Tie Ying Kathleen - Project Portfolio

PROJECT: iFridge

Overview

iFridge is a desktop grocery management application to encourage home cooks to manage their food waste. The user interacts with it using a CLI, and it has a GUI created with JavaFX. It is written in Java, and has about 20kLoC.

Summary of contributions

- **Major enhancement**: adding **the ability to add a template to the shopping list**
  - What it does: allows the user to add items in a specified template that are not currently in their grocery list. The feature is able to deal with duplicate items as well as exclude expired items when checking the grocery list.
  - Justification: This feature improves the product significantly by making it easy for the user to generate shopping lists without having to check the fridge. In addition, the feature makes it easy for the user to generate shopping lists involving regular grocery items instead of having to add them one by one. Users can also use it to store the ingredients list of recipes.
  - Highlights: This feature is able to subtract the amount of each item already in the grocery list from the template item to be added into the shopping list. In the case that the item to be added is already in the shopping list, the item amount is updated with the sum of needed amount and the previous amount recorded. To complement this function, a UnitDictionary class was added in order to enable cross-unit calculations as long as the units were of the same unit type.
- **Minor enhancement**: added template list that allows the user to create and manage templates
  - For templates: Add, Edit, Delete, Clear, and List templates in the template list
  - For individual template: Add, Edit, Delete, and List items in a template
- **Minor enhancement**: added unit dictionary that keeps tracks of the unit types of items with the same name (note: non case-sensitive)
  - Supports the ability of the app to carry out arithmetic functions on similar items with different units by checking that all items with the same name have the same unit type. E.g. Volume unit type: ml, L or Weight unit type: g, kg, oz, lbs
- **Code contributed**: The samples of my functional and test code can be viewed here (https://nus-cs2103-ay1920s1.github.io/tp-dashboard/#=undefined&search=teika97).
- **Other contributions**:
  - Project management:
  - Enhancements to existing features:
    - Updated the Food class to include an Amount field #34 (https://github.com/AY1920S1-CS2103-F10-2/main/pull/34)
  - Documentation:

- Updated ContactUs, AboutUs: #30 (https://github.com/AY1920S1-CS2103-F10-2/main/pull/14)
- Updated Class diagram for Logic component

- Community:
  - Reported bugs and suggestions for other teams in the class (examples: 1 (https://github.com/teika97/ped/issues/1), 3 (https://github.com/teika97/ped/issues/3), 4 (https://github.com/teika97/ped/issues/4))
  - Contributed to forum discussions (examples: 1 (https://nus-cs2103-ay1920s1.slack.com/archives/CM4H41AN9/p1568817306026000))

Contributions to the User Guide

> *Below are some notable sections I contributed to the User Guide which showcase my ability to write documentation targeting our target end-users.*

Templates List Management

Add new template: `tlist add`

Adding new templates into your template list can help make shopping for groceries easier. For example, you can add an ingredients list for a favourite recipe, or add a template for items you buy on a regular basis. For easy viewing, templates in the template list are sorted automatically by alphabetical order.

Add a new template into template list.
Format: `tlist add n/TEMPLATENAME`

Examples:

- `tlist add n/Weekly Necessities`
- `tlist add n/Birthday Party Prep`
- `tlist add n/Beef Goulash`

> - All templates in the template list must have a unique name. You will not be able to add a template that has the same name as another existing template. Checking of names is case-insensitive.

Edit template name: `tlist edit`

You can also edit the names of existing templates. In the app, you can use the index number of the template to indicate which template you want to edit.

Edit the name of a specified template in template list.
Format: `tlist edit INDEX n/TEMPLATENAME`

> - Edits the template at the specified `INDEX` . The index refers to the index number shown in the displayed template list. The index **must be a positive integer** 1, 2, 3, …
> - All templates in the template list must have a unique name. You will not be able to edit a template with the name of another existing template.

Examples:

- `tlist edit 1 n/Daily Necessities` Edits the name of the 1st template in the template list to `Daily Necessities`

Deleting a template: `tlist delete`

If you no longer need a existing template, delete it to keep your template list neat and organised. You can use the index of a template to indicate the template you want to delete.

Deletes a specified template from the template list.
Format: `tlist delete INDEX`

- Deletes the template at the specified `INDEX` . The index refers to the index number shown in the displayed template list. The index **must be a positive integer** 1, 2, 3, …

Examples:

- `tlist delete 1` Deletes the 1st template in the template list

Clear template list: `tlist clear`

If you find it annoying to have to delete templates one by one, you can clear the template list completely using the clear command.

Clears all template entries from the template list.
Format: `tlist clear`

Show list of all templates: `tlist list`

If you need to view the templates in your template list, use the list command to open the template list view.

Shows all entries in the template list
Format: `tlist list`

Template Management

Add new template item: `tlist template add`

You can add a new item to an existing template.

Adds an item into a specified template.
Format: `tlist template add TEMPLATEINDEX n/NAME a/AMOUNT`

- Adds item into the template under the specified `INDEX` as shown in the displayed template list.
- All items in the template must be unique. Checking of similar names is case-insensitive.
- Amount must have both magnitude and unit.
- The system only recognises metric units. e.g. kilogram, liter. Items with the same name (check is non-case-sensitive and across templatelist, grocerylist and shoppinglist) can have different units as long as the unit belongs in the same unit category.
- Only the units in the following unit categories are supported:

- `Weight : kg , g , oz , lbs`
- `Volume : L , ml`
- `Quantity : units`

Examples:

- `tlist template add 1 n/Milk a/1L`
- `tlist template add 2 n/Eggs a/12units`

Edit item name: `tlist template edit`

You can also edit the name and the amount of existing items.

Edits a specified item in the specified template `TEMPLATENAME` .
Format: `tlist template edit TEMPLATEINDEX i/ITEMINDEX [n/NAME] [a/AMOUNT]`

- Edits the food item at the specified `ITEMINDEX` in template at specified TEMPLATEINDEX. The index refers to the index number shown in the displayed template. The index **must be a positive integer** 1, 2, 3, …
- At least one of the optional fields must be provided.
- Existing values will be updated to the input values.
- Amount must have both magnitude and unit.
- The system only recognises metric units. e.g. kilogram, liter. Note that unit type in the amount field must belong to the same unit category of the item specified and pre-existing items with the same name (check for names is non case-sensitive).
- Only the units in the following unit categories are supported:
  - `Weight : kg , g , oz , lbs`
  - `Volume : L , ml`
  - `Quantity : units`

Examples:

- `tlist template edit 1 i/1 n/Low-Fat Milk` Edits the name of the first food item in the first template to `Low-Fat Milk`
- `tlist template edit 1 i/1 a/2L` Edits the amount of the first food item in the first template to 2 litres.

Delete food item: `tlist template delete`

You can also delete items in a specific template.

Deletes the specified item from the specified template.
Format: `tlist template delete TEMPLATEINDEX i/ITEMINDEX`

- Deletes the food item at the specified `ITEMINDEX` . The index refers to the index number shown in the displayed template. The index **must be a positive integer** 1, 2, 3, …

Examples:

- `tlist template delete 1 i/1` Deletes the first food item in the first template.

Shows template: `tlist template list`

You can also view all the items in a specific template currently.

Shows all entries in the specified template.
Format: `tlist template list TEMPLATEINDEX`

Examples:

- `tlist template list 1` Shows all entries in the first template

Add items in template into shopping list: `slist addTemp`

If you want to make sure you have all the items in a template, You can automatically generate a shopping list using the template without having to check your grocerylist .

Adds all template items that are not currently found in the grocery list into the shopping list.
Format: `slist addTemp INDEX`

Examples:

- `slist addTemp 1`

Contributions to the Developer Guide

> *Below are some notable sections I contributed to the Developer Guide that showcase my ability to write technical documentation and the technical depth of my contributions to the project*

For example, the `Logic` component (see the class diagram given below) defines it's API in the `Logic.java` interface and exposes its functionality using the `LogicManager.java` class.

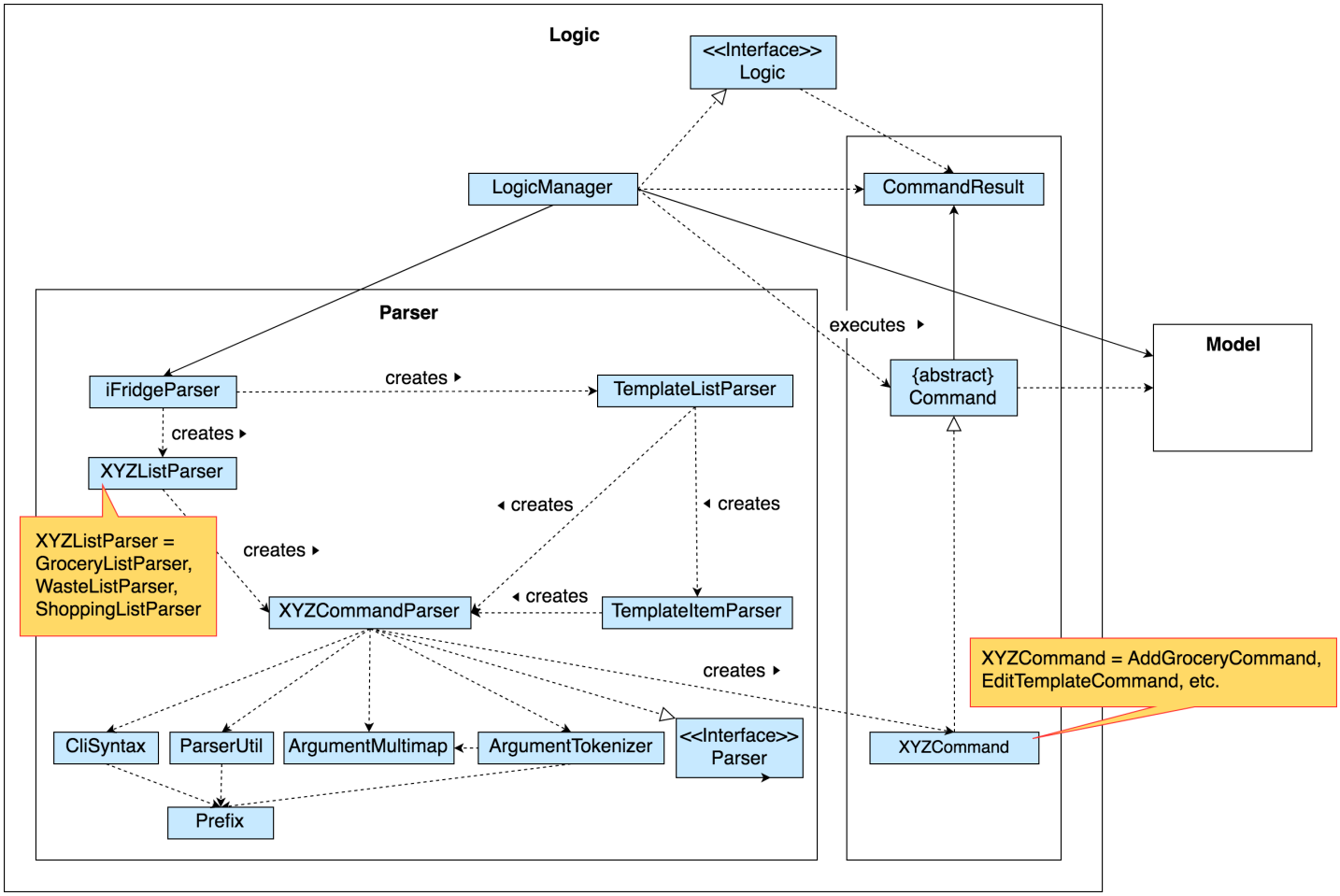


Figure 1. Class Diagram of the Logic Component

Add Template Feature

The add template mechanism is facilitated by `UniqueTemplateItems` , `TemplateList` .

The `TemplateList` is an observable list of `UniqueTemplateItems` while the UniqueTemplateItems contains an observable list of template items.

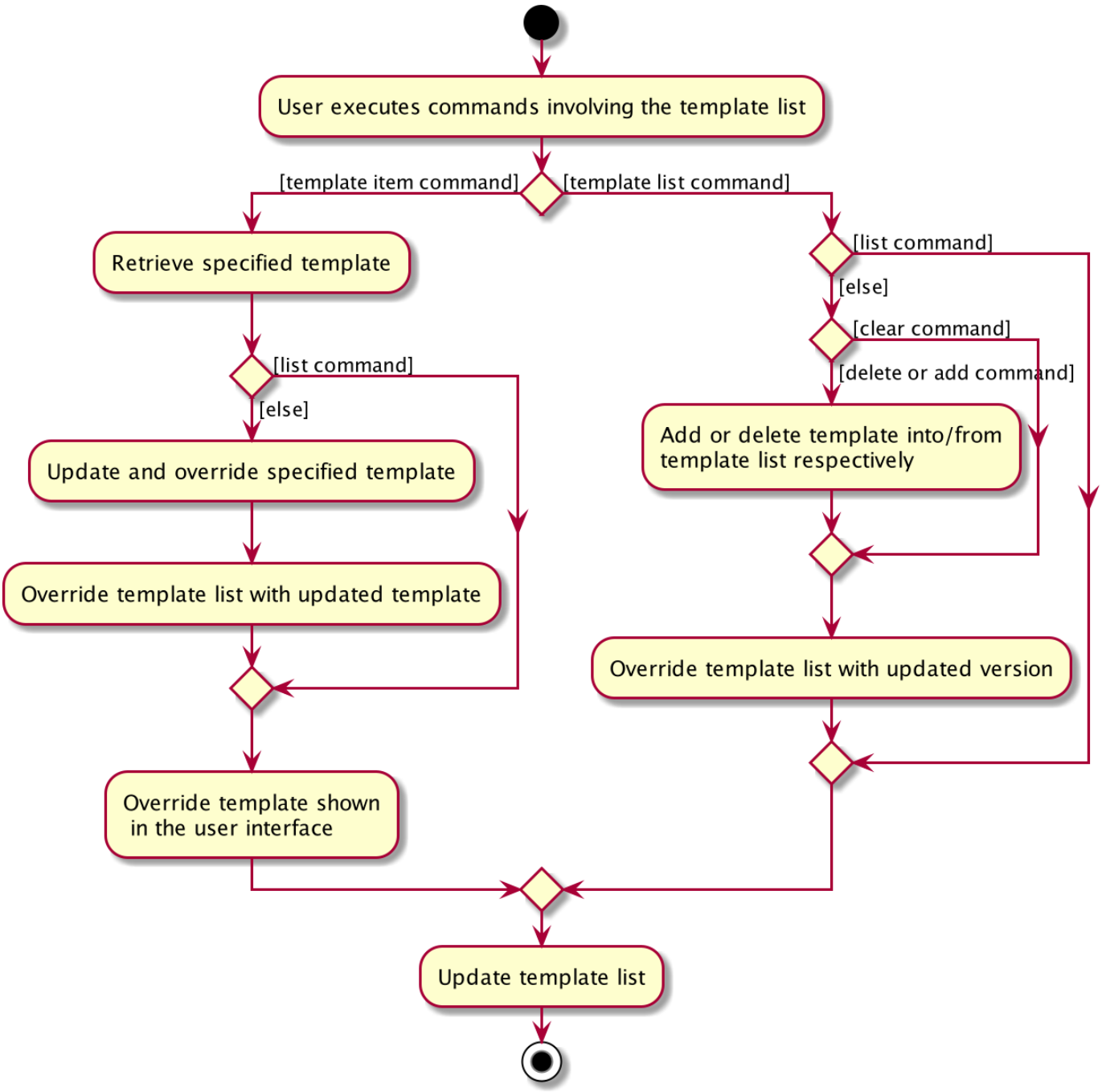
To add a template into the `TemplateList` , a new `UniqueTemplateItems` object is created with the entered name. The model is updated with the new `TemplateList` , the `TemplateToBeShown` , which is an instance of the object `UniqueTemplateItems` containing the details of the template being edited or viewed will not be updated. Only a `TemplateItemCommand` will involve an update of the `TemplateToBeShown` .

The following activity diagram summarizes what happens when a user executes a new command related to managing of template items: image::TemplateListCommandUIActivityDiagram.png[]

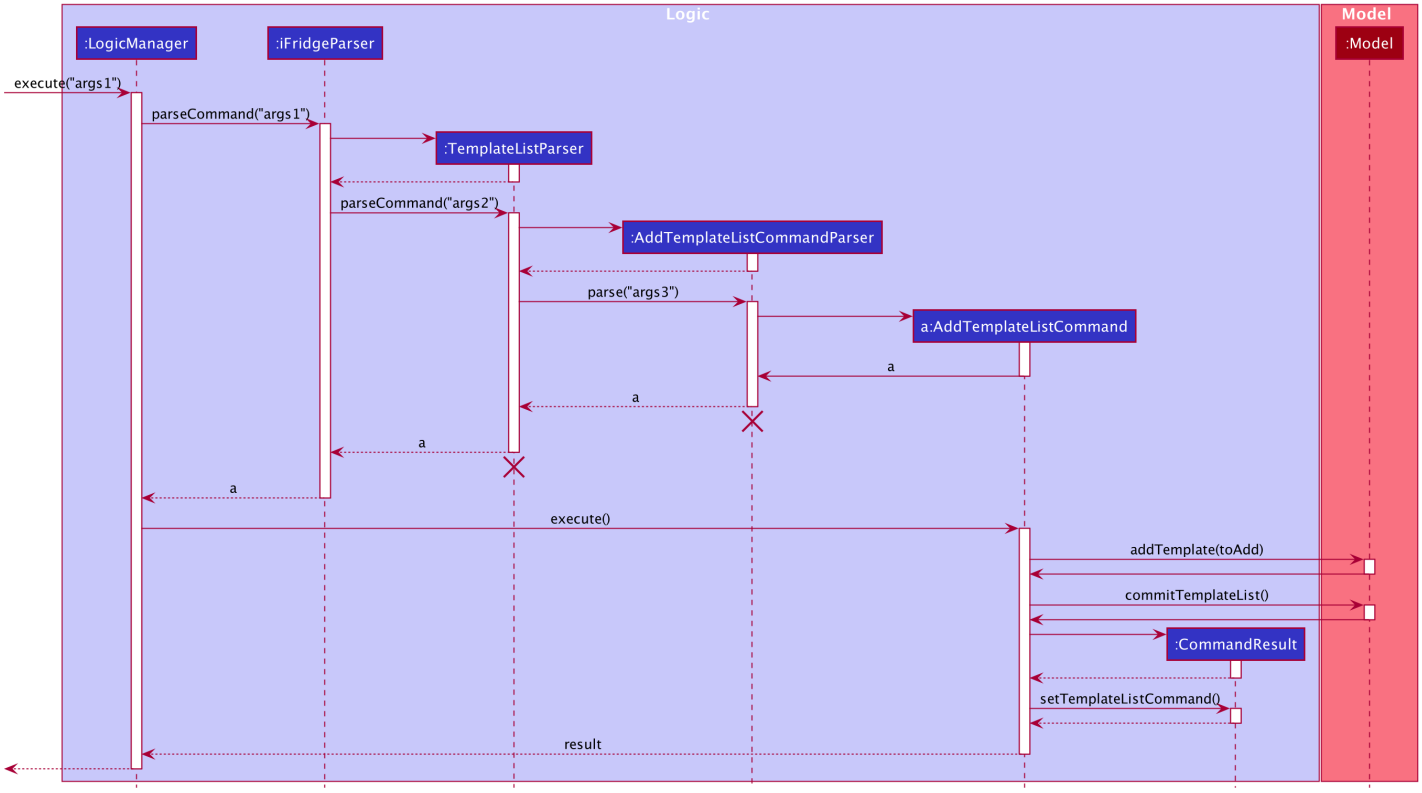
NOTE

> Due to multiple lists in the iFridge app, the template list will only be usable to the user when executing a TemplateList or a TemplateItem Command. For example, an AddTemplateListCommand or a AddTemplateItemCommand.

The following activity diagram summarizes what happens when a user executes a new template list command related to the managing of templates and illustrates some differences in the UI as compared to when a template item command is executed:



The following sequence diagram shows how the edit template item operation works for the logic component:



Due to lack of space, please refer to the below list for args1, args2, args3 shown in the diagram above.

- args1: "tlist add n/Tomato Soup"
- args2: "add n/Tomato Soup"
- args3: "n/Tomato Soup"

NOTE

> The lifeline for `TemplateListParser`, `AddTemplateListCommandParser` should end at the destroy marker (X) but due to a limitation of PlantUML, the lifeline reaches the end of diagram.

Design Considerations

Aspect: How add command is parsed

Refer to the above class diagram about the logic component that shows the relationship between the TemplateListParser and the TemplateItemParser.

- Alternative 1 (current choice): Create a separate parser for template list management and template item management
  - Pros: Easy to implement. Parser structure follows the same structure as the model. More OOP.
  - Cons: May be confusing to differentiate between TemplateItemParser and TemplateListParser.
- Alternative 2: The TemplateListParser is able to distinguish between template and template item management and call the respective XYZCommandParsers
  - Pros: Less confusing as there is only one parser toggling between the different command parsers to managing the template list.
  - Cons: Implementation of the parser becomes less OOP.

- Alternative 3: The TemplateItemParser is at the same hierarchy as the TemplateListParser instead of inside.
  - Pros: The user command can be shorter. E.g. "template edit …" instead of "tlist template edit …"
  - Cons: Not as obvious to the user that both commands involve the same template list.

Adding a template

a. Prerequisites: Template list must contain a template with the name 'Birthday Party'.

b. Test case: `tlist add n/x` (where x is a template name that does not currently exist in the template list)
Expected: Template added into the list. The templates in the list are sorted by alphabetical order.

c. Test case: 'tlist add n/Birthday Party'
Expected: No template added to the template list. Error details shown in the status message.

d. Other incorrect add commands to try: `add`, `tlist add`, `tlit add`

Deleting a template

a. Prerequisites: There must be at least one template in the template list.

b. Test case: `tlist delete 1` Expected: Template deleted from the list.

c. Test case: `tlist delete x` (where x is larger than the template list size) Expected: No template deleted. Error details shown in the status message.

d. Other incorrect delete commands to try: `tlist delete`, `tlist 0`

Editing a template

a. Prerequisites: There must be at least one template in the template list with no templates named `Monthly Necessities`.

b. Test case: `tlist edit 1 n/Monthly Necessities` Expected: Template edited with the name `Monthly Necessities`.

c. Test case: `tlist edit 0 n/Monthly Necessities` Expected: No template edited. Error details shown in the status message.

Adding a template item

a. Prerequisites: Template list must contain at least one template.

b. Test case: `tlist template add 1 n/x a/y` (where x is a item name that does not currently exist in the first template and y is a valid amount for the item). Expected: Template item added into the list.

c. Test case: `tlist template add 0 n/x a/y` (where x is a item name that does not currently exist in the first template and y is a valid amount for the item). Expected: Template item not added into the list. Error details shown in status message.

d. Other incorrect add template item commands to try:

   i. `tlist template add 1 n/x a/y` (where x is an item name that already exists in the template)

   ii. `tlist template add 1 n/x a/y` (where x is an item name that already exists in another template or list, and y is an invalid amount with a unit type that conflicts that of the other item entry)

Editing a template item

a. Prerequisites: Template list must contain at least one template, which contains at least one item.

b. Test case: `tlist template edit 1 i/1 n/x` (where x is an item name that does not already exist in the

template). Expected: Template item added into the template.

c. Test case: `tlist template edit 1 n/x` (where x is an item name that does not exist in the template). Expected: Template item not added into the list. Error details shown in the status message.

d. Other incorrect edit template item commands to try:

i. `tlist template edit 1 i/1 n/x` (where x is an item name that does not exist in the template).

ii. `tlist template edit 1 i/1 n/x a/y` (where x is an item name that already exists in another template or list, and y is an invalid amount with a unit type that conflicts that of the other item entry)

iii. `tlist template edit z i/1 n/x a/y` (where x and y are valid name and amounts and z is greater than the template size.

Deleting a template

a. Prerequisites: Template list must contain at least one template, which contains at least one item.

b. Test case: `tlist template delete 1 i/1` Expected: Template item deleted from the template.

c. Test case: `tlist template delete 1 i/0` Expected: Template item not deleted. Error details shown in the status message.

d. Other incorrect delete template item commands to try: `tlist template delete x i/1` where x is greater the the template list size, `tlist template 1 i/x` where x is greater than the template size.

Adding a template to the shopping list

a. Prerequisites: Template must contain at least one template with one item with name 'FullFat Milk' and amount '300ml. Grocerylist should have several entries of FullFat Milk with slight variations in name and amount. For e.g.:

i. For name: fullfat milk, Fullfat Milk, FULLFAT MILK

ii. For amount: 300ml, 400ml, 1L, 0.3L

b. Test case: `slist addTemp 1` Expected: All items in the template that are either not found in the grocery list or are already expired will be added into the shopping list. The amount of the item that is added to the shopping list is the result after subtracting all non-expired grocery item amounts from the amount stated in the template item entry. In the case that the sum in the grocerylist exceeds the required amount, the item will not be added to the shopping list and message will be shown stating all items are in stock. In the case that the shopping list already has FullFat Milk, the result is added to original amount recorded.

c. Test case: `slist addTemp 0` Expected: No items added into the shopping list. Error details shown in the status message.

d. Other incorrect commands to try: `slist addTemp x` where x is greater than the template list size.