

Wang Tianyi - Project Portfolio for *TravelPal*

Portfolio Overview

This project portfolio page documents my contribution to the team project *TravelPal*. The page includes an overview of the project, summary of the features I have implemented and other contributions, such as User Guide and Developer Guides.

About our Project

The application developed by our team, *TravelPal*, is an all-in-one trip planner which provides frequent leisure travellers with a highly customisable, intuitive and integrated trip management system. We aim to ease pre-trip planning process with features such as trip itinerary and expense manager, and offer a platform for users to record their trip highlights with our diary feature.

As *TravelPal* is for those who prefer using the keyboard over a mouse, it uses a Command Line Interface (CLI) for keyboard input, but it also provide alternative input with mouse. As for output, it is displayed in a Graphical User Interface (GUI) - which is a display that includes panes, menus and message boxes. We believe such implementation makes our application more user-friendly.

In our project, my role was to implement the *Expense Manager* and *Currency* feature. The following sections will elaborate on these enhancements.

Summary of contributions

This section encompasses an overview of my contribution to the *TravelPal* application, including the summary of my code implementation and how they add value to the application in terms of functionality and user experience.

Major enhancements: Implementation of the *Expense Manager* feature

- **What it does:** The *Expense Manager* allows users to micromanage a list of expenses. User can choose to manually add a new expense to a day or include an expense when adding an event in trip itinerary. All expenses will be displayed according to the days, and a summary of the daily expenses and the total expenses for the trip will be displayed.
- **Justification:** This is one of the most useful features for pre-trip planning. A potential user may use the manager to keep track of their estimated expenses, compare it with the budget and adjust their schedules accordingly.
- **Highlights:** This enhancement requires understanding of other features as the expenses needs to be updated whenever there is a change in trip events. The expenses also need to be grouped under each days. Therefore, I had to make sure that the interactions between event class, day list and expenses are bug-free and efficient.

- **Credits:** I used the concepts and logic from the existing code of address book to implement my features.

Second enhancements: Implementation of the *Currency* feature.

I added a *Currency* feature which allows users to create and select customised currency with exchange rate.

- **What it does:** The *Currency* feature allows users to add and select currencies with customised currency name, symbol and exchange rate. When a currency is selected, all the monetary valued will be displayed in that currency.
- **Justification:** Given the limited domestic tourism size, currency conversion is very relevant to travellers in Singapore. International travellers may want to view the expenses in local currency so that our trip manager can be more intuitive and easy to use.
- **Highlights:** To help users type currency symbols more efficiently, I added a list of preset currency symbols with index. This feature allows users to simply enter an index, which is automatically converted into currency symbols. The implementation demands effective handling of javafx.

Other Contributions

- Code contributed: [[RepoSense code](#)][[Github pull requests](#)]
- Idea generation:
 - Contributed main idea of *TravelPal* and its various sub-features (itinerary, diary, expenses, etc.) during project inception. (see [travelpal idea generation google docs](#))
- Enhancements to existing features:
 - Wrote test cases for *Expense Manager* and *Currency* related classes.
 - Updates UI to make it more user friendly [#160](#)
- Project management:
 - Reviewed and approved several PRs
 - Updated AboutUs page [#18](#)

Contributions to the User Guide

This section displays the section of User Guide relating to the usage of *Expense Manager* feature and *Currency* customisation, created and edited by me.

Expense Manager

Introduction

TravelPal's *Expense Manager* is an integrated expense planning and management system. It keeps track all the expenses generated in your trip, and provides an intuitive overview of daily and total

expenses and budgets. Gui alternatives are available for executing the same operations as command line input.

This section of the user guide explains how to view and manage your expenses using *Expense Manager*.

User Interface Overview

Shown below is the landing page of the *Expense Manager*:

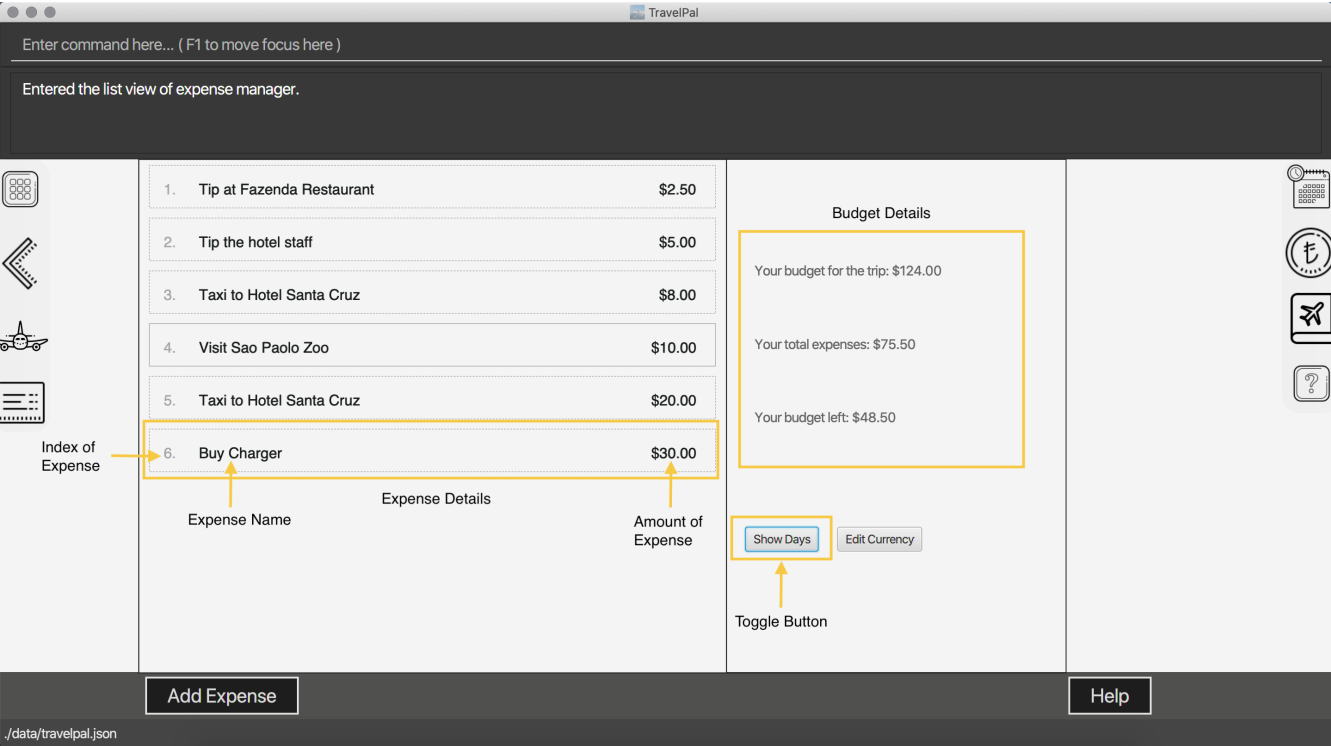


Figure 1. Overview of Expense Manager user interface

To toggle the display of expenses between list view and days view, use the command **showdays** or **showlist**. Alternatively, you may click on the *toggle button* on the page.

The following screenshot is the days view of the *Expense Manager*:

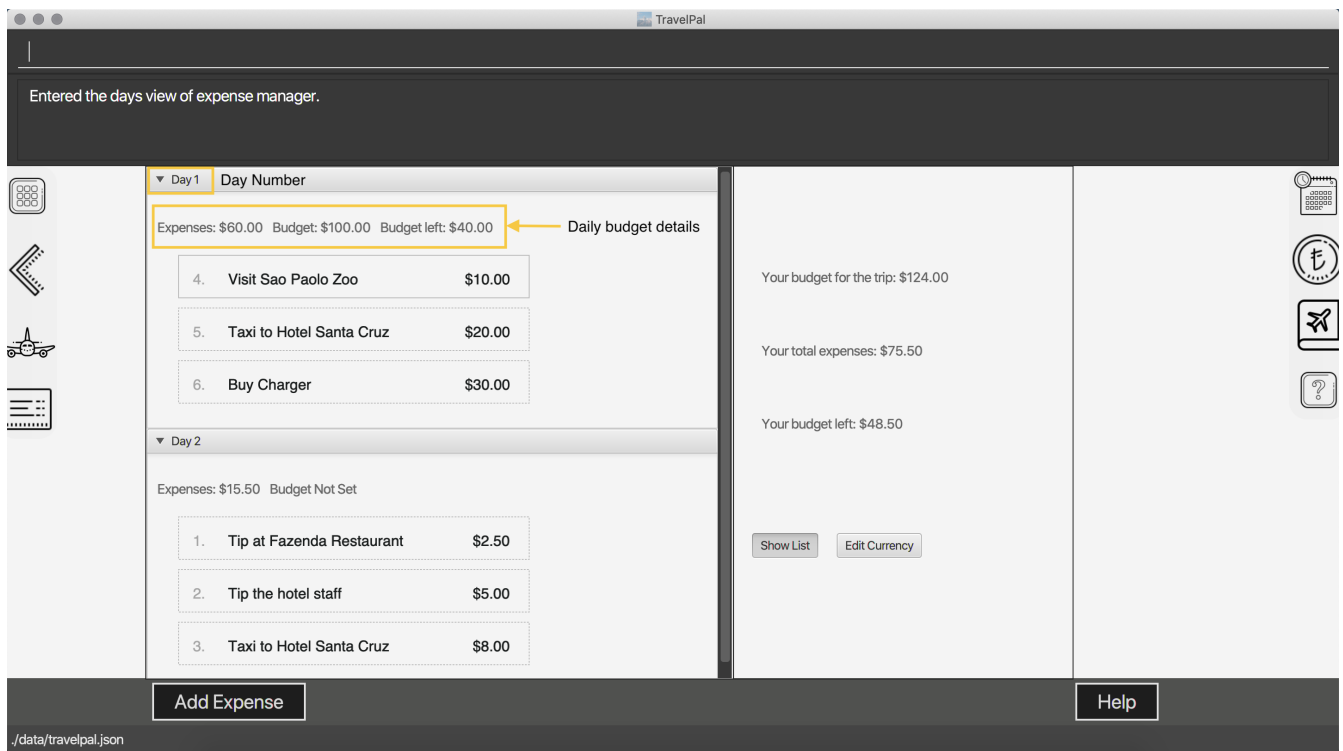


Figure 2. Expense Manager user interface showing the daily expenses and budget summary

Expenses are connected to bookings/events to automatically update the current known expense for any date/trip/event. There are two types of expense:

1. Planned expense (auto-generated from event)
2. Miscellaneous expense (can be created and deleted)

Commands

On the *Expense Manager* page, the following command are available:

- **create**: creates a expense, can also be accessed by clicking the **Add Expense** button.
- **edit <index of expense>**: edit an expense, this command bring up the expense setup page.
- **delete <index of expense>**: delete an expense, note that only miscellaneous expenses can be deleted.
- **showdays**: enter the days view of expense manager, the expenses will be grouped according to the days they belong to.
- **showlist**: enter the list view of expense manager, the expenses will be shown in one list.
- **goto <index of expense>**: go to the event page containing the event associated with this expense.
- **sort name**: sort the expenses according to the name lexicographically. Enter the command again to sort in reverse order.
- **sort amount**: sort the expenses according to the amount of expense in ascending order. Enter the command again to sort in descending order.
- **currency**: enter the *Currency* page of TravelPal, can also be accessed by clicking on the *Edit Currency* button

NOTE

delete command can only be used on miscellaneous expense, however, you may delete the event associated with a planned expense, which will delete the expense as well. **goto** command is only for planned expense associated with an event.

Expense Setup

Expense Setup creates/edits properties of a specified expense. To access the *Expense Setup* page, use **create** or **edit** `<index of expense>` command on *Expense Manager* page. Shown below is an screenshot of the page:

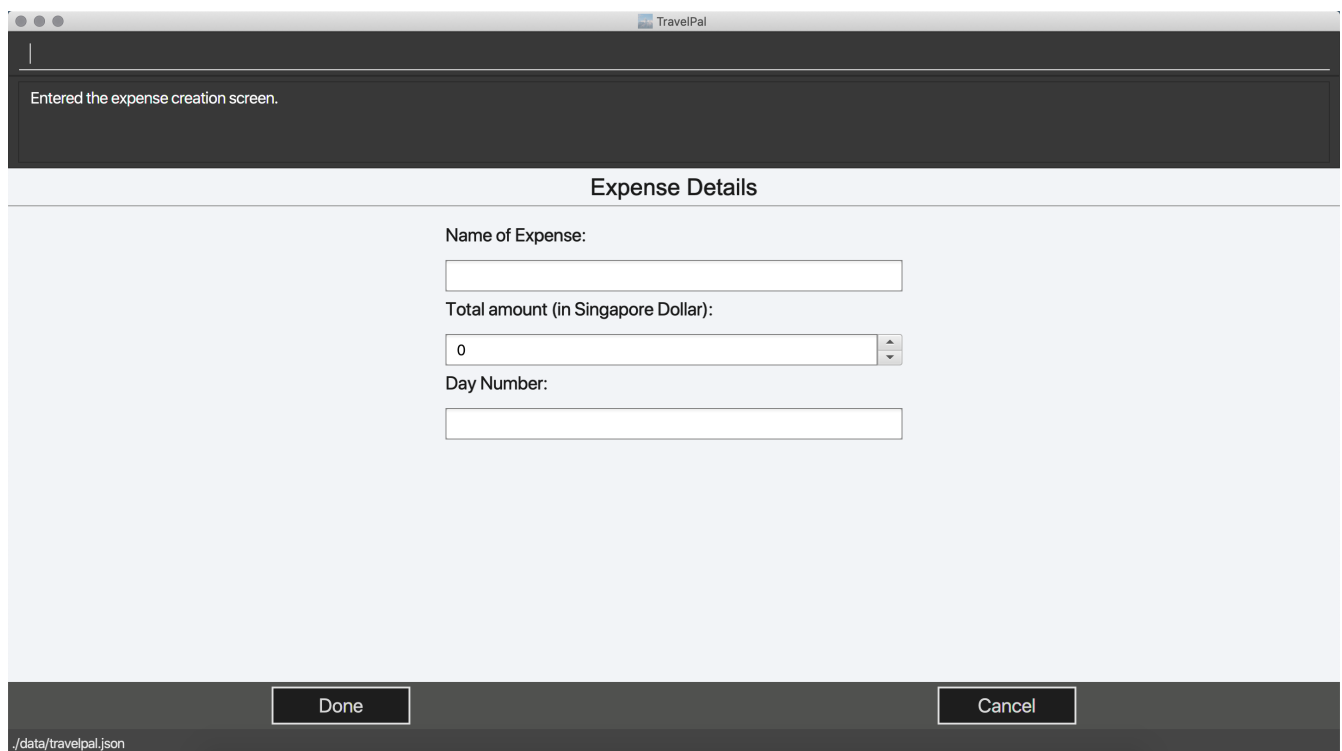


Figure 3. *Expense Setup* page user interface

It is necessary for expense to contain a name, an amount and a day number.

The following commands are available on *Expense Setup* page.

- **edit** `<prefix>/<value>` `...` : edit the field of the expense to be created/edited.
- **done** : confirm and commit the changes, go back to the expense manager page.
- **cancel** : go back to the expense manager page without committing the changes.

The prefixes refer to editing each fields as follows:

- **n/** Name of the expense
- **b/** The amount of expense, in Singapore dollars.
- **dn/** The day number the expense belongs to.

NOTE

For planned expense linked to an event, the **name** and **day number** fields are not editable. However, you may edit the name of the corresponding event, this will also update the name of the planned expenses.

Example Usage for `edit` command:

To add an expense with the name *Miscellaneous Expenses* of \$10.5 SGD to day 2, use the following command:

```
edit n/Miscellaneous Expenses b/10.5 dn/2
```

Currency

Introduction

On *Currency* page, you can add and select currencies with customised currency name, symbol and exchange rate. When a currency is selected, all the monetary valued will be displayed in that currency

User Interface Overview

Shown below is a screenshot of the *Currency* page:

The screenshot displays the 'Currency Manager' interface. At the top, a message states 'Entered the currency creation screen.' Below this, there are two main sections. The left section contains input fields for 'Name of Currency' and 'Symbol of Currency'. Under the 'Symbol of Currency' field, there is a grid of 'Preset currency symbols' with buttons for \$ (1), £ (2), ¥ (3), € (4), ₱ (5), ₹ (6), and ₩ (7). Below these buttons, a text input field shows 'SGD \$1.00 =' and a numeric input field with '0'. The right section, titled 'Currency in use', lists two currencies: '1. RMB (¥)' with an exchange rate of '5.02' and '2. SGD (\$)' with an exchange rate of '1.00'. Arrows point from labels to these elements: 'Selected currency' points to the RMB entry, 'index' points to the number '1', 'name of currency' points to 'RMB (¥)', and 'exchange rate to \$1 Singapore Dollar' points to '5.02'. At the bottom, there are two buttons: 'Add Currency' and 'Return'. The footer shows the file path '/data/travelpal.json'.

Figure 4. Overview of *Currency Manager* user interface

The left half of the *Currency Manager* page consists of editable text fields for creating a new currency.

Under the **Symbol of Currency** section, you can find *preset currency symbols*, in which the most commonly used currency symbols are indexed. You can select a currency by entering the index in place of the actual symbol, or just by clicking on the button. You can also manually input other currency symbols.

On the right hand side of the page, the customised currencies are listed. *Singapore Dollar(SGD)* is pre-defined as the base currency. You may select or delete a customised currency.

Commands

The following commands are available on *Currency Manager* page:

- **select <index of currency>**: select the currency with the specified index as the currency in use.
- **delete <index of currency>**: select the currency with the specified index, note that the default Singapore dollar cannot be deleted.
- **edit <prefix>/<value> ...**: edit the fields of a new currency to be created.
- **add**: confirm and commit the changes, the newly added currency will be chosen as the currency in use, displayed in the currency list
- **return**: return to the expense manager.

NOTE

the **<value>** for editing the currency symbol can either be an integer representing the index of the preset currencies, or a non-numerical string with no more than 3 characters.

The prefixes refer to editing each fields as follows:

- **n/** name of the currency
- **s/** symbol of currency,
- **r/** exchange rate of the currency, using Singapore dollar as base for comparison.

NOTE

Singapore Dollar (SGD) is used as the default currency, it cannot be deleted.

Example Usage for edit command:

To add an currency with name *USD*, symbol \$ (*pre-set symbol with index 1*), and an exchange rate of 1 SGD : 0.74 USD, use the following command:

```
edit n/USD s/1 r/0.74
```

NOTE

See the [user guide](#) for this usage scenario, not included in this project portfolio page.

Contributions to the Developer Guide

This section displays the section of Developer Guide showing the logic, model and user interface implementation of the *Expense manager*, created and edited by me.

[Expense] Expense Manager

The *Expense Manager* is one of the main features of TravelPal, it maintains a list of `Expense` stored in an `ExpenseList`. *Expense Manager* is also capable of calculating and displaying budget, sorting expenses and toggling display options.

Aspect : Model

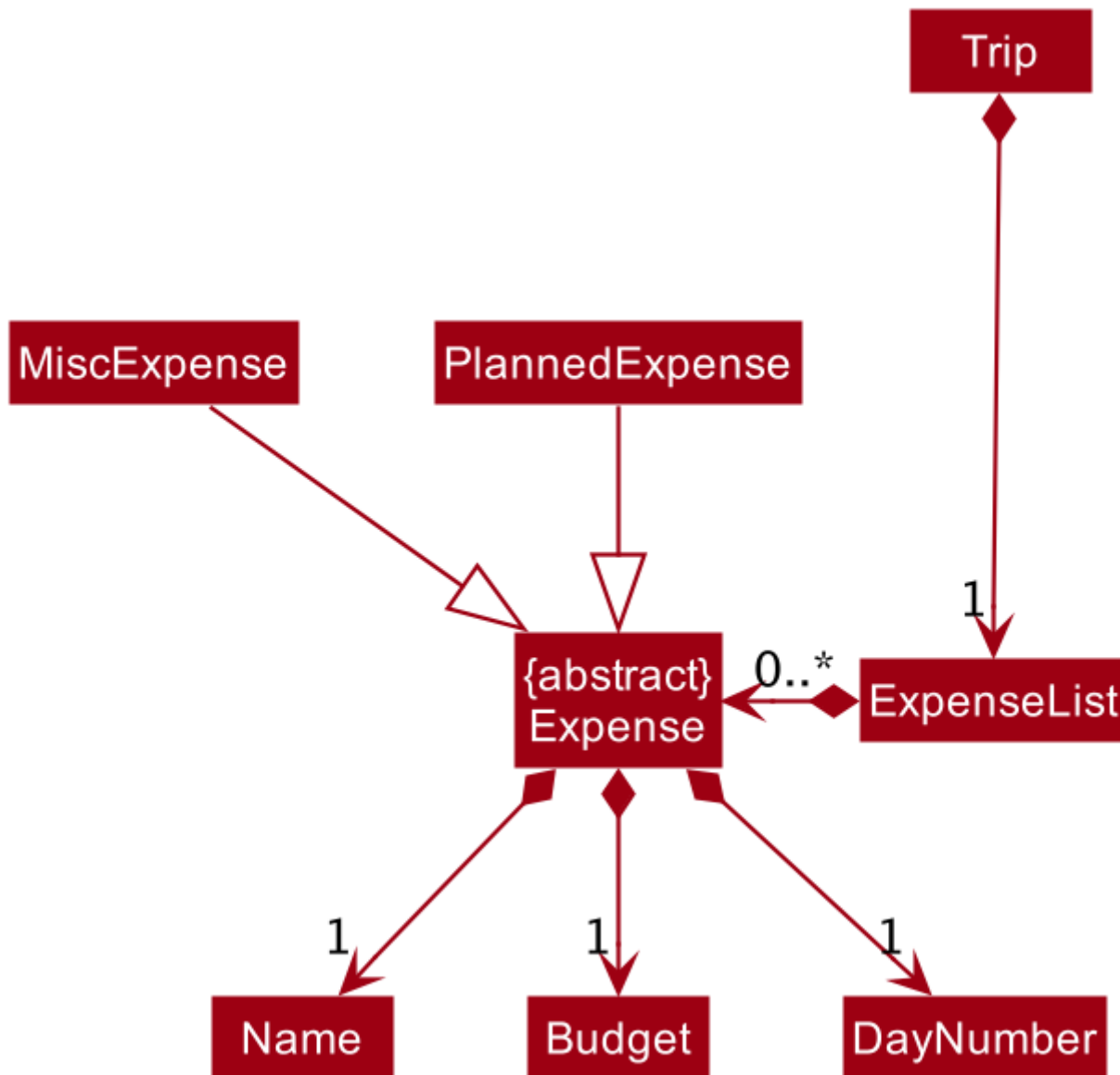


Figure 5. Class diagram showing the expense model

Expense

Expense is an abstract class storing expense model.

It has three compulsory fields, the *name of expense*, the *amount of expense*, and a *day number*. These fields are used to store information related to an expense. They are implemented as instance of class `Name Budget` and `DayNumber` respectively.

MiscExpense and PlannedExpense are the child classes extending from Expense class, they are used to

present the two types of expenses: *miscellaneous expense* and *planned expense*.

ExpenseList

`ExpenseList` is a class that stores the `Expense` models. It supports wrapper methods around the underlying `ObservableList` to facilitate the use in the logic components.

Aspect : UI

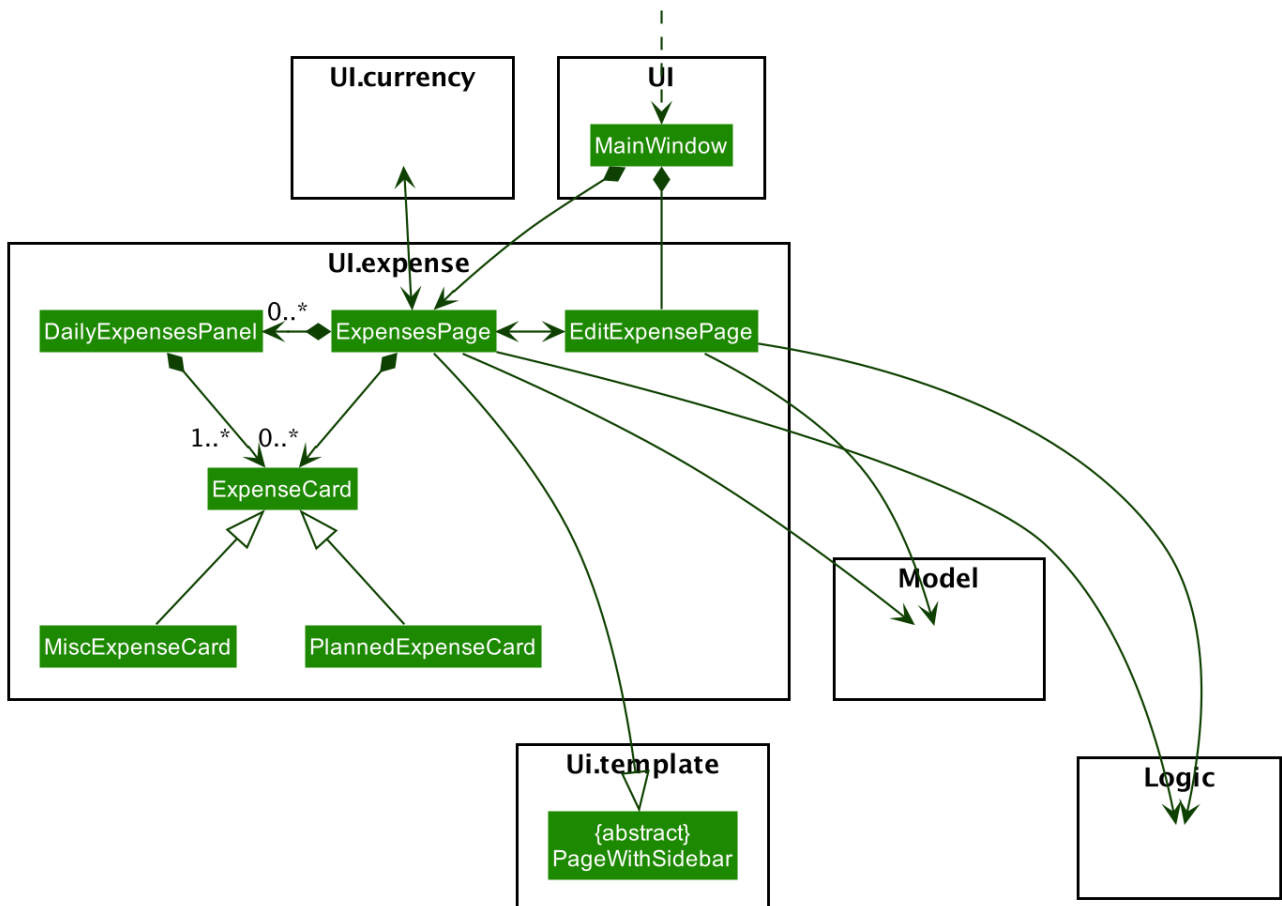


Figure 6. Class diagram showing the user interface of expense

NOTE

The `ExpensesPage` implements an `UiChangeConsumer` interface to facilitate the toggling between the *Days View* and *List View* of *Expense Manager*, which is not shown in the diagram above.

The UI of expense manager mainly consists of two `Page`: `ExpensesPage` and `EditExpensePage`.

`ExpensePage` is the component in charge of displaying expense and budget information. It has a list of `ExpenseCard`, a component that contains individual expense details. `ExpensePage` extends `PageWithSidebar` as it contains navigation bar that helps user to navigate between different features.

In `ExpensePage`, user can toggle between *Days View* and *List View*. In *Days View*, a list of `DailyExpensesPanel` is used to group `ExpenseCard` according to date.

`EditExpensePage` is the main page for creating and editing of expense. Both `ExpensesPage` and

`EditExpensePage` have access to `Model` and `Logic` of the application, for handling of stored data and parsing commands.

From `ExpensePage`, user can navigate to *Currency* feature of application through CLI or GUI.

Aspect : Logic

Create an expense

The creation of a new `Expense` is done in two ways:

1. The creation of a `PlannedExpense` is created when a new `Event` with a `Budget` is created. The execution happens in `DoneEditingEventCommand`.
 - When the `Name` or `Budget` field of `Event` is modified, a method call replaces the current `Expense` associated with the `Event` with an updated `Expense`.
2. The creation of a `MiscExpense` is done by calling `EnterCreateExpenseCommand`, which brings user to an *Expense Setup Page*.

Edit an expense

Editing of expense can be accessed from `ExpensePage`. The execution of command is handled by `ExpenseManagerParser` and the command accesses the model through `Model#getPageStatus()` method. The details of execution is similar to *Edit trip/day/event* feature (see [\[Edit-Trip\]](#))

Only the *amount of expense* field of a `PlannedExpense` can be edited with an `edit` command. When `done` command is executed after the *amount of expense* is edited, both the expense in `ExpenseList` and the `Event` will be updated.

The following sequence diagram shows the sequence of method call when `DoneEditCommand#execute(model)` is called in `LogicManager`.

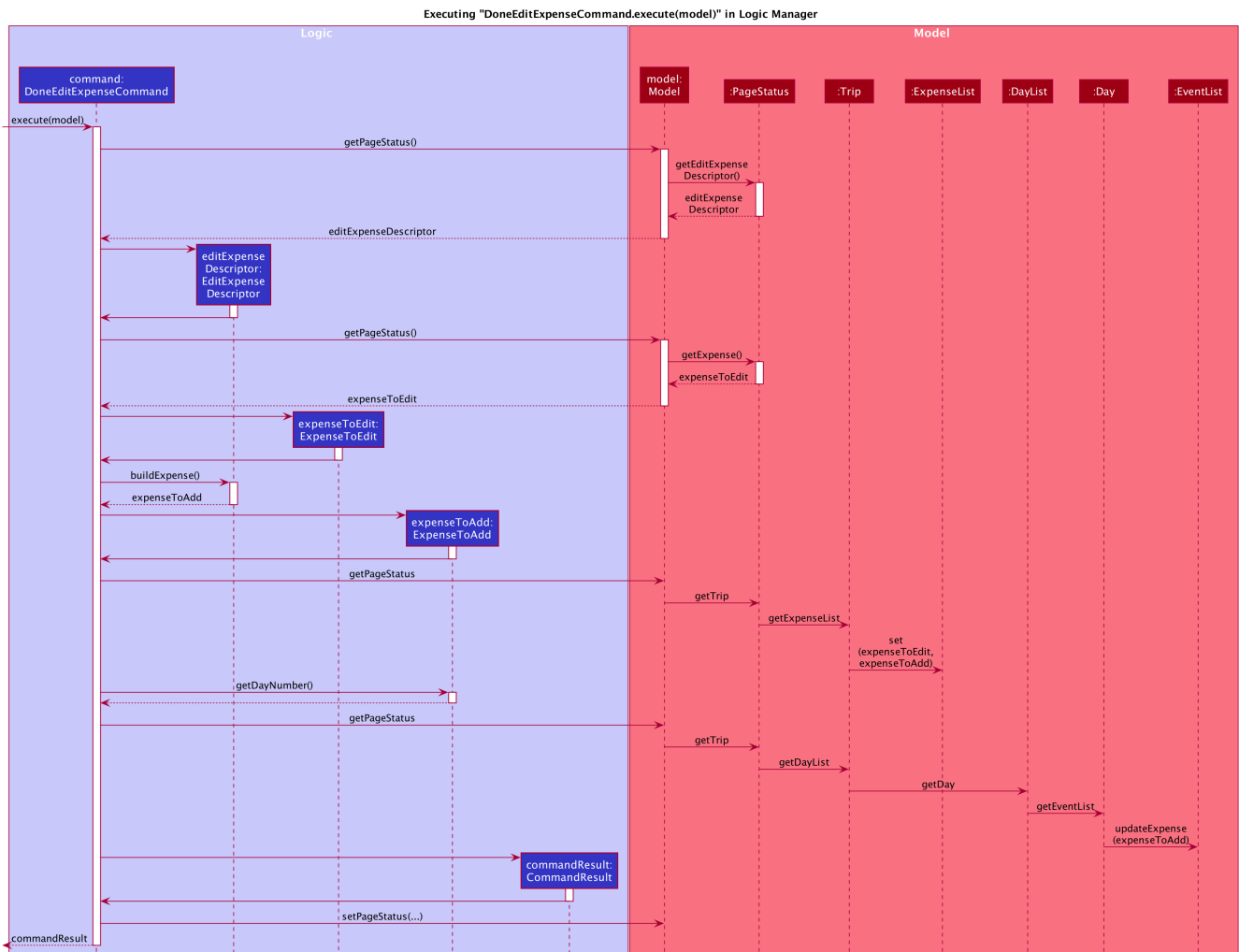


Figure 7. Sequence diagram showing the execution of DoneEditCommand

When `DoneEditCommand#execute(model)` is called, `getPageStatus()` is used to fetch information / update information from `model`. The following steps shows the sequence of event happened within the method:

1. The current instance of `EditExpenseDescriptor` and `Expense` in `model` are returned and stored as `editExpenseDescriptor` and `expenseToEdit` in logic.
2. A new instance of `Expense`, `expenseToAdd` is created by calling `buildExpense()` in `editExpenseDescriptor`.
3. Through `getPageStatus()`, `set(expenseToEdit, expenseToAdd)` is called on `EventList`, which updates the unedited `expenseToEdit` by replacing it with the edited `expenseToAdd`.
4. The `DayNumber` in `expenseToAdd` is returned so that logic can update the associated `Event` by going to the corresponding `Day` in `DayList`.
5. `updateExpense(expenseToAdd)` is called on `EventList` so that the target `Event` will have its `Expenditure` updated.
6. By calling `setPageStatus()`, the current `EditExpenseDescriptor` and `Expense` will be reset, the current page will be set to `Expense Manager Page`.
7. `CommandResult` is returned to give user feedback and update UI.

Delete an Expense

Deletion of expense is similar to the *Delete Trip/Day/Event* feature (see [\[Delete-Trip\]](#)).

The `DeleteExpenseCommand` in logic checks for index of deletion and type of expense. Only `MiscExpense` can be deleted through this command. Below is an example execution of deleting an expense:

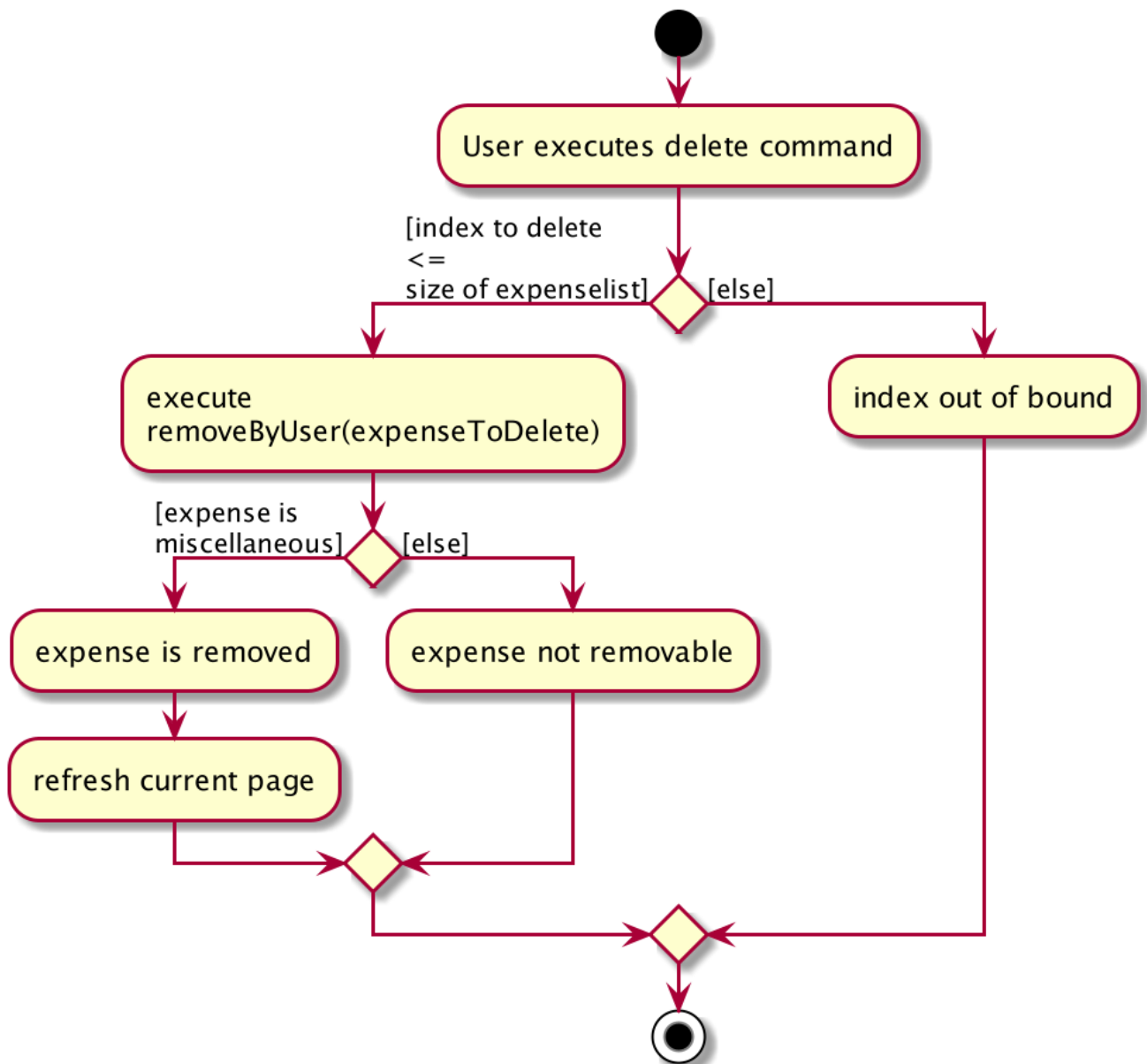


Figure 8. Activity diagram showing the execution of deleting an expense

Achievements and Learnings

Development of *TravelPal* is one of my first brown-field software engineering projects. During the process, I have gained many useful programming knowledge and consolidated my understanding of application design. I also learned how to work on a large code base as a team. It has been an educational and rewarding team working experience.