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

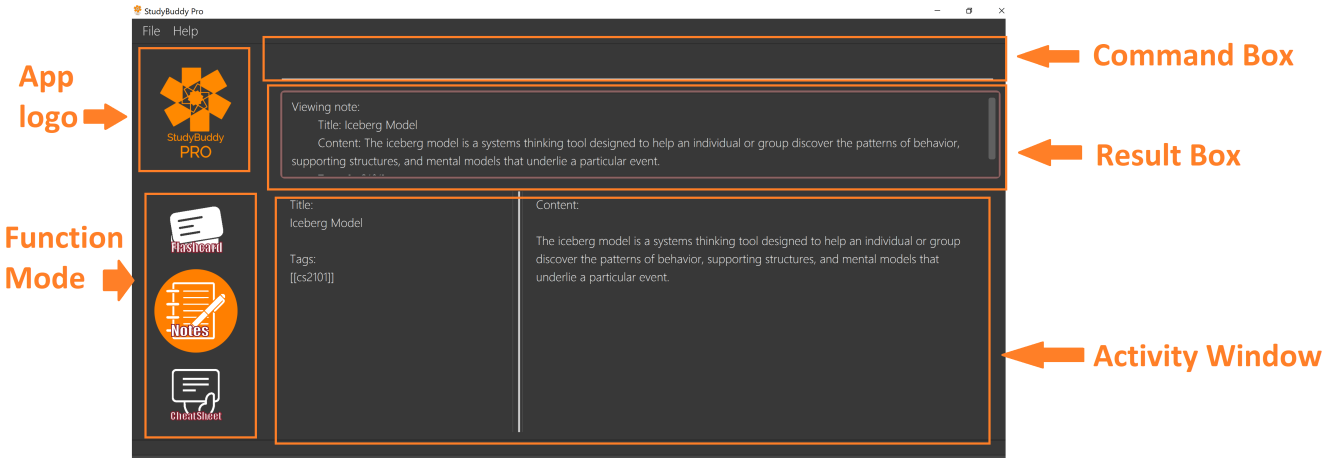

*Figure 1. The graphical user interface for **StudyBuddyPro**.*

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.

Major Enhancement 1: 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.

Minor Enhancement 1: Added the deletion prompting feature of the applcation.

- What it does

When the user attempts to delete any item (Flashcard, Note, CheatSheet) in StudyBuddyPro, the user will be prompted once to confirm his/her deletion before the item is successfully deleted.

- Justification

The deletion prompt feature will prevent users from accidentally deleting wrong items in our application. This feature is also much simpler to implement as compared the undo/redo feature, which offers almost the same outcome in this aspect.

Minor Enhancement 2: Added the ability for CheatSheets to pull NoteFragments during auto-generation.

- What it does

Upon creating a CheatSheet with a specified Tag, NoteFragments that are stored within Notes that matches the specified Tag will be added to this CheatSheet.

- Justification

NoteFragments are portions that can be highlighted within the Notes Feature. It is important that these portions are added to the CheatSheet. Also, there are instances where the entire Note is too long, and only a required portion of the Note is required. As such, the NoteFragment can be tagged differently from the Note such that the auto-generation feature of the CheatSheet will only add the tagged NoteFragments.

Code contribution:

- Please refer here to view my code contributions.

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:
  - 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`

NOTE	`taglist` and `filterall` are global commands that can be used in any mode in StudyBuddyPro.

Displays a full list of all tags currently in StudyBuddyPro.

```
Format: taglist
```

```
Expected output:
Here are all the tags in StudyBuddyPro.
Listing all tags :
[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.

IMPORTANT

- The user must specify at least one tag.
- The user is able to specify multiple tags.
- CheatSheets that match all of the specified tags will be displayed.

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]
Note: 5.
Title: Pipelining Definition
Content: Pipelining is a process where..
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`

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]
Please use `delete 8` 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 need to key in the delete command once more to confirm his/her deletion.

This will prevent any accidental deletion of wrong items.

IMPORTANT	Inputting invalid commands will **NOT** offer another prompt to the user. For example, calling `delete 3`, then an `invalid command` and then `delete 3` again will immediately delete the 3rd item without a prompt.

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.

For instance, if the user has items that are tagged [cs2100] and [midterm], and the user wishes to make use of such items to include in a cheatsheet for CS2100 finals, the user can input

```
edit tag/midterm tag/finals
```

to conveniently change, for instance, all notes tagged with [midterm] to be tagged with [finals].

This then allows the user to more conveniently generate a cheatsheet for his/her final exams.

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.

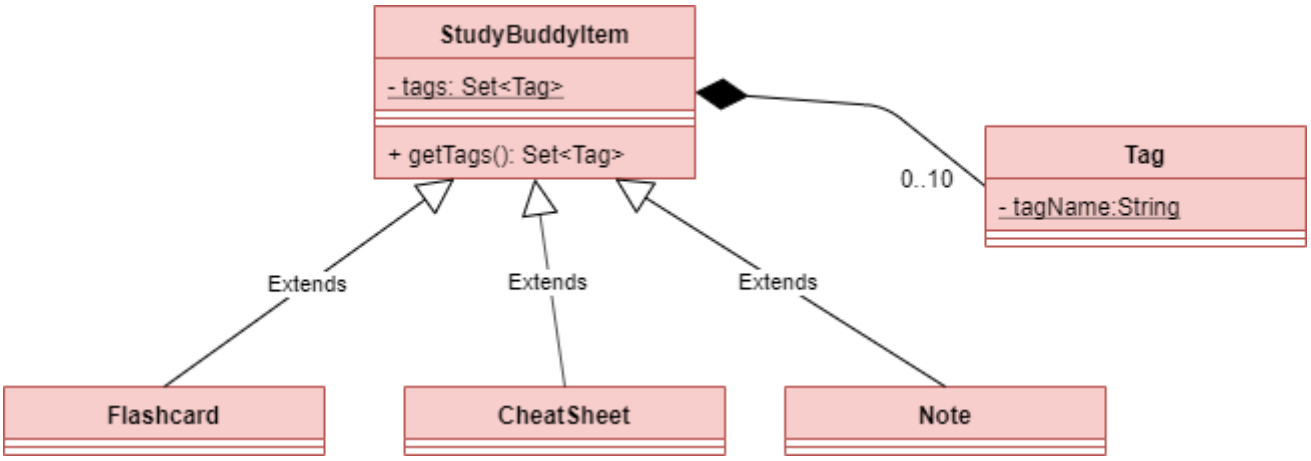

Figure 2. Implementation of StudyBuddyItem

- Design Considerations
  - As explained in the class diagram above, each StudyBuddyItem is limited to a total number of 10 tags.
  - The user is not able to create an item in StudyBuddyPro with more than 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

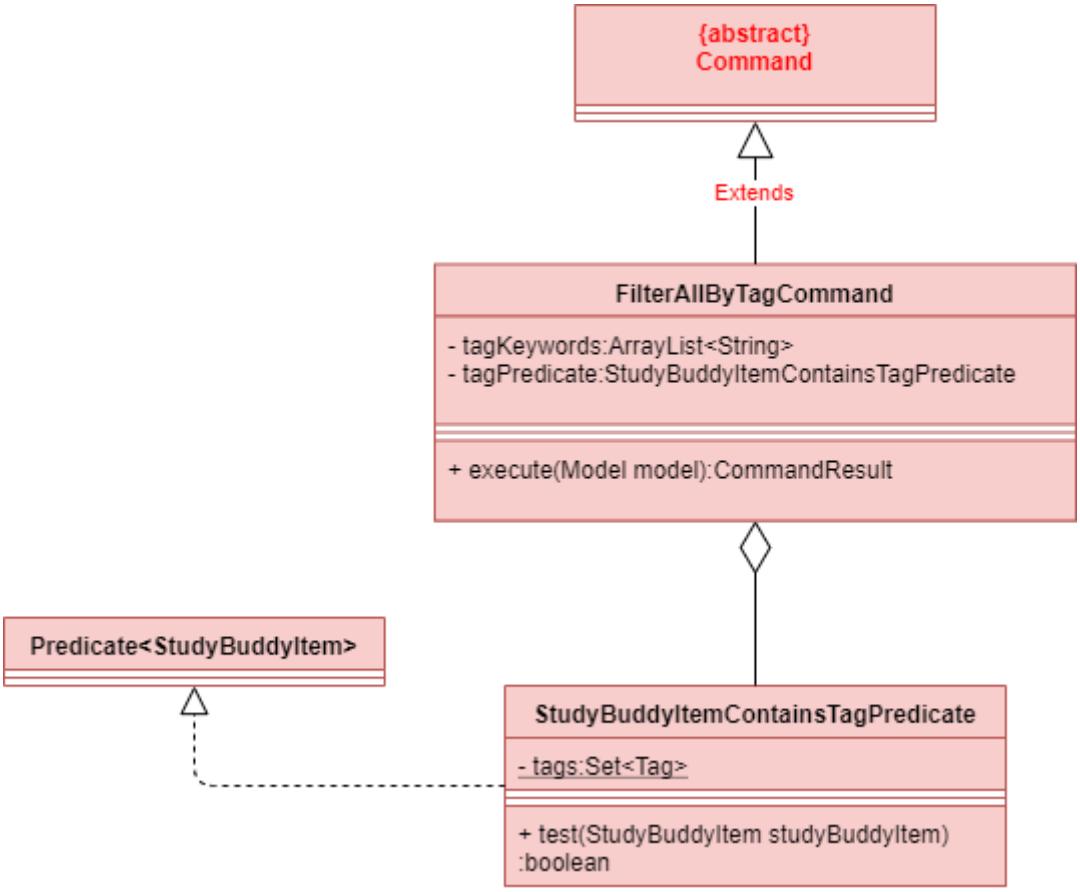

Figure 3 : Class Diagram of how StudyBuddyItemContainsTagPredicate is implemented

- 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.

```
18      @Override
19      public boolean test(StudyBuddyItem studyBuddyItem) {
20          boolean hasMatchingTags;
21          if (tags.isEmpty()) {
22              hasMatchingTags = false;
23          } else {
24              hasMatchingTags = tags.stream()
25                      .allMatch(studyBuddyItem::containsTag);
26          }
27          return hasMatchingTags;
28      }
```

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].
- Such logic for filtering items by their tag is similar throughout the whole StudyBuddyPro.

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`).
- The user can also call the global command, `taglist` to get a listview of all tags in StudyBuddyPro.
- 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.

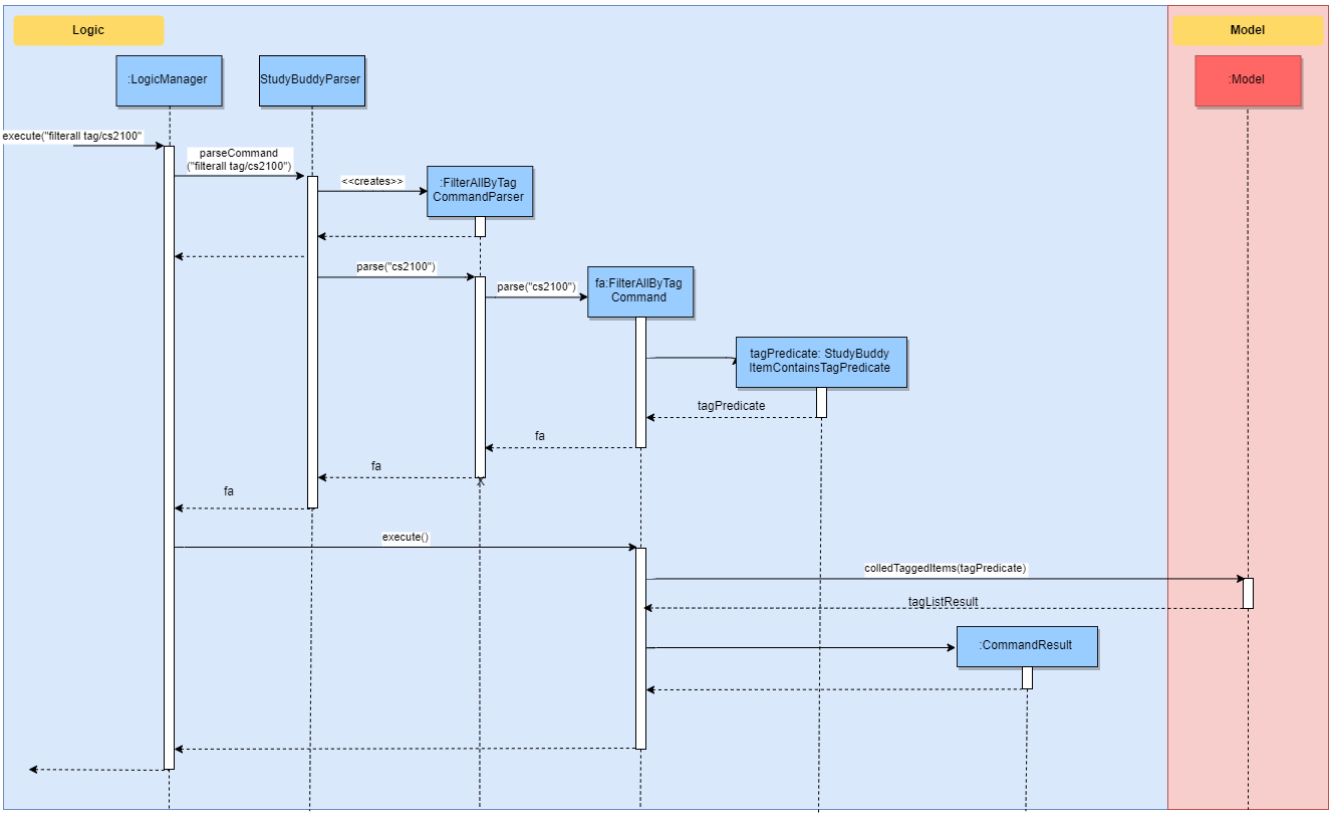

Figure 5. Sequence diagram of listing items by a specified tag

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

b) For Auto-generation of CheatSheets

- Upon adding a cheatsheet, the cheatsheet will make use of tags to automatically pull contents from other features of StudyBuddyPro.
- StudyBuddyItems with tags that match the user's input will be pulled over.
- This feature will be further elaborated in the next section, Section 4.4.

c) For TimeTrial Mode

- The TimeTrial Mode of the flashcard feature, will make use of the tagging feature.
- It will do so by filtering out flashcards with tags that match the user's input.
- For instance, if a user wishes to revise only important flashcards, he/she could enter the following command, `timetrial important`.

IMPORTANT	The syntax used here is slightly different. The user need not specify the `tag/` keyword to indicate that the item is a tag.

- The TimeTrial feature will be further elaborated in Section 4.5.

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

Supporting deletion/editing of Tags

Allows the user to delete/edit a specified Tag.

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

- Allow the user to delete a specified Tag.
- All StudyBuddyItems must be updated in response to the deletion.
- 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.