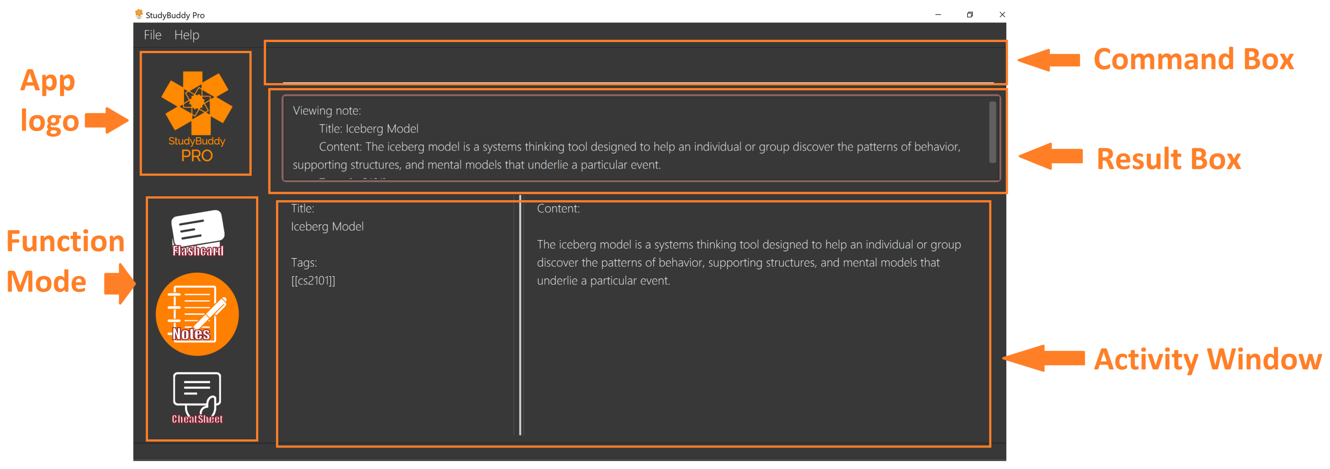
Chen Kai Bin - Project Portfolio for StudyBuddyPro

About the project

My team of 4 software engineering students and I were tasked to enhance a basic command line interface desktop addressbook application for our Software Engineering Project. We chose to morph it into an application that aids users in their studies, called StudyBuddyPro. This enhanced application has 3 main features, the flashcard feature which allows users to test their concepts and offer reminders, notes feature and the cheatsheet feature, which has a special auto-generation functionality.

This is what our project looks like



Viewing note:
Title: Iceberg Model
Content: The iceberg model is a systems thinking tool designed to help an individual or group discover the patterns of behavior, supporting structures, and mental models that underlie a particular event.

Title:
Iceberg Model

Tags:
[[cs2101]]

Content:

The iceberg model is a systems thinking tool designed to help an individual or group discover the patterns of behavior, supporting structures, and mental models that underlie a particular event.

Figure 1. The graphical user interface for ***StudyBuddyPro****.*

Note the following symbols and illustrations :

`filterall`

A highlight would indicate that this is a command that can be inputted in the command line of our application.

Summary of contributions

This section shows a summary of my coding, documentation, and other helpful contributions to the team project.

Enhancement added: I added the tagging functionalities for all items in the application.

- What it does

Allows all StudyBuddyItems (Flashcard, Note, NoteFragment, CheatSheet) to be accessed through specific tags.

- Justification

The usage of tags are highly important in our application. For instance, the auto-generation feature of CheatSheet, pulls contents from the other modes in StudyBuddyPro that matches the tags. The timetrial feature of the flashcard mode, also makes use of matching tags to create a deck of flashcards to display to the user. In addition, users are also able to filter any StudyBuddyItem by specifying tags, to easily search for items they want.

- Highlights

This enhancement is highly straightforward and works well with the current application.

Other contributions:

- Project management:
    - Mainly managed milestone v1.3.
    - Designed the mockup for the application’s Ui (#49).
- Developing existing features:
    - Added barebones for the cheatsheet functionality, which was later on finished up by my groupmate (#124).
    - Helped with the linking of the Storage component to our new features (#124).
- Enhancements to existing features:
    - Upgraded the auto-generation cheatsheet feature of the application to pull intra-content tags from the content feature of the application (#244).
    - Enhanced delete commands to prompt the user once to confirm his/her deletion(#244).
    - Updated the appIcons for the application, and created the logo for the application using an online software, logomakr (#254).
    - Wrote additional tests for existing features to increase the test coverage(#316).
- Documentation:
    - Added NFRs for the application(#77).
- Community:
    - Reported bugs and offered suggestions for the other team during the Practical Exam dry run.

Contributions to the user guide

We had to update the original addressbook User Guide with instructions for the enhancements that our group has added. The following is an excerpt from our **StudyBuddyPro User Guide**, showing the additions that I have made for the filter features.

The following section describes how user are now able to make good use of tags to find their desired items in our app. The section also contains an excerpt on how the tagging feature can be further improved in the next version of StudyBuddyPro.

Listing all current Tags in StudyBuddyPro : `taglist`

Displays a full list of all tags currently in StudyBuddyPro.

```
Format: taglist
```

```
Expected output:
Here are all the tags in StudyBuddyPro.
Listing all tags :
[cs2100] |  flashcards : 0 notes : 3 cheatsheets : 1
[cs2101] |  flashcards : 6 notes : 2 cheatsheets : 1
[cs2104] |  flashcards : 20 notes : 8 cheatsheets : 3
[math] |  flashcards : 10 notes : 2 cheatsheets : 1
[pipelining] |  flashcards : 1 notes : 5 cheatsheets : 2
```

- The user can make use of `taglist` to quickly see which tag they would like to view.
- The list of tags is also automatically sorted alphabetically.
- Subsequently, the user can use the `filterall` and specify a tag to get a list of related items.

Listing all StudyBuddy items by their tag : `filterall`

Each StudyBuddyItem has a set of tags tied to them. `filterall` allows the user to list all StudyBuddy items with matching tags in the application. The user is able to use this command regardless of which mode they are currently in.

Let’s say the user wishes to view the definition of pipelining. Pipelining is taught in CS2100, a Computer Organization module taught in the School of Computing at NUS. Hence, the user can make use of filterall to find all flashcards, cheatsheets and notes that are tagged “CS2100”. Note that for simplicity, all tags will be converted to lowercase upon input. Hence, ‘CS2100’ will be read as ‘cs2100’ by our application.

```
Format: filterall tag/TAG [tag/TAG]...
```

```
Example usage: filterall tag/CS2100
```

```
Expected output:
List the whole StudyBuddy by tag(s) :
cs2100
Flashcard: 6.
Question: What is 101 Binary in its Decimal form?
Title: BinaryQn
Tags: [cs2100]
CheatSheet: 7.
Title: cs2100 stuff
Tags: [cs2100]
Contents: [ 1. Pipelining is a process where.. ]
    [ 2. Question: What is 101 Binary in its Decimal form?; Answer: 5 ]
Note: 5.
Title: Pipelining Definition
Content: Pipelining is a process where..
Tags: [cs2100]
Note Fragment: 3.1.
Title: About
Content: highlighted
Tags: [cs2100]
```

All Study Buddy Items in the application will be displayed to the user, alongside with their corresponding indexes. This helps the user to quickly get to their desired flashcard/cheatsheet/note.

Let's say the user is currently in the flashcard mode. In this case, the user will see that the definition for pipelining is currently the 5th Note in the Notes feature of StudyBuddyPro. Hence, the user will first key in the following input:

```
switch notes
```

which will have the expected output of :

```
You are currently using the notes function!
```

which brings the user to the Notes Function of StudyBuddyPro. Following this, the user will simply key in the following input:

```
view 5
```

To view the specific Note on the definition of Pipelining.

The user is also able to specify a multiple number of tags to filter by. For instance,

```
filterall tag/CS2100 tag/important
```

This will be especially useful if the user wishes to view the more important items of a certain module.

Listing by tags: `filter`

In addition to the filterall command, the user is able to use the command filter to list the items with the specified tag in the mode the user is currently in.

It is similar to the filterall command, except it is for individual features. It will be truncated in this Product Portfolio, please refer to the UserGuide for more details.

Deleting a cheatSheet: `delete`

Deletes a cheatSheet by the specified index.

The user will be prompted once to confirm their deletion.

```
Format: delete (index)
```

```
Example usage: delete 8
```

```
Expected output:
Are you sure you would like to delete the following cheatsheet?
Title: CS2100 Finals CheatSheet Tags: [finalcheatsheet]
Contents: [ 1. Question: What is 110 Binary in its Decimal Form?; Answer: 6 ]
    [ 2. 110 in Binary is 6 is Decimal ]
Hit enter again to confirm your deletion.
```

Upon deleting any StudyBuddyItem (Flashcard, Note, NoteFragment, CheatSheet), the user will be prompted once to confirm his deletion. The user would only need to hit enter once more to confirm his/her deletion.

This will prevent any accidental deletion of wrong items.

Editing a tag: `edit tag/` (proposed in v2.0)

Edits a tag by the specified index.

```
Format: edit tag/CURRENT tag/NEW
```

```
Example usage: edit tag/midterm tag/finals
```

```
Expected output:
Tag editted!
All items and contents in StudyBuddy tagged midterm is replaced with tag finals.
```

This allows the user to easily modify the tags of all the items with a single command.

Contributions to the developer guide

The following shows my additions to the StudyBuddyPro Developer Guide for the tagging feature.

Tagging Feature

Implementation

The current implementation of StudyBuddyItems in StudyBuddyPro is such that it contains a Set of Tags.

The following objects of each individual feature shares similar Tagging behaviour, as shown in the class diagram 2 below.

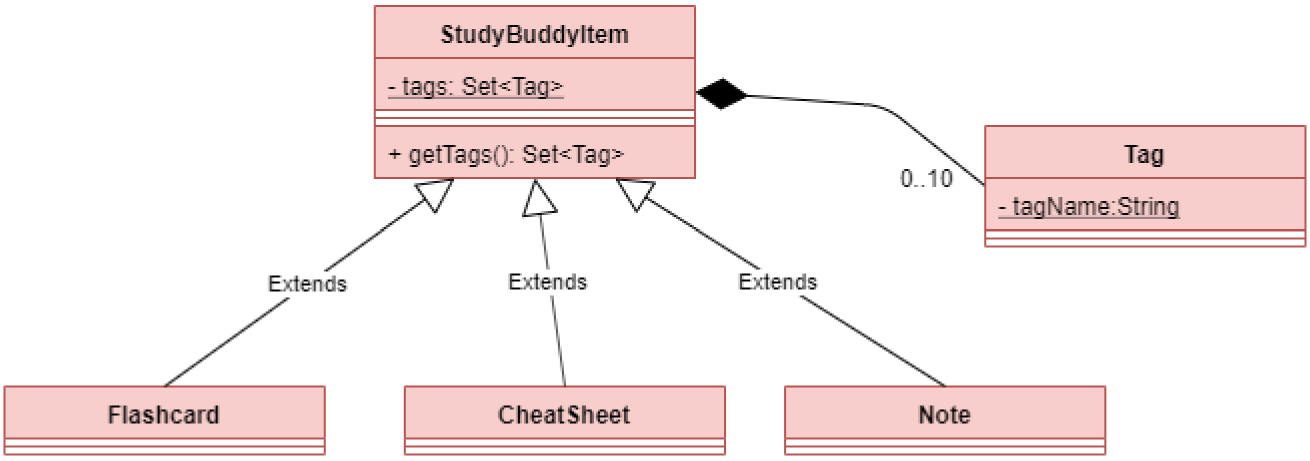
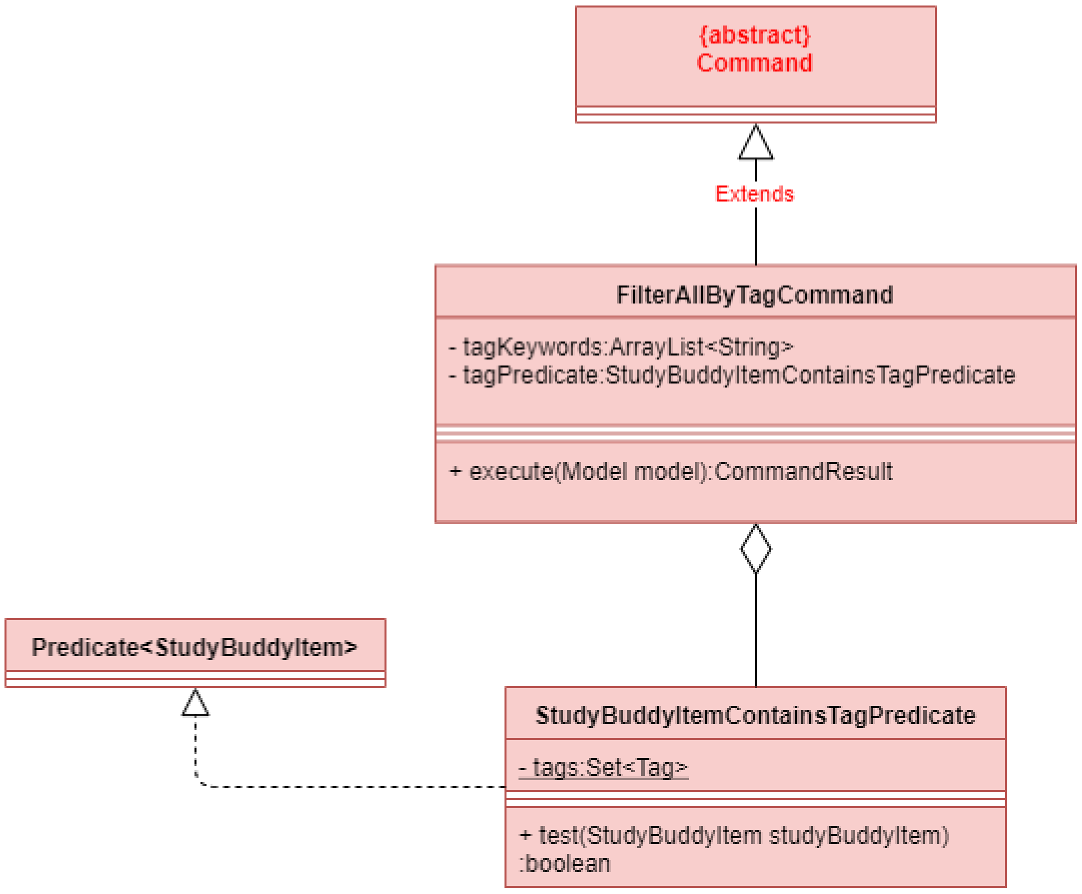


Figure 2. Implementation of StudyBuddyItem

- Design Considerations
    - As explained in the class diagram above, each StudyBuddyItem is limited to a total number of 10 tags.
    - It is designed as such to prevent users from over-cluttering the result display when they view items that have too many tags.
    - To reduce confusion for the user, all tags will be converted to lower-case upon initialization.

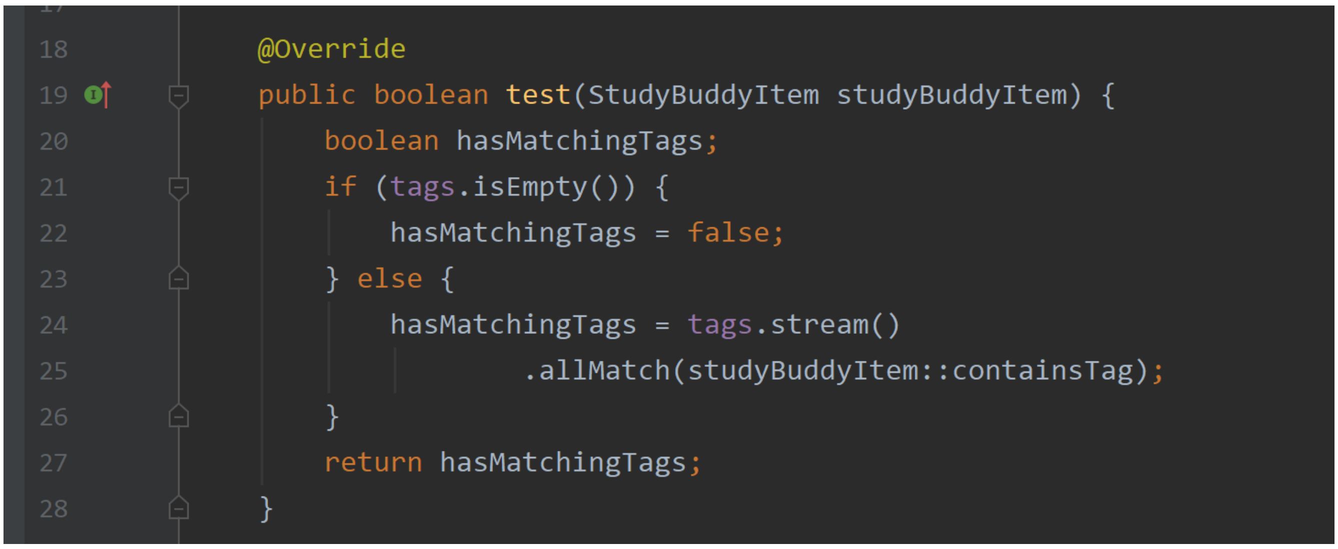
Aspect: How tag predicates are implemented



{abstract}
Command
Extends
FilterAllByTagCommand
- tagKeywords:ArrayList<String>
- tagPredicate:StudyBuddyItemContainsTagPredicate
+ execute(Model model):CommandResult
Predicate<StudyBuddyItem>
StudyBuddyItemContainsTagPredicate
- tags:Set<Tag>
+ test(StudyBuddyItem studyBuddyItem)
:boolean

Figure 3 : Class Diagram of how StudyBuddyItemContainsTagPredicate is implemented

- The above class diagram shows how tag predicates are being implemented.
- The set of tags that is stored in `StudyBuddyItemContainsTagPredicate` refers to the tags specified by the user.



```
    @Override
    public boolean test(StudyBuddyItem studyBuddyItem) {
        boolean hasMatchingTags;
        if (tags.isEmpty()) {
            hasMatchingTags = false;
        } else {
            hasMatchingTags = tags.stream()
                    .allMatch(studyBuddyItem::containsTag);
        }
        return hasMatchingTags;
    }
```

Figure 4 : Code Snippet of StudyBuddyItemContainsTagPredicate#test()

The current implementation is that test() only returns true if all tags specified by the user matches the current Item.

As such, there will be more correctness when auto-generating cheatsheets and filtering flashcards, as seen in the following example.

If a user wishes to generate a cheatsheet and pull items with tags [cs2100] and [difficult], it would strictly only pull difficult CS2100 contents, and not pull other items that might have tags containing [difficult].

Usage of Tags

a) **To search for items**

Inside each feature

- The user is able to specify a tag name to get a list view of all the items with that specified tag in the mode they are currently in (e.g. `filter tag/cs2100`).

Searching using Tags globally

- The user is also able to indicate a tag name get a list view of all the StudyBuddyItems across all 3 modes in StudyBuddyPro (e.g. `filterall tag/ma1521`).
- Currently, the user is able to specify multiple tags in his/her query (e.g. `filter tag/cs2100 tag/difficult`). If multiple tags are specified, only items that match all the specified tags will be listed.
- The sequence diagram below shows how listing all items across StudyBuddyPro by a specified tag works.

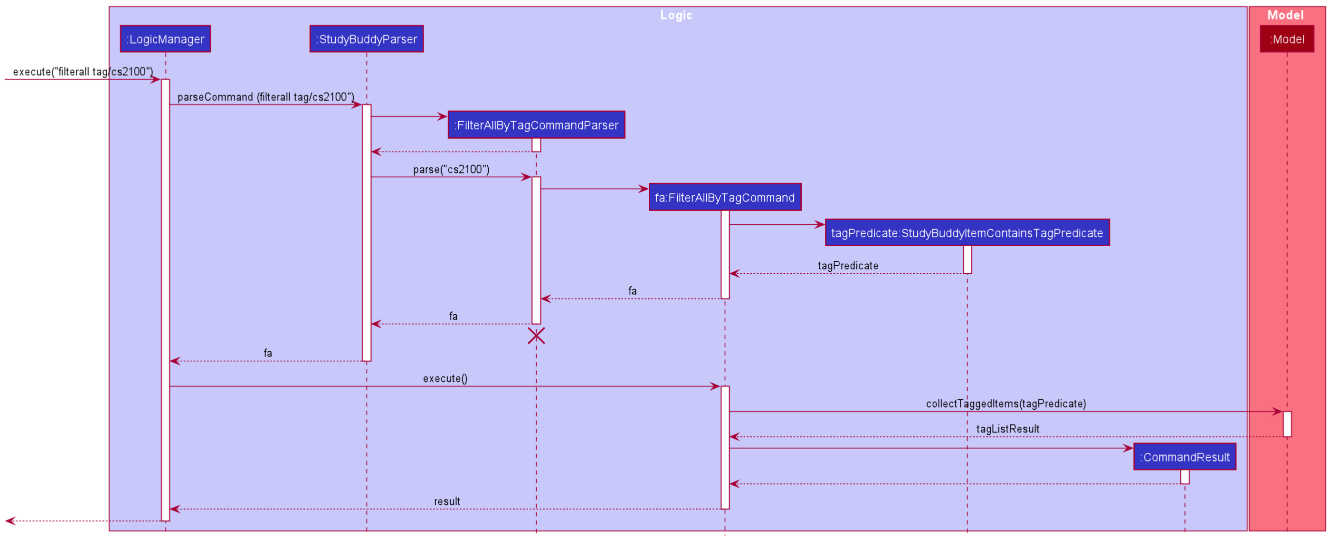

Logic
Model
:LogicManager
:StudyBuddyParser
:Model
execute("filterall tag/cs2100")
parseCommand (filterall tag/cs2100")
:FilterAllByTagCommandParser
parse("cs2100")
fa:FilterAllByTagCommand
tagPredicate:StudyBuddyItemContainsTagPredicate
tagPredicate
fa
fa
fa
execute()
collectTaggedItems(tagPredicate)
tagListResult
:CommandResult
result

Figure 5. Sequence diagram of listing items by a specified tag

[Proposed] Future improvements (Coming in v2.0)

Supporting deletion/editing of Tags

Allow the user to delete/edit a specified Tag.

All StudyBuddyItems must be updated in response to the deletion/edit.

- A proposed implementation would be to store all Tags in a Global Data Structure, and have each StudyBuddyItem reference to that Data Structure.

- As such, we can apply an Observer pattern to update each StudyBuddyItem upon deletion of a tag.

b) **For Auto-generation of CheatSheets (truncated)**

c) **For TimeTrial Mode (truncated)**

Deletion prompt feature

NOTE The deletion prompting applies to all three modes in StudyBuddyPro.

The activity diagram below shows how the user is prompted upon calling a delete command.

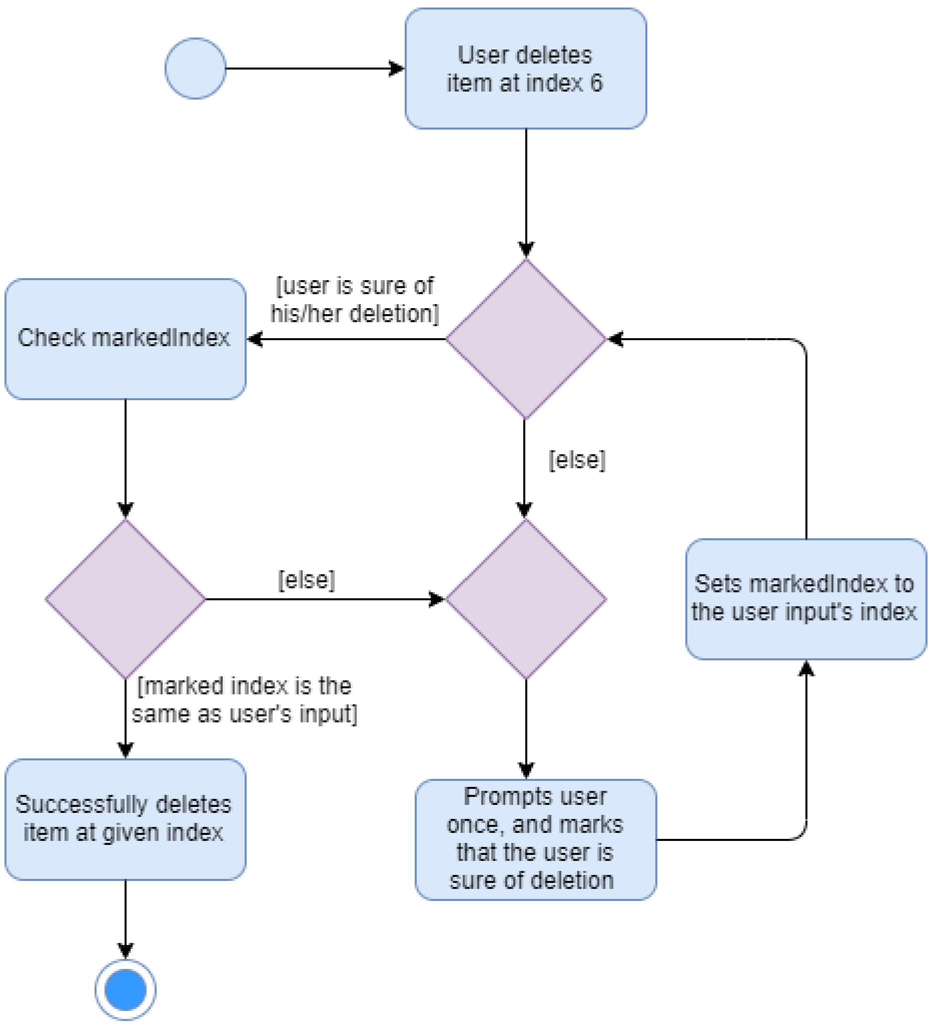

User deletes item at index 6
[user is sure of his/her deletion]
Check markedIndex
[else]
[else]
Sets markedIndex to the user input's index
[marked index is the same as user's input]
Successfully deletes item at given index
Prompts user once, and marks that the user is sure of deletion

Diagram 5 : Activity Diagram for prompting user upon deletion.

- The marked index has to be constantly tracked, to prevent the user from successfully deleting a different indexed item without any prompts.

Usage of deletion prompts

- Upon entering a delete command, e.g. `delete 6`, the item that is to be deleted will be displayed to the user, and they will be required to confirm their deletion.
- The user only needs to hit enter once more to successfully delete the item.

Design considerations

Aspect : Deletion prompts versus Undo feature.

Current Implementation (prompting user before delete)

- Pros
  - Much easier to implement as compared to undo feature.
  - Hitting enter once more to delete is highly convenient for the user, as compared to perhaps asking the user to input a 'yes' or 'no' to confirm their deletion.
- Cons
  - Less versatile than the undo feature.
  - User might still accidentally delete the wrong item.