Hon Hao Chen - Project Portfolio

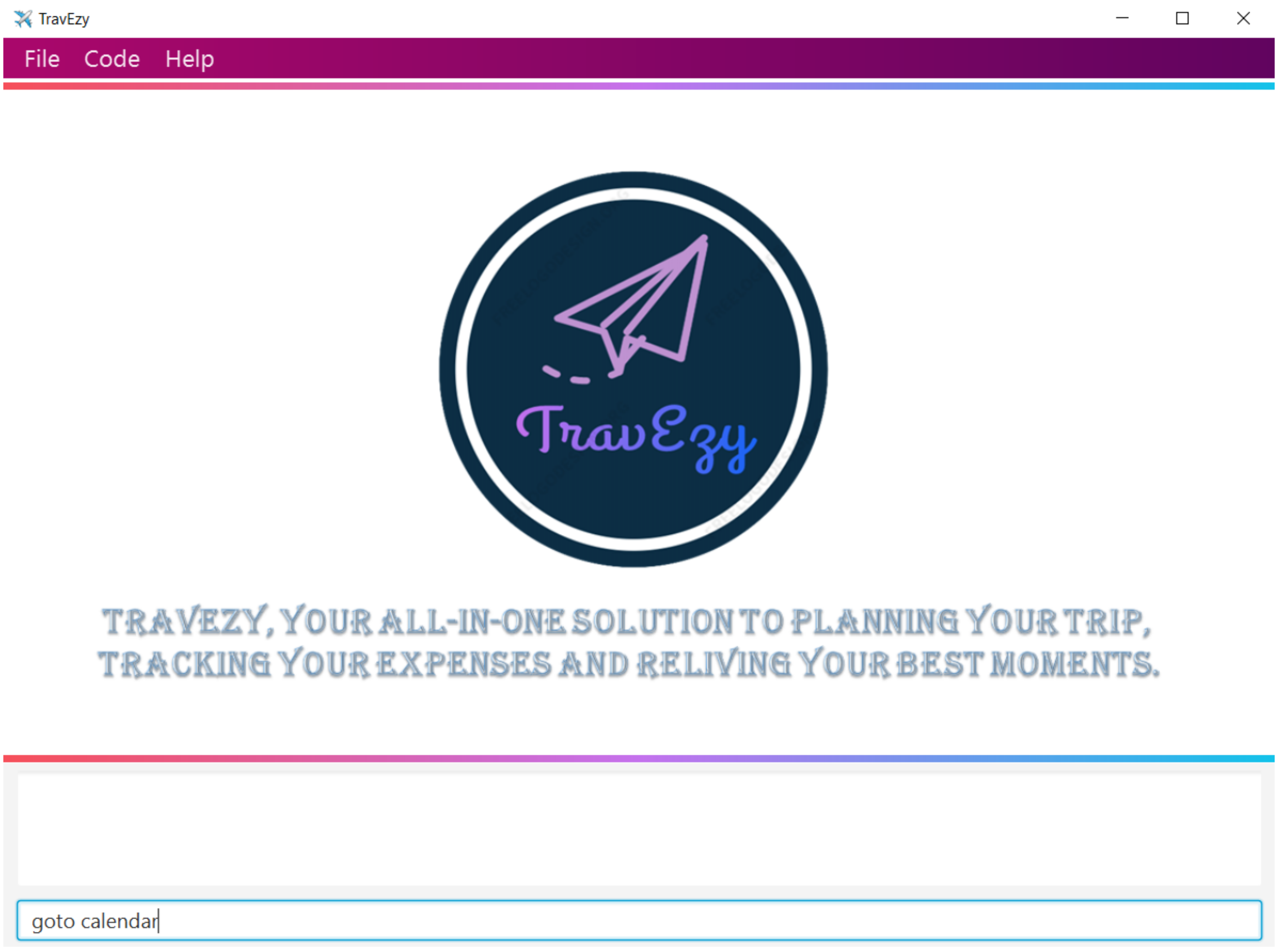
PROJECT: TravEzy

About the project

My team of 4 software engineering students and I were tasked with enhancing a basic Command Line Interface (CLI) desktop addressbook application for our Software Engineering project. Our constraint is that we have to target users who are able to type fast and prefer typing over other means of input.

With the stated constraint in mind, we choose to morph the given addressbook application into an travelling companion application which synthesis various kinds of features including financial tracker, diary, itinerary planner, contacts and achievement. This enhanced application allows users who love to travel to plan their trip beforehand, record down their expenses and write down their precious memories during travel. We ensured that the users have to type commands and instructions for all of the added features.

This is what our project looks like:


TravEzy
File
Code
Help
TravEzy
TRAVEZY, YOUR ALL-IN-ONE SOLUTION TO PLANNING YOUR TRIP, TRACKING YOUR EXPENSES AND RELIVING YOUR BEST MOMENTS.
goto calendar

My Role

My role was to design and write the codes for the Financial Tracker feature. The following sections illustrate my contributions in detail, as well as the relevant documentation I have added to the user and developer guides in relation to these contributions.

Summary of contributions

This section shows a summary of my contributed feature and other relevant contributions to the team project.

Major contributions

Financial Tracker:

- What is it?

  ```
  The Financial Tracker is an expense recorder whereby users can keep track of the
  records of their spending during their trips to different countries.
  ```

- Justification

  ```
  Our target users which we have chosen are NUS students who love to travel. During
  our users analysis, we realised that students nowadays are very likely to overspend
  their budget during
  travelling. Therefore, this financial tracker feature allows our target users to
  keep a record of their spending in order to remind them of their budget.
  ```

- More to it

  ```
  Users could also generate an overview of their expenses in an easy-to-look pie
  chart form!
  ```

Code contributed: 96, 76, 47

Minor contributions

Project management:

- Started implementing Ui components and link them to FXML. This provides skeletal codes for the rest of my teammates to adapt with.
- Modified command Command class to be generic so that the other Command classes in other features can take in their own Model classes.

Enhancements to existing features:

- Updated the Main page’s Ui:

Community:

- PRs reviewed (with non-trivial review comments):
- Reported bugs and suggestions for other teams in the class:

Contributions to the User Guide

_Given below are sections which I contributed to the User Guide. They showcase my ability to write documentation targeting end-users. I shall focus on 3 relevant commands here which are the `help`, `sort` and `summary` commands.

Switching between countries: `switch`

Currently in somewhere else? Switch to that expense list instead!

Format:

`switch COUNTRY`

TIP Do realise that you can always use the drop down menu instead!

WARNING You can only type in countries which are only listed from the *Countries dropdown box*

Example:

```
switch Japan
```

Step by step:

Step 1. Type `switch Japan` in the *Command box* and press *Enter*.

[FinancialTrackerSwitch1] | *FinancialTrackerSwitch1.png*

Step 2. The *Result box* will display the message "Expense list switched".

Step 3. Now your expense list inside the *Expense list panel* has been switched to that which is in Japan!

[FinancialTrackerSwitch2] | *FinancialTrackerSwitch2.png*

Sort out your expense list: `sort`

The default sorting way of the expense list is not your thing? Just sort your expense list according your needs!

TIP All of the sorting is done in reversed order :)

Format:

`sort CRITERIA`

Where `CRITERIA` can be `amount`, `date`, `time`, `type` and `default`

Example: to sort by amount

```
sort amount
```

Step by step:

Step 1. Notice the amount field in each expenses are currently not in order.

Step 2. Now, type `sort amount` in the *Command box* and press *Enter*.

[FinancialTrackerSort1] | *FinancialTrackerSort1.png*

Step 3. The *Result box* will display the message "Expense List sorted!".

Step 4. Now all your expenses are sorted in descending order of your amount!

[FinancialTrackerSort2] | *FinancialTrackerSort2.png*

Generate an overview of your spending: `summary`

Do you ever have difficulty summarising your spending? Afraid not! You can view you expenses statistics easily!

Format:

`summary`

Example:

```
summary
```

Step by step:

Step 1. Type `summary` in the *Command box* and press *Enter*.

[FinancialTrackerSummary1] | *FinancialTrackerSummary1.png*

Step 2. The result box will display the message "Currently viewing the Summary Window".

Step 3. As you should have noticed, the Summary Window has been popped out showing you statistics of your expenses in a nice-looking pie chart and bar chart form!

[FinancialTrackerSummary2] | *FinancialTrackerSummary2.png*

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.

Financial Tracker

The Financial Tracker feature in TravEzy allows users to keep track of their financial expenses with appropriate categorized expenditure by different countries.

Architecture:

The Architecture Diagram given below explains the high-level design of the financial tracker feature.

[FinancialTrackerDiagram] | *FinancialTrackerDiagram.png*

Figure 1. Architecture Diagram of the Financial Tracker feature

Inputs given by the user are channeled from the `textUI` and handled by `Logic` before different commands are formed which execute on the `Model` and updates the `FinancialTracker` object which contains the expense list.

The `textUI` and `Log Centre` stem from the common package of the main TravEzy application while `Logic`, `Model,` and `Storage` all have similar architecture design with the other features (Calendar, Itinerary, etc.).

Financial Tracker Class Diagram:

[FinancialTrackerClassDiagram] | *FinancialTrackerClassDiagram.png*

Figure 2. Class Diagram of the Financial Tracker feature

The `Model` stores the `FinancialTracker` data which includes several `ExpenseList` each associated with a country that keeps track of all the expenses.

`Financial Tracker` exposes an unmodifiable `ObservableMap<String, ExpenseList>` that can be observed. The UI is bound to the `Model` whereby the `Model` returns an expense list associated with the current `Country` of the `Financial Tracker` and updates when the querying `Country` changed.

NOTE	Each time a new `Country` is selected from the Countries dropdown box, the respective expense list will be returned for that specific country, this implementation is to ease categorising and sorting.

Sequence Diagram example:

The sequence diagram below shows the sequence of events that will take place when a user calls `Summary` command.

[FinancialTrackerSummaryCommandDiagram] | *FinancialTrackerSummaryCommandDiagram.png*

Figure 3. Sequence Diagram of the Financial Tracker feature

The sequence of events are very likely to those of the other commands in Financial Tracker feature.

Activity Diagram example:

The activity diagram below shows how `edits` command works.

[FinancialTrackerActivityDiagram] | *FinancialTrackerActivityDiagram.png*

Figure 4. Activity Diagram of the Financial Tracker feature