

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 [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 important information or definition.

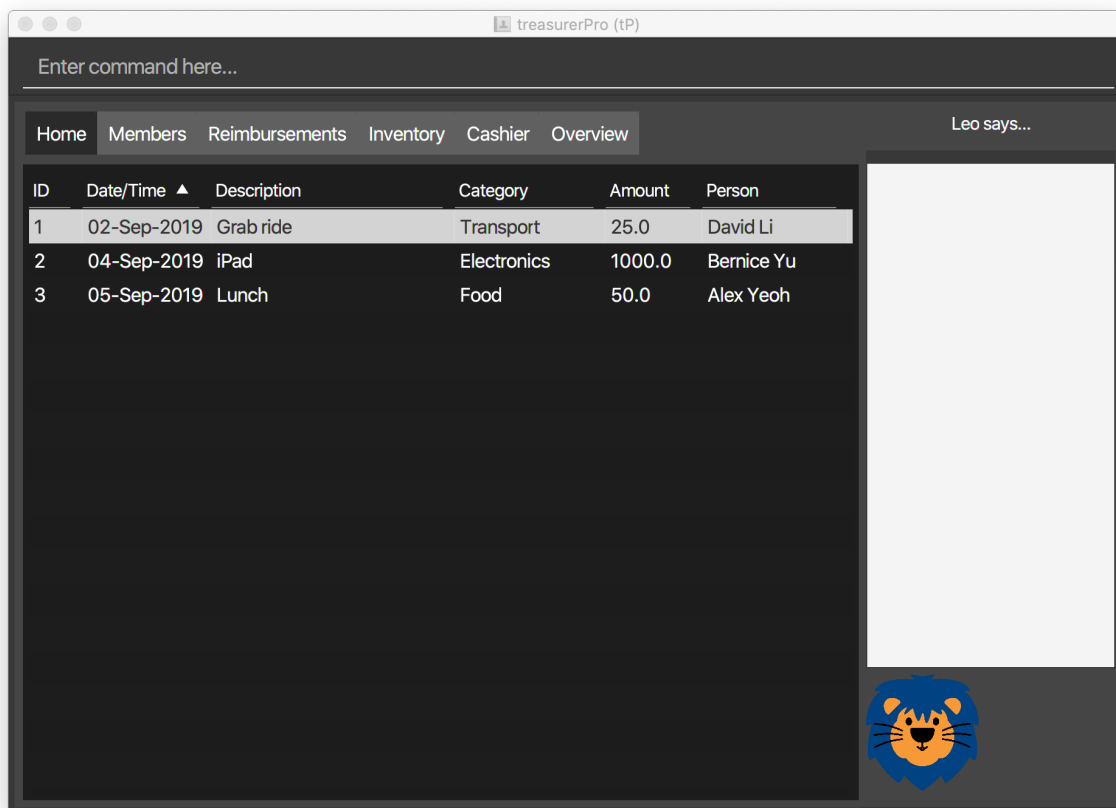
Model : Text with grey highlight indicates a component, class or object in the architecture of the application.

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. The home tab keeps track of individual transactions. The members tab keep track of all contact details of members. The reimbursement tab keeps track of reimbursement records for members that have spent for the club. The inventory tab keeps track of items for sale. The cashier tab is a convenient way for the club or society to do cashiering duties which helps to directly input sales into the system. The overview tab allows treasurers to plan the club or society's finances.

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

2.1.3. Find Command

A **find** command was also implemented to help find specific transactions that match the keywords inputted by the user.

- What it does: The command allows user to filter out the transactions to only show those that match the keywords from all the transactions recorded.
- Justification: This allows users to easily find and filter out transactions to call another command to manage the transactions better.

2.1.4. Go and Exit Command

Lastly, a **go** command and **exit** command was implemented to help navigate to another tab and leave the application.

- What it does: The **go** command helps the user to navigate to another tab without having to click on the tab and the **exit** command allows the user to leave the application without having to click on the window's exit icon as well.
- Justification: This ensures the application is a command-line interface application which requires no clicking at all.

2.2. Code contributed

The code contributed by me 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 [\[PR\]](#) related to the contribution.

- Integration of **Transaction Tab** with other newly implemented tabs
 - Integrated **Transaction Tab** with **Reimbursement Tab**: [pull request #49 \[PR\]](#)



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 [\[GUI\]](#): [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 [\[TravisCI\]](#) for the repository.



By version 1.2, our team ensures that we only merge pull requests that pass all the tests to ensure we can meet the milestones with higher 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

3.1. Current enhancement

(start of extract from User Guide) 5.1 Home Tab

This section will contain the details on all commands available to the Home tab.

Summary of Features of the Home Tab

- Home tab shows the transaction history of each expenditure.
- The columns will show the date of the transaction, description of the expenditure, category it is under, amount spent and the person who bought it.
- At the side, our mascot lion 'Leo' will help to give replies to indicate successful addition, deletion, edits of the command line input.
- Leo will also give you replies to guide you when there is a wrong input.
- There is a function to sort the transactions by date so from latest to oldest, person so by alphabetical order of the person and amount so from most to least.
- The amount should be inputted as positive if the transaction is considered revenue and adds to the club or society's reserves
- The amount should be inputted as negative if the transaction is a spending to be reimbursed which will be shown in the Reimbursements tab.

5.1.1 Add a Transaction to the Table

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:

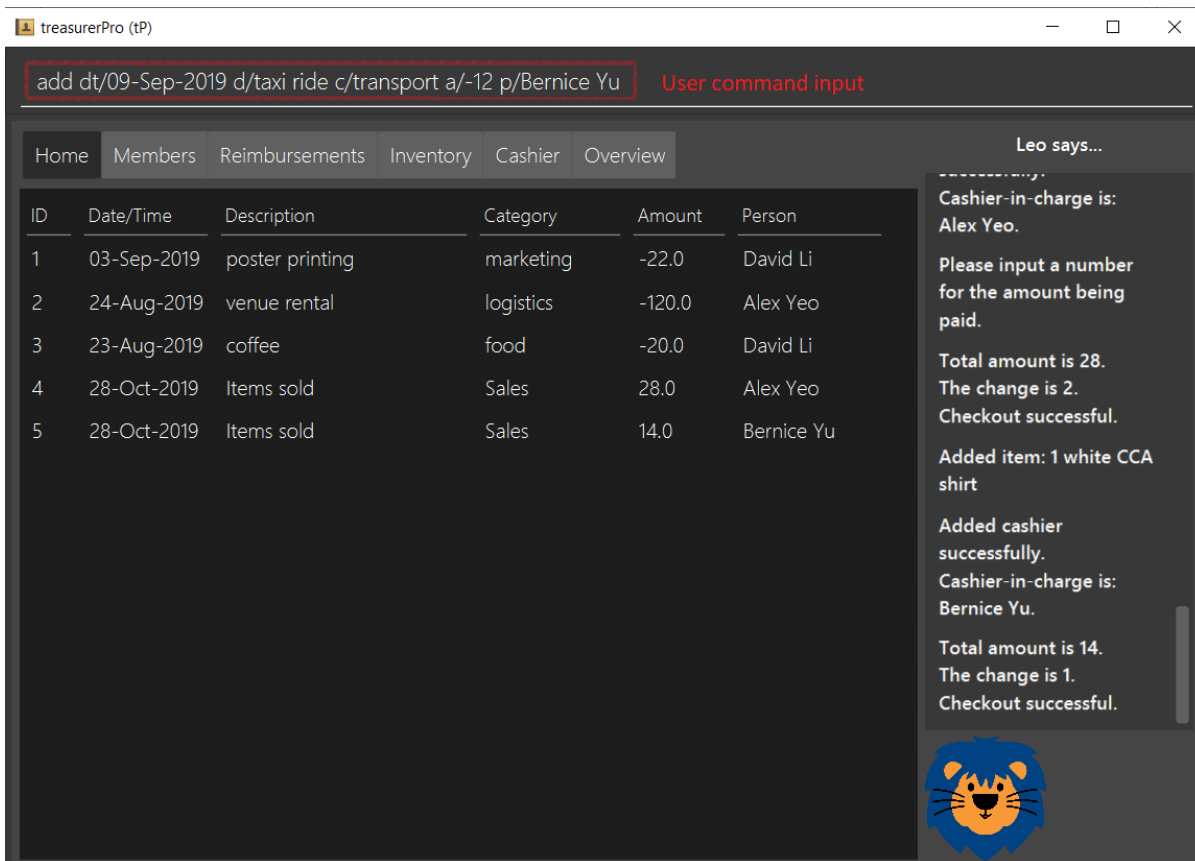


Figure 1. Screenshot of user input into command box for add command in Home tab

2. Enter the command.

If the command is successfully added, there will be a response from Leo and the transaction record will be shown in the table. This is shown in the screenshot below:

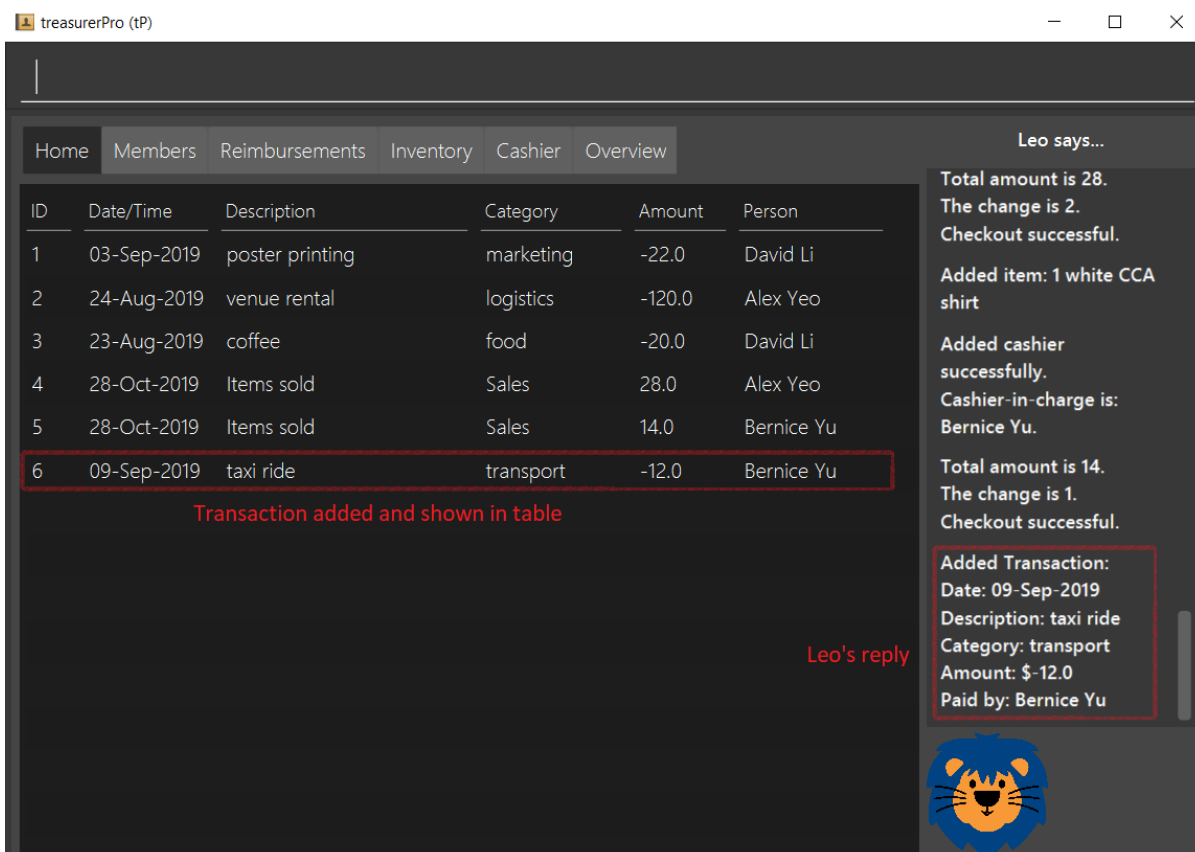


Figure 2. Screenshot of successful user input for add command in Home tab

If the person's name is not someone in the Members tab then there will be a response from Leo and the transaction record will not be added. This is shown in the screenshot below:

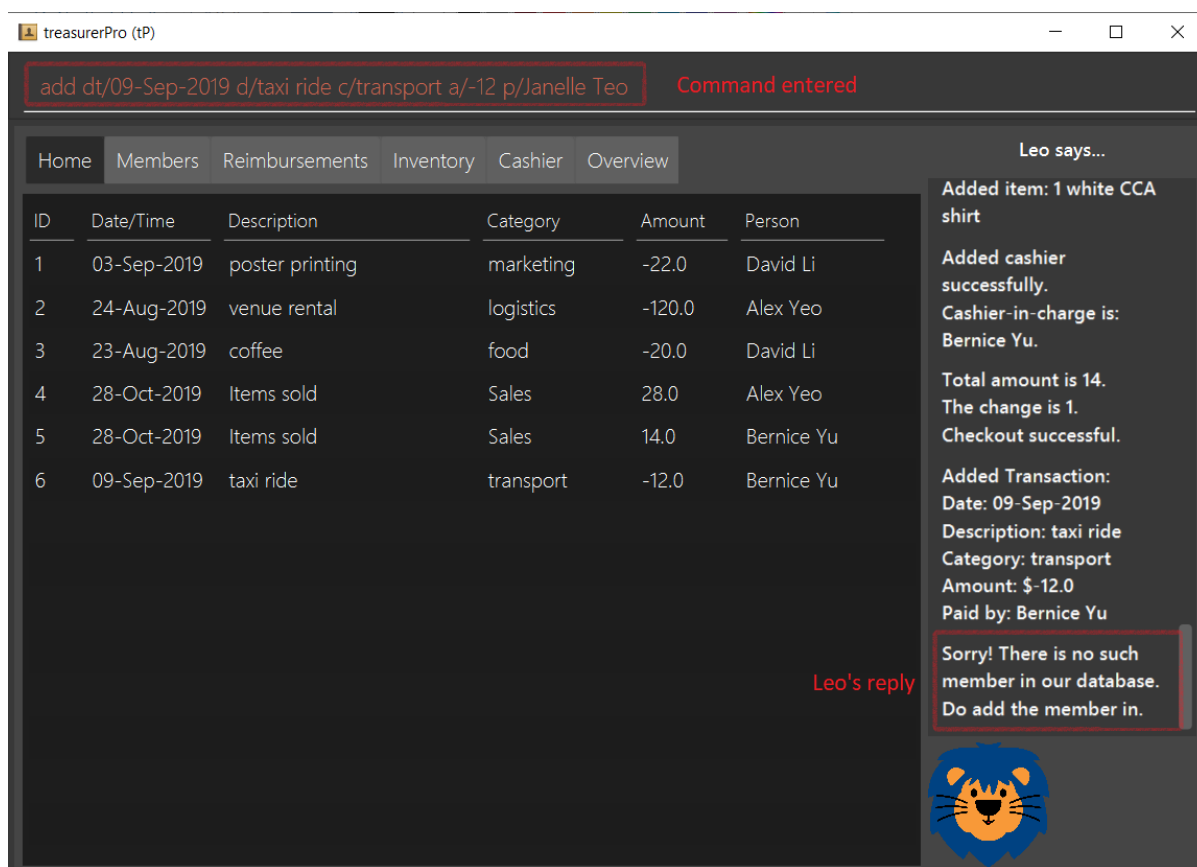


Figure 3. Screenshot of unsuccessful user input for add command in Home tab due to invalid person's name input

If the added transaction has a negative amount which indicates a spending, it will be shown in the Reimbursement Tab as a pending reimbursement record for the member that spent it. If there is an existing reimbursement record for that member still pending, the amount of the newly added transaction will be added to the existing pending reimbursement amount. This is shown in the screenshot below:

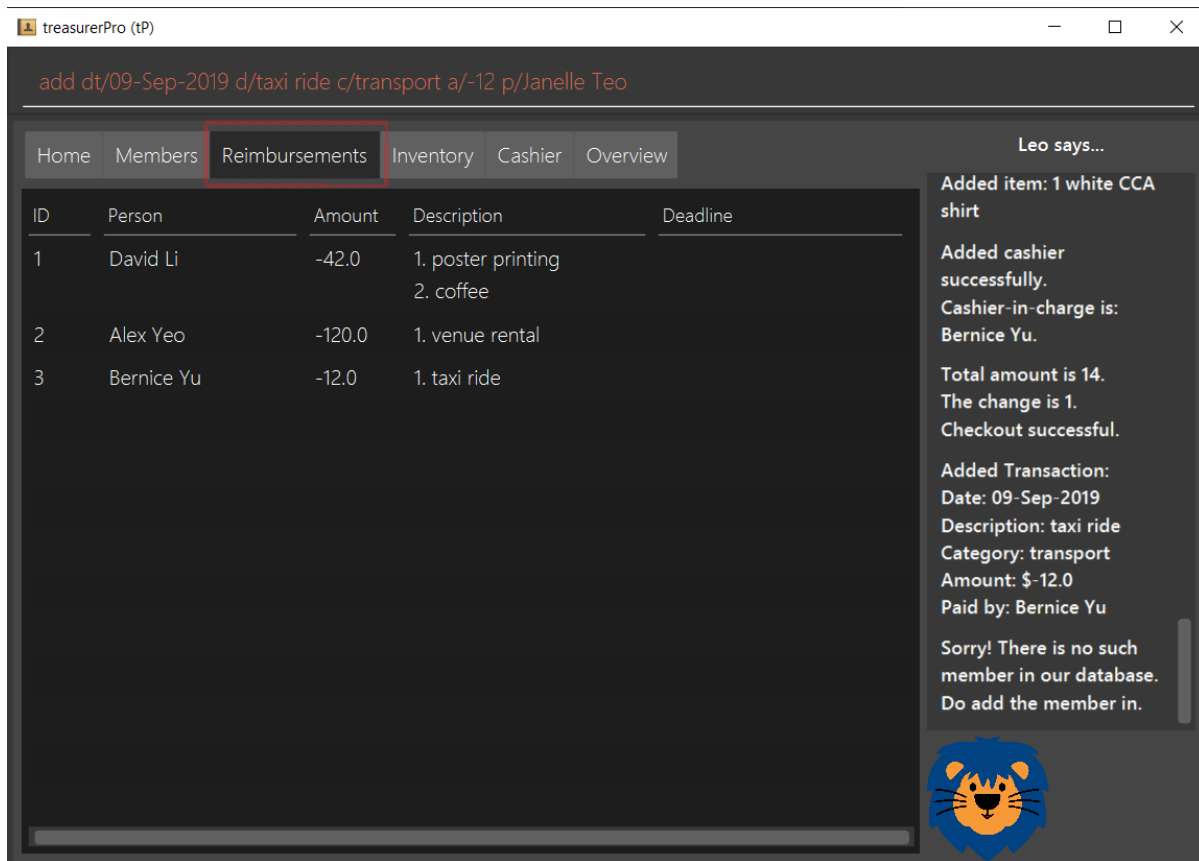


Figure 4. Screenshot of Reimbursement tab after successfully adding transactions

3.2. Delete Transaction(s) from the Table

This command helps to delete all transactions of a one person or a single transaction of a specific ID from the table.

- Command Format: `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:

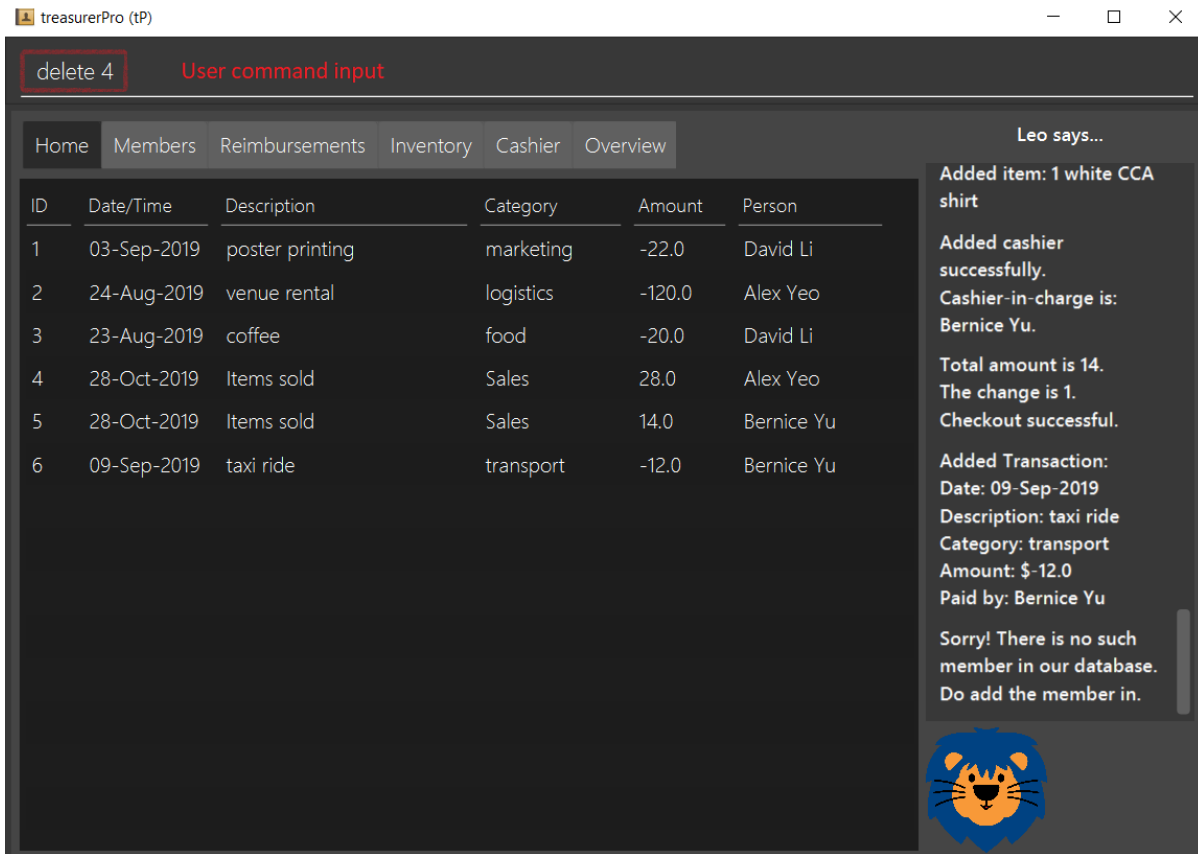


Figure 5. Screenshot of user input into command box for delete by ID command in Home tab

2. Enter the command.

Leo will respond with the successful deletion and the transaction will be removed from the table as shown below:

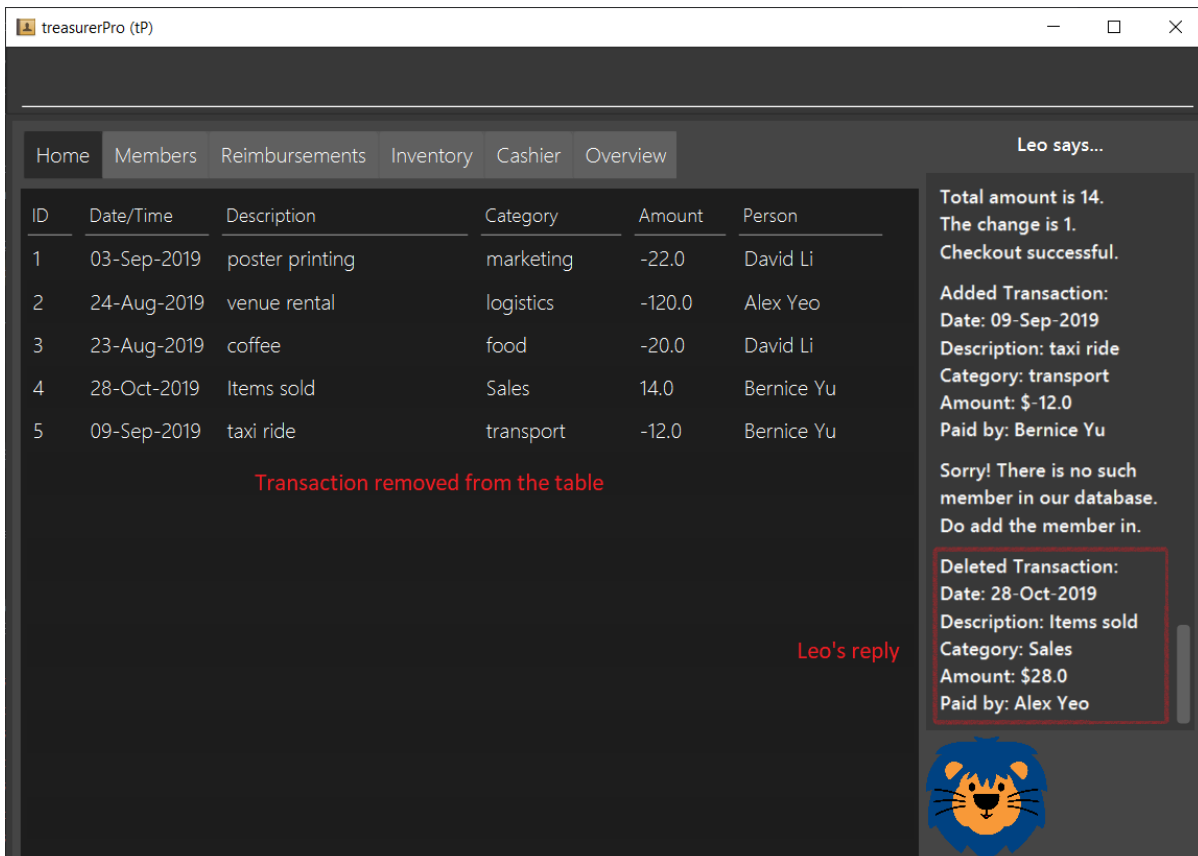


Figure 6. Screenshot of successful user input for delete by ID command in Home tab

- Steps for Deleting By Person:
 1. Type the command with the person's name for all corresponding transactions related to that person to be deleted as shown in the screenshot below:

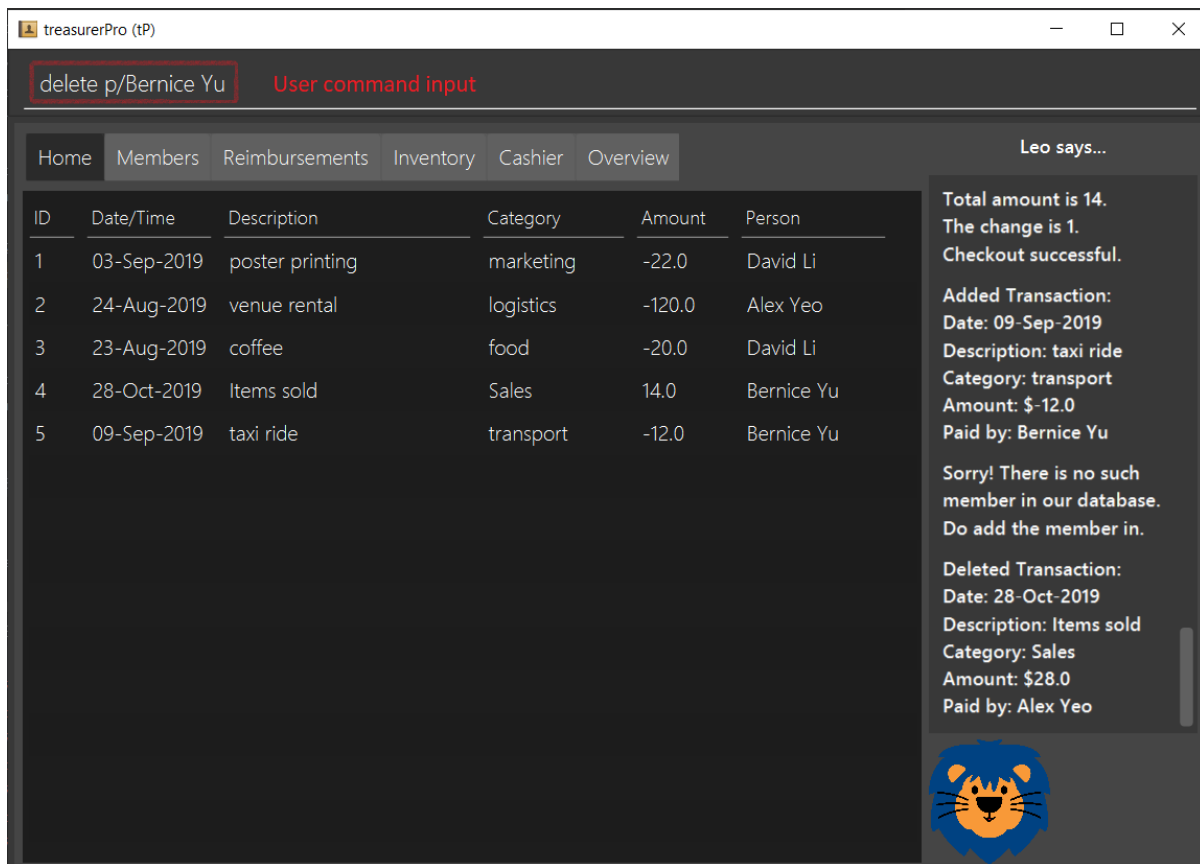


Figure 7. Screenshot of user input into command box for delete by ID command in Home tab

2. Enter the command.

Leo will respond with the successful deletion and the transactions will be removed from the table as shown below:

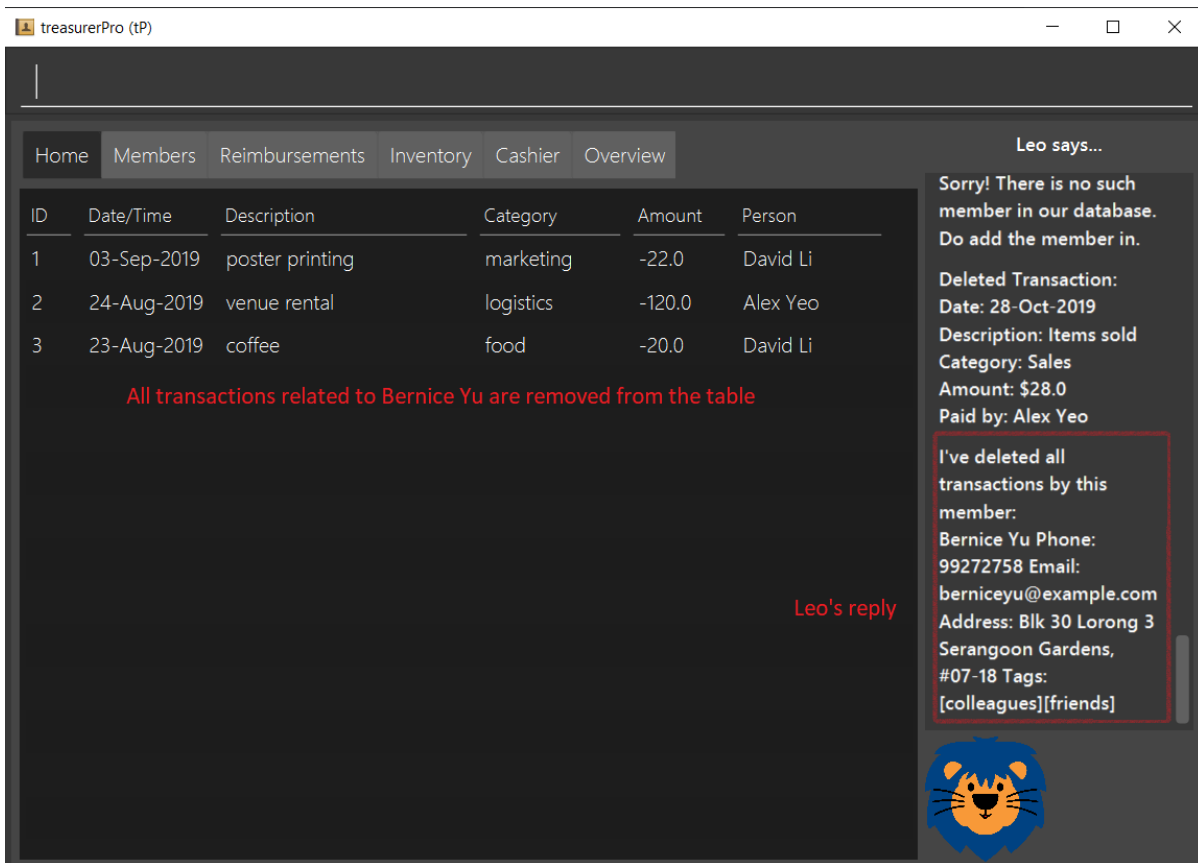


Figure 8. Screenshot of successful user input for delete by person command in Home tab

If the person inputted does not have any transactions related to him or her, Leo will also respond to inform you.

3.3. Edit a Transaction in the Table

This command helps to edit an existing transaction record in a table with different parameters.

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



The fields above can vary and be in different 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 edit with the new parameters to be changed to as shown in the screenshot below:

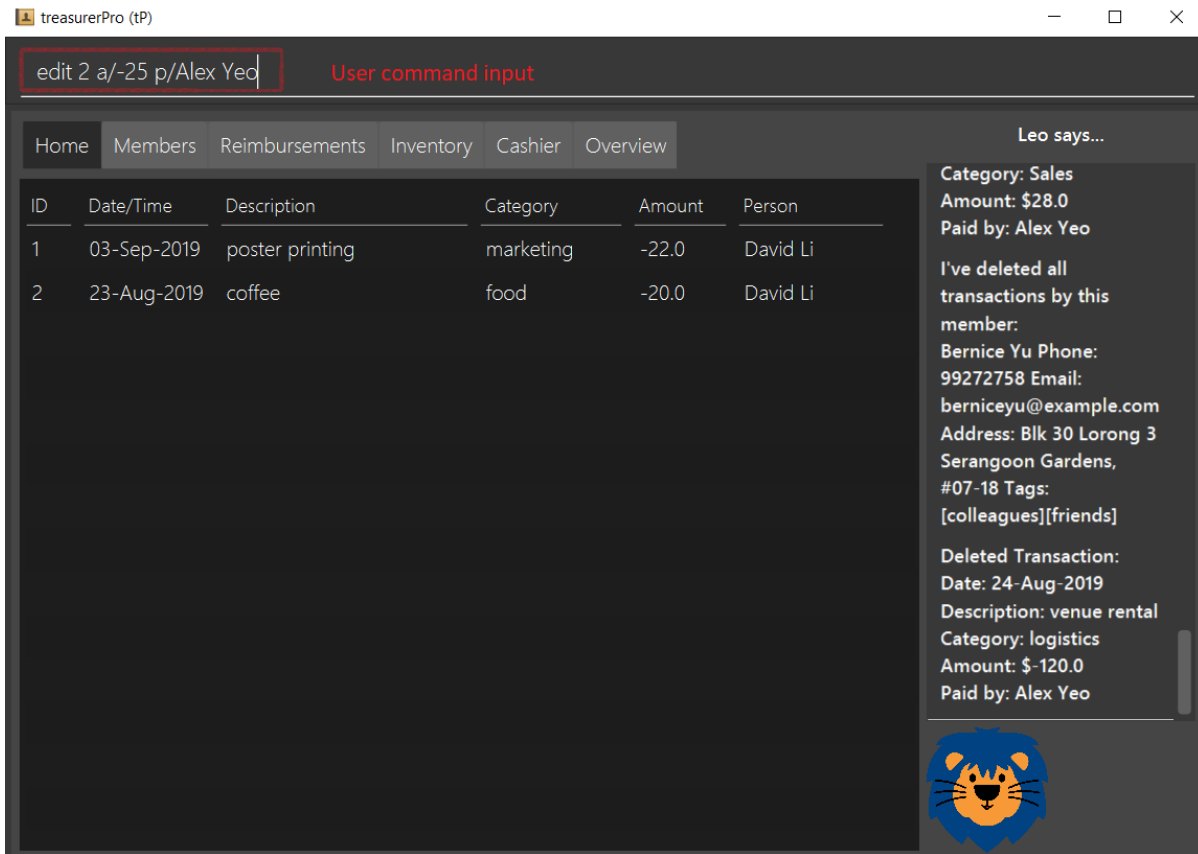


Figure 9. Screenshot of user input into command box for edit command in Home tab

2. Enter the command. Leo will respond with the successfully edited transaction and the transactions will be shown in the table as shown below:

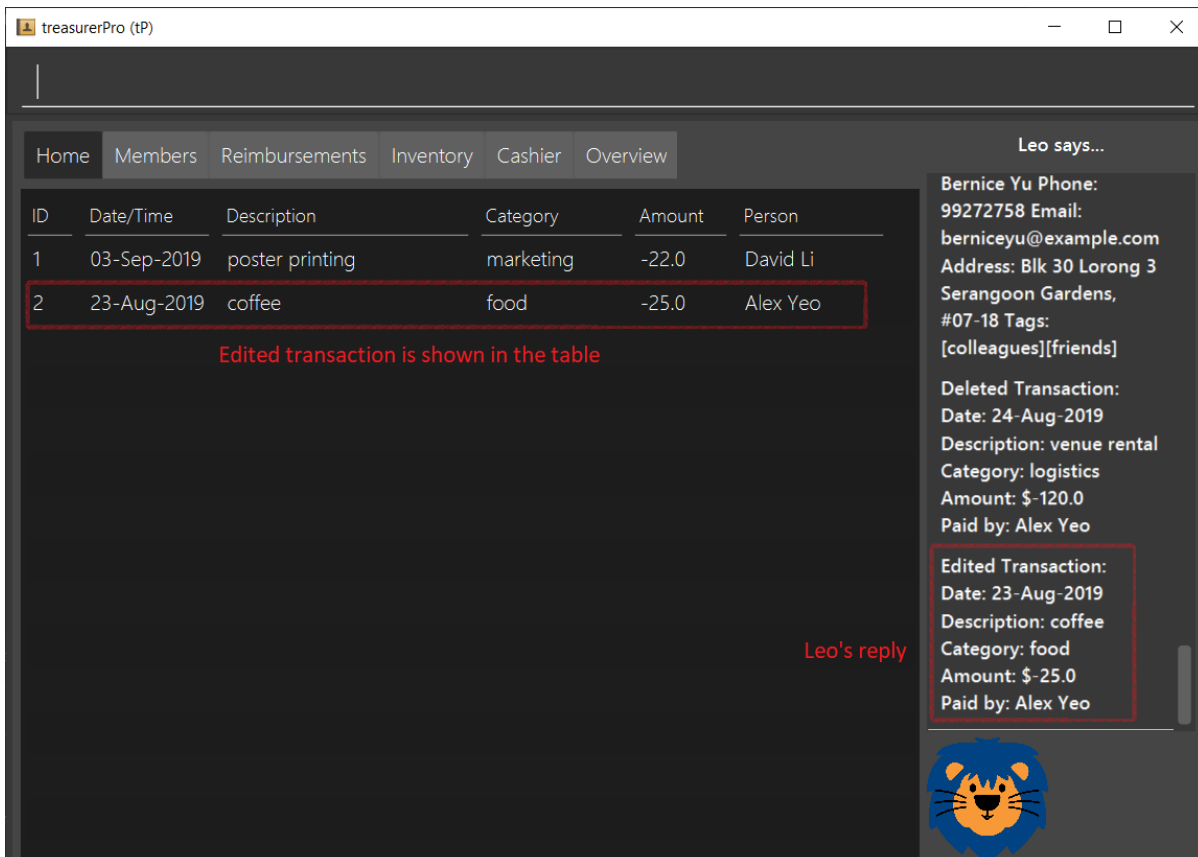


Figure 10. Screenshot of successful user input for edit command in Home tab

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](#).

3.4. Sort Transactions in the Table

This command helps to sort the table of transactions into specific orders for viewing and for carrying out subsequent commands.

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



The undo sort command will allow the user to view the table of transactions in the order of when the current session of this application was just opened, not when the transactions was first entered.

3.5. Find Transactions that Match Keywords:

This command helps to find transactions that match the entered keywords for viewing and for subsequent commands to be inputted based on the filtered table of transactions shown.

- Command Format: `find KEYWORDS`



The keyword can be one or more words. An entire keyword must match a word in any of the parameters of the transaction intended to be found and shown in the table. If the keyword is just a part of the word in the transaction, the transaction will not be shown in the table after the command is inputted.

- Examples:
 - `find Alex Yeoh`
 - `find Alex`
- Steps: Shown below is the table of all transactions:

treasurerPro (tP)

Home Members Reimbursements Inventory Cashier Overview

Leo says...

ID	Date/Time	Description	Category	Amount	Person
1	03-Sep-2019	poster printing	marketing	-22.0	David Li
2	23-Aug-2019	coffee	food	-8.0	Alex Yeo
3	23-Aug-2019	tea	food	-10.0	Alex Yeo
4	23-Aug-2019	fairy lights	decorations	-30.0	Bernice Yu
5	23-Aug-2019	biscuit	food	-14.0	Alex Yeo
6	23-Aug-2019	photo props	deco	-32.0	Bernice Yu
7	23-Aug-2019	flyers printing	marketing	-25.0	David Li
8	23-Aug-2019	video filming	marketing	-55.0	Alex Yeo
9	23-Aug-2019	photographer	marketing	-20.0	David Li
10	23-Aug-2019	gift of appreciation	gift	-30.0	Alex Yeo
11	23-Aug-2019	poster printing	marketing	-25.0	Alex Yeo
12	23-Aug-2019	mahjong paper	stationary	-6.0	Bernice Yu
13	23-Aug-2019	markers	stationary	-12.0	Bernice Yu
14	23-Aug-2019	pens	stationary	-15.0	Bernice Yu




Figure 11. Screenshot of all transactions in Home tab

1. Type the command with keyword(s) to find transactions as shown in the screenshot below:

treasurerPro (tP)

find marketing User command input

Home Members Reimbursements Inventory Cashier Overview

Leo says...

ID	Date/Time	Description	Category	Amount	Person
1	03-Sep-2019	poster printing	marketing	-22.0	David Li
2	23-Aug-2019	coffee	food	-8.0	Alex Yeo
3	23-Aug-2019	tea	food	-10.0	Alex Yeo
4	23-Aug-2019	fairy lights	decorations	-30.0	Bernice Yu
5	23-Aug-2019	biscuit	food	-14.0	Alex Yeo
6	23-Aug-2019	photo props	deco	-32.0	Bernice Yu
7	23-Aug-2019	flyers printing	marketing	-25.0	David Li
8	23-Aug-2019	video filming	marketing	-55.0	Alex Yeo
9	23-Aug-2019	photographer	marketing	-20.0	David Li
10	23-Aug-2019	gift of appreciation	gift	-30.0	Alex Yeo
11	23-Aug-2019	poster printing	marketing	-25.0	Alex Yeo
12	23-Aug-2019	mahjong paper	stationary	-6.0	Bernice Yu
13	23-Aug-2019	markers	stationary	-12.0	Bernice Yu
14	23-Aug-2019	pens	stationary	-15.0	Bernice Yu




Figure 12. Screenshot of user input into command box for find command in Home tab

2. Enter the command. Leo will respond with the number of matching transactions to the keywords and the table will only show the filtered transactions matching the keywords. This is shown in the screenshot below:

The screenshot shows the treasurerPro (tP) application interface. At the top, there is a navigation bar with tabs: Home, Members, Reimbursements, Inventory, Cashier, and Overview. The 'Home' tab is selected. Below the navigation bar, there is a table with the following columns: ID, Date/Time, Description, Category, Amount, and Person. The table contains five rows of data, all of which have the 'marketing' category. To the right of the table, there is a chat window titled 'Leo says...'. The chat window shows a message from 'Leo's reply' that says 'I've found 5 matching transactions!'. Below the chat window, there is a small icon of a lion's head. At the bottom of the table, there is a red text message that says 'Table shows all transactions that match the keyword 'marketing''.

ID	Date/Time	Description	Category	Amount	Person
1	03-Sep-2019	poster printing	marketing	-22.0	David Li
2	23-Aug-2019	flyers printing	marketing	-25.0	David Li
3	23-Aug-2019	video filming	marketing	-55.0	Alex Yeo
4	23-Aug-2019	photographer	marketing	-20.0	David Li
5	23-Aug-2019	poster printing	marketing	-25.0	Alex Yeo

Table shows all transactions that match the keyword 'marketing'

Leo's reply: I've found 5 matching transactions!

Figure 13. Screenshot of successful user input for find command in Home tab

3. Enter **back** to return to the table of all transactions or enter your next command to be executed. The edit command being inputted as the next command can be shown in the screenshot below:

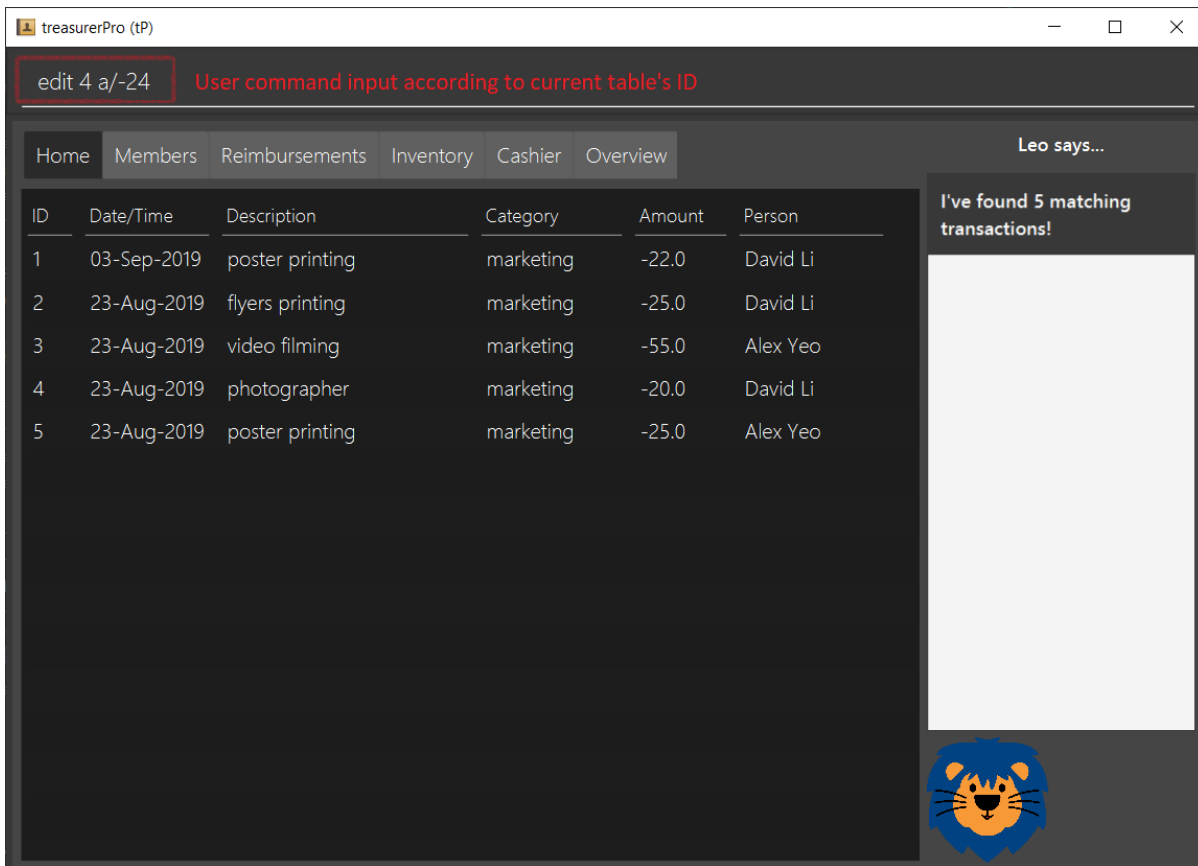


Figure 14. Screenshot of user input for edit command according to filtered table's ID in Home tab

The table will continue to show the filtered table with the transaction edited according to the command shown in the screenshot above.



For the add command, the table will automatically show the full list of all transactions. The rest of the commands will require the user to input **back** to return to the full list of all transactions.

The screenshot below shows the table after entering **back** which shows all the transactions in the table again:

treasurerPro (tP)						Leo says...	
Home Members Reimbursements Inventory Cashier Overview							
All transactions are shown again						I've found 5 matching transactions!	
ID	Date/Time	Description	Category	Amount	Person	Edited Transaction:	
1	03-Sep-2019	poster printing	marketing	-22.0	David Li	Date: 23-Aug-2019	
2	23-Aug-2019	coffee	food	-8.0	Alex Yeo	Description: photographer	
3	23-Aug-2019	tea	food	-10.0	Alex Yeo	Category: marketing	
4	23-Aug-2019	fairy lights	decorations	-30.0	Bernice Yu	Amount: \$-24.0	
5	23-Aug-2019	biscuit	food	-14.0	Alex Yeo	Paid by: David Li	
6	23-Aug-2019	photo props	deco	-32.0	Bernice Yu		
7	23-Aug-2019	flyers printing	marketing	-25.0	David Li		
8	23-Aug-2019	video filming	marketing	-55.0	Alex Yeo		
9	23-Aug-2019	photographer	marketing	-24.0	David Li		
10	23-Aug-2019	gift of appreciation	gift	-30.0	Alex Yeo		
11	23-Aug-2019	poster printing	marketing	-25.0	Alex Yeo		
12	23-Aug-2019	mahjong paper	stationary	-6.0	Bernice Yu		
13	23-Aug-2019	markers	stationary	-12.0	Bernice Yu		
14	23-Aug-2019	pens	stationary	-15.0	Bernice Yu		

Figure 15. Screenshot of user input for back command after entering find command in Home tab

(end of extract)

3.6. 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 Home Tab

This tab will help to show records of individual transactions from miscellaneous spending, revenue from sales and cost of buying items to sell. Each transaction will require an input of its date, description, category, amount and member that is accountable for it.

Revenue from each cashier checkout will also be automatically inputted as a transaction with positive amount in this tab with the person being the cashier. The inputted transactions that corresponds to a spending with a negative amount will be tabulated for each member in the reimbursement tab to keep track of reimbursements.

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](#):

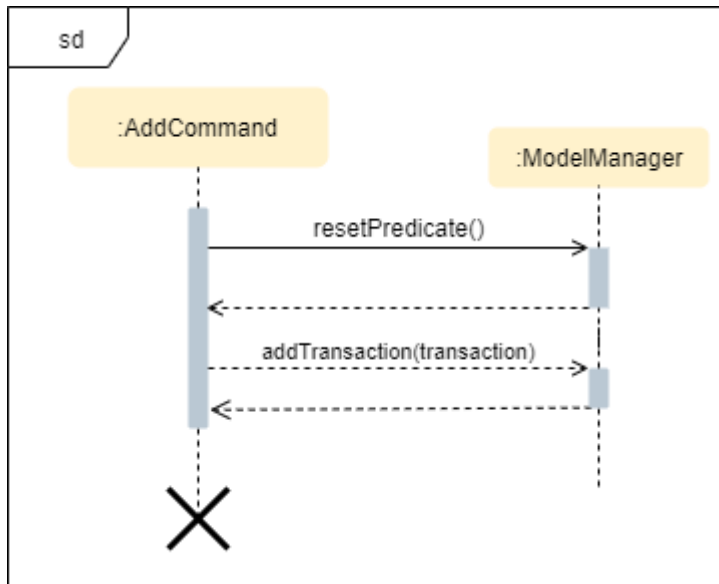


Figure 10. Sequence Diagram of Add Command in Home Tab (transaction package)

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:

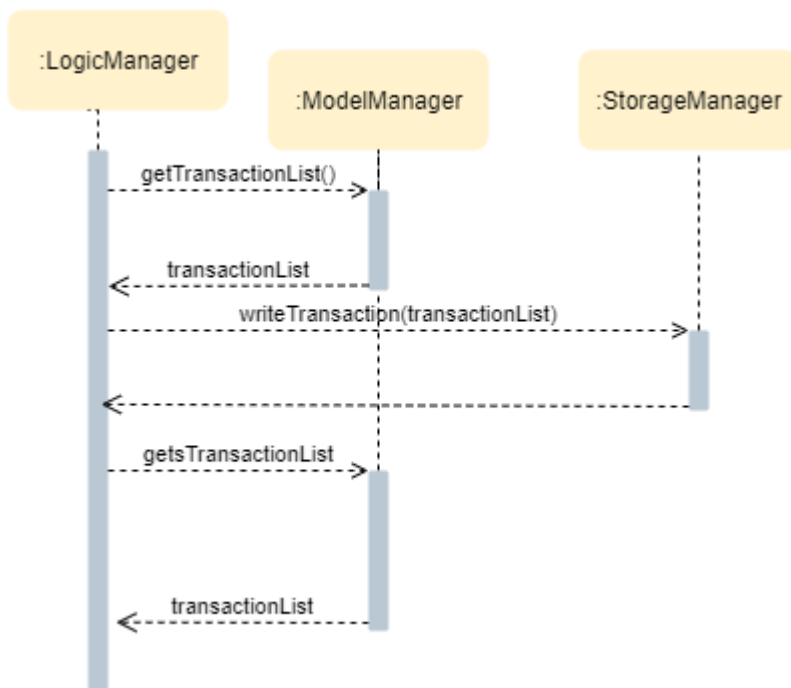


Figure 11. Sequence Diagram of updating the transaction list in Home Tab (transaction package)



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



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:

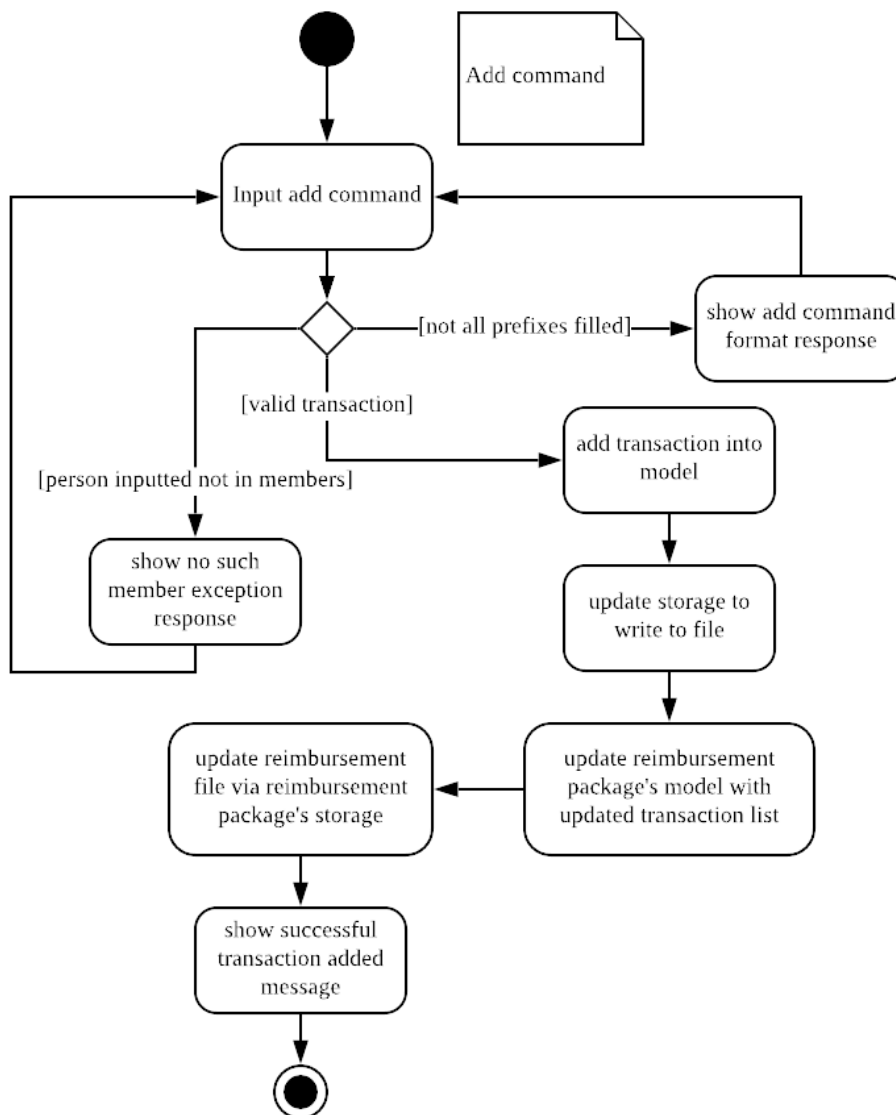


Figure 13. Activity Diagram of Add Command in Home Tab (transaction package)

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:

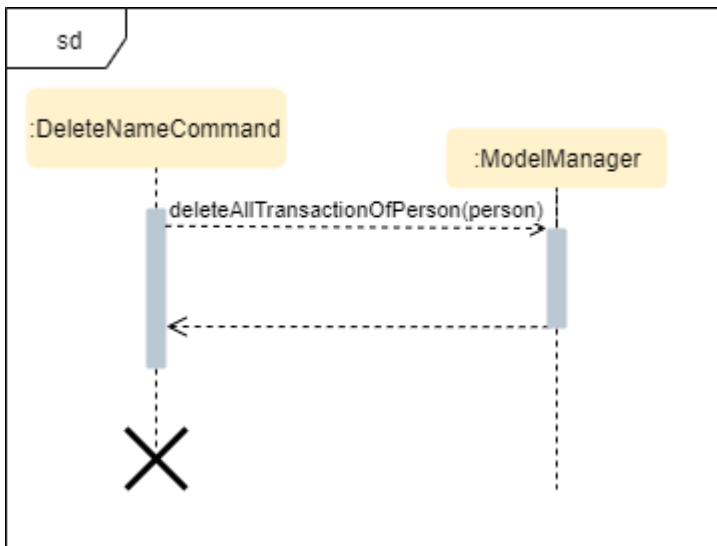


Figure 14. Sequence Diagram of Delete Command in Home Tab (transaction package)

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

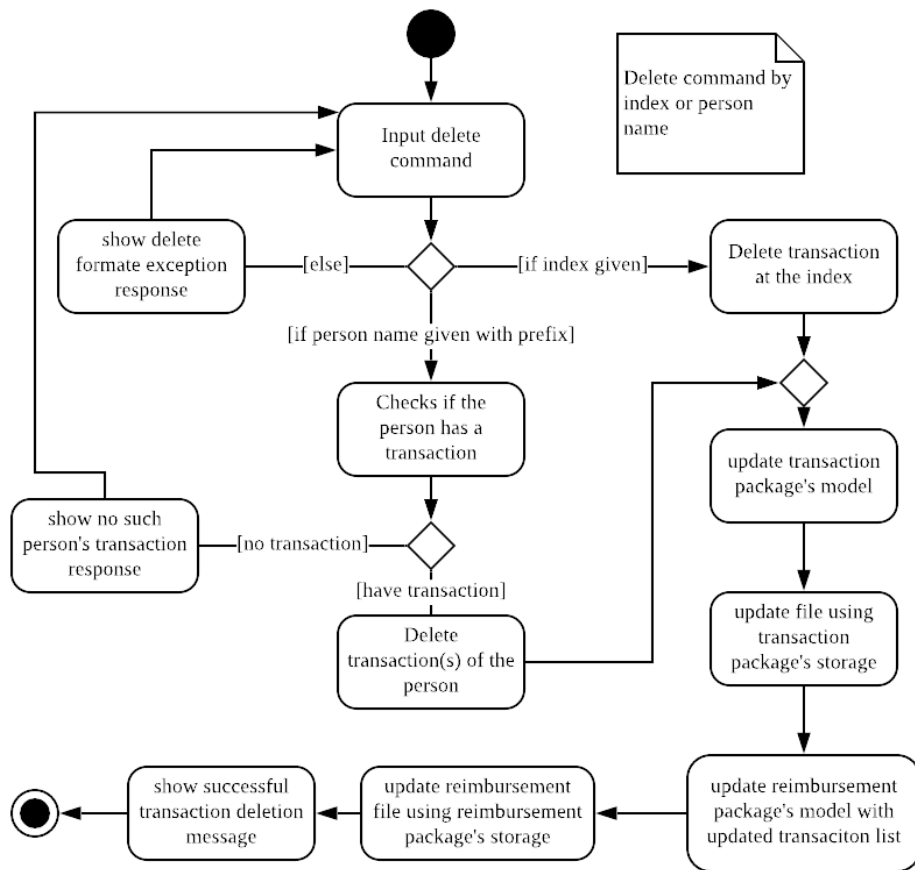


Figure 15. Activity Diagram of Delete Command in Home Tab (transaction package)

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, response on other incorrect inputs will also be shown. When a successful deletion is done, a response message is shown as well.

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:

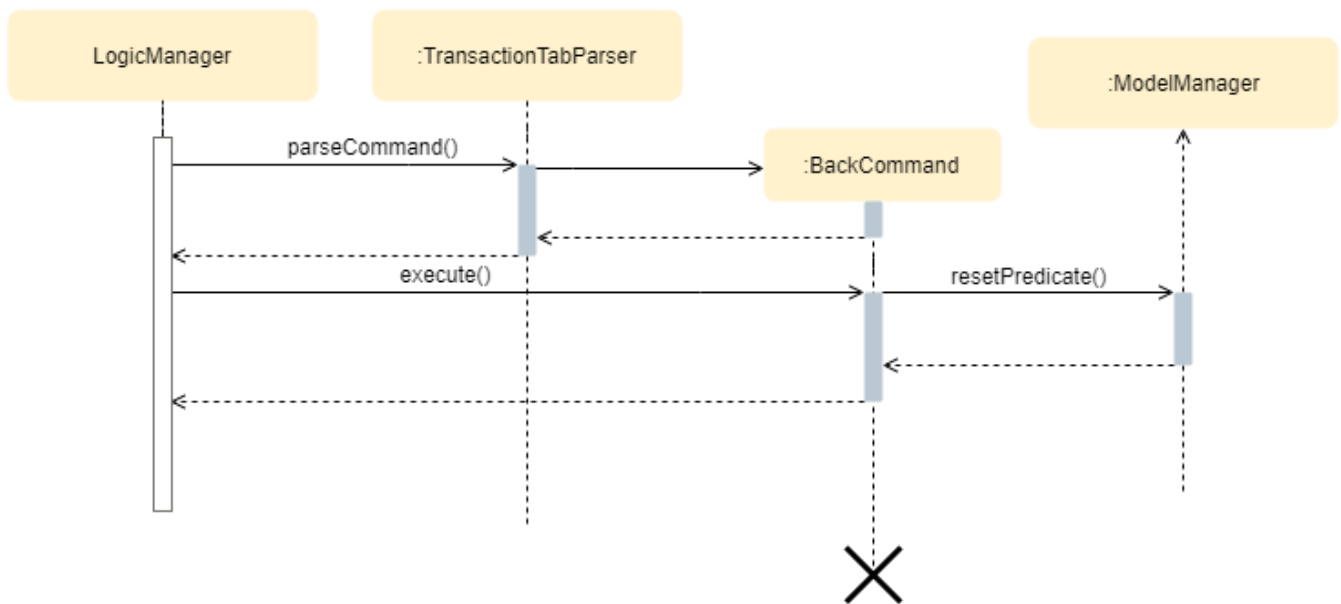


Figure 16. Sequence Diagram of Back Command in Home Tab (transaction package)

(end of extract)

4.2. Proposed enhancement for v2.0

Glossary

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

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.

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

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.