If the person does belong to any transactions, Leo will also respond with a message to inform you.

5.1.3 Edit a Transaction in the Table

This command edits an existing transaction in the table, changing its details.

- Command: `edit ID dt/DATE d/DESCRIPTION c/CATEGORY a/AMOUNT p/PERSON`

The fields above can vary in their order. It is not compulsory to include all of them.

- Examples:
  - `edit 1 p/Bernice Yu dt/23-Aug-2019`
  - `edit 3 a/12`
- Steps:
  1. Type the command with the ID of the transaction to be edited, along with the new parameters to be changed, as shown in the screenshot below:

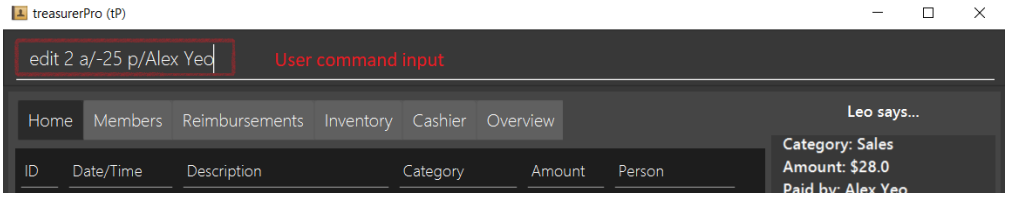


Figure 8. Screenshot of user input into Command Box for Edit Command in Home Tab

2. Hit `Enter`.

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

Figure 9. Screenshot of a successful user input for Edit Command in Home Tab

If the person entered into the command is not found in the Members tab, Leo will respond to inform you which is similar to Figure 3.

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)

3.2. Proposed enhancement for v2.0

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)

3.1.1 Add Command feature

This feature requires access to the `Model` of the person package which the `AddressBook` implementation is contained in. All fields in the transactions are compulsory to be inputted by the user: date, description, category, amount, person full name. The person's name inputted has to match a name already existing in the `AddressBook` which is shown in our Members Tab.

The following sequence diagram shows how the `AddCommand` works and is the reference from Interactions Inside the Logic Component for a Command:

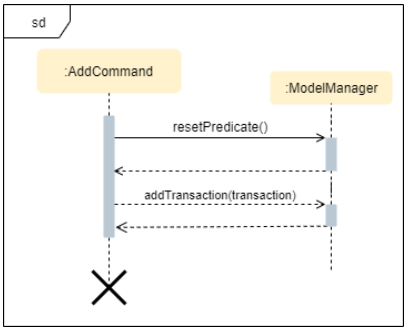


Figure 10. Sequence Diagram of Add Command in Home Tab (transaction package)

In addition, the `resetPredicate()` method from `ModelManager` is called in the `AddCommand`. Thus, the UI table will immediately shows the full transaction list regardless of the list shown at the start of the activity diagram. If the prior command was a Find Command, then the list in the beginning of the activity diagram would be a filtered list but after the add command is executed, the full list of transactions would be shown.

After the command is executed, the `LogicManager` updates the in-app list of transactions via the `ModelManager` and updates the data file via the `StorageManager`. The following sequence diagram

shows how the updating of the list of transactions in the app and in the data file:

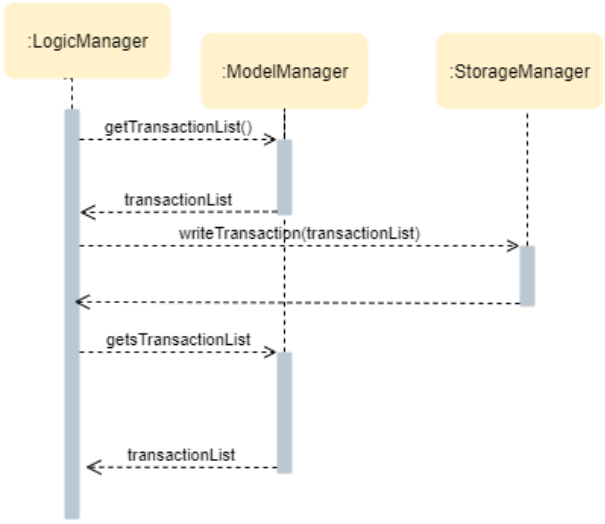


Figure 11. Sequence Diagram of updating the transaction list in Home Tab (transaction package)

This update of the list of transactions is done for every command that is executed successfully in the Home Tab.

Finally, the `StorageManager` and `ModelManager` inside the Reimbursement package will be updated with the latest list of transactions to generate an updated list of reimbursements for the user to view in the Reimbursement Tab. The following sequence diagram shows how the Reimbursement Tab is updated from the `MainWindow`:

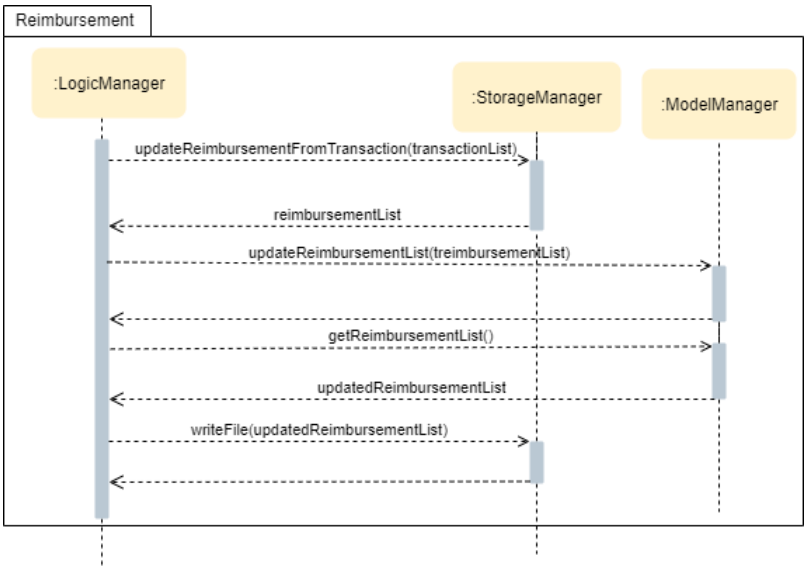


Figure 12. Sequence Diagram of updating the reimbursement list in Reimbursement Tab (transaction package)

This update of the Reimbursement Tab is done for every command after the list of transactions is updated (shown in Sequence Diagram of updating the transaction list in Home Tab) when there is a command executed successfully in the Home Tab.

To better illustrate the flow of events from the moment a user inputs a command till the completion of the command, the activity diagram for the Add Command is shown below:

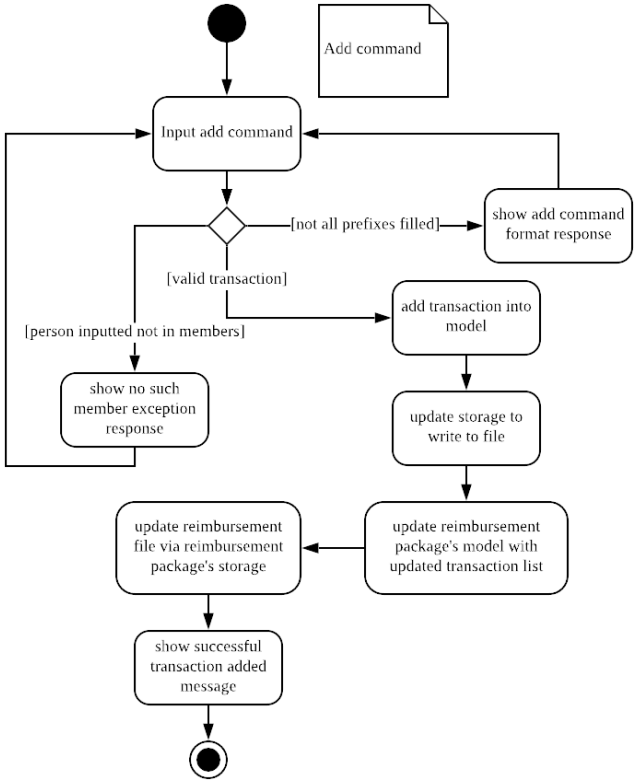


Figure 13. Activity Diagram of Add Command in Home Tab (transaction package)

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:

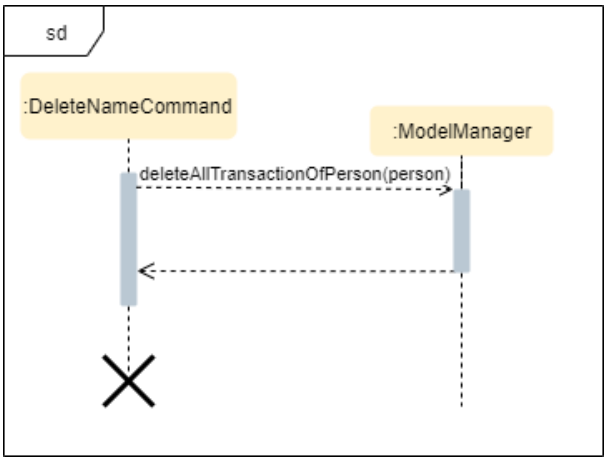


Figure 14. Sequence Diagram of Delete Command in Home Tab (transaction package)

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 by the Find Command's keywords, it will continue to show the filtered list at the end of the 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.

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:

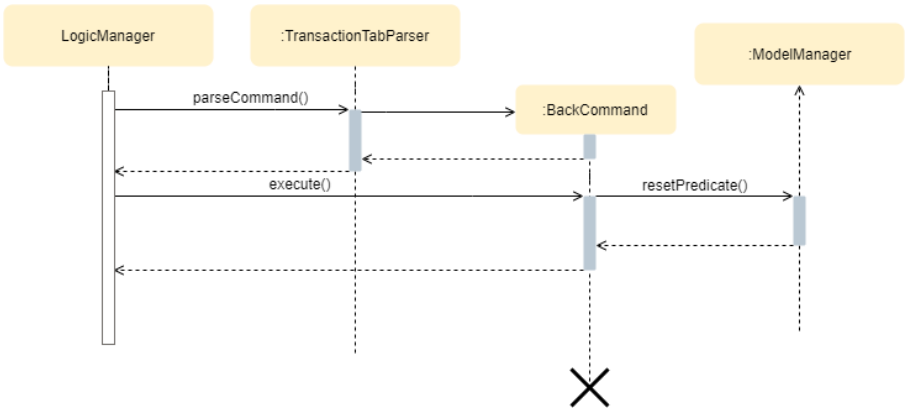


Figure 16. Sequence Diagram of Back Command in Home Tab (transaction package)

(end of extract)

4.2. Proposed enhancement for v2.0

Glossary

1. Command line interface (CLI): Command line interface applications allow users to use all the features in the app by typing and without a need to click on anything

2. Graphical user interface (GUI): Refers to the visual means of interacting with the application, such as windows, buttons and menus.

3. Pull request (PR): Making a pull request is a way of submitting contributions to an open development project. It allows the developers to approve code before it is added or merged with the current code in the project.

4. Travis CI: Travis will automatically detect when a commit has been made and pushed to the GitHub repository that allows it access, and each time this happens, it will try to build the project and run tests according to the standards set by the developers.