

Chan Jun Ren - Project Portfolio for StudyBuddyPro

About The Project

My team of 4 Computer Science students and I were tasked with enhancing a basic command line interface desktop address book application for our CS2103T module, and we chose to modify it into a 3-in-1 application that serves as a tool to aid revision called StudyBuddy. This application allows students to store and utilize flashcards, create notes that can be later used to generate a reference to create a cheat sheet.

This is what our project looks like:

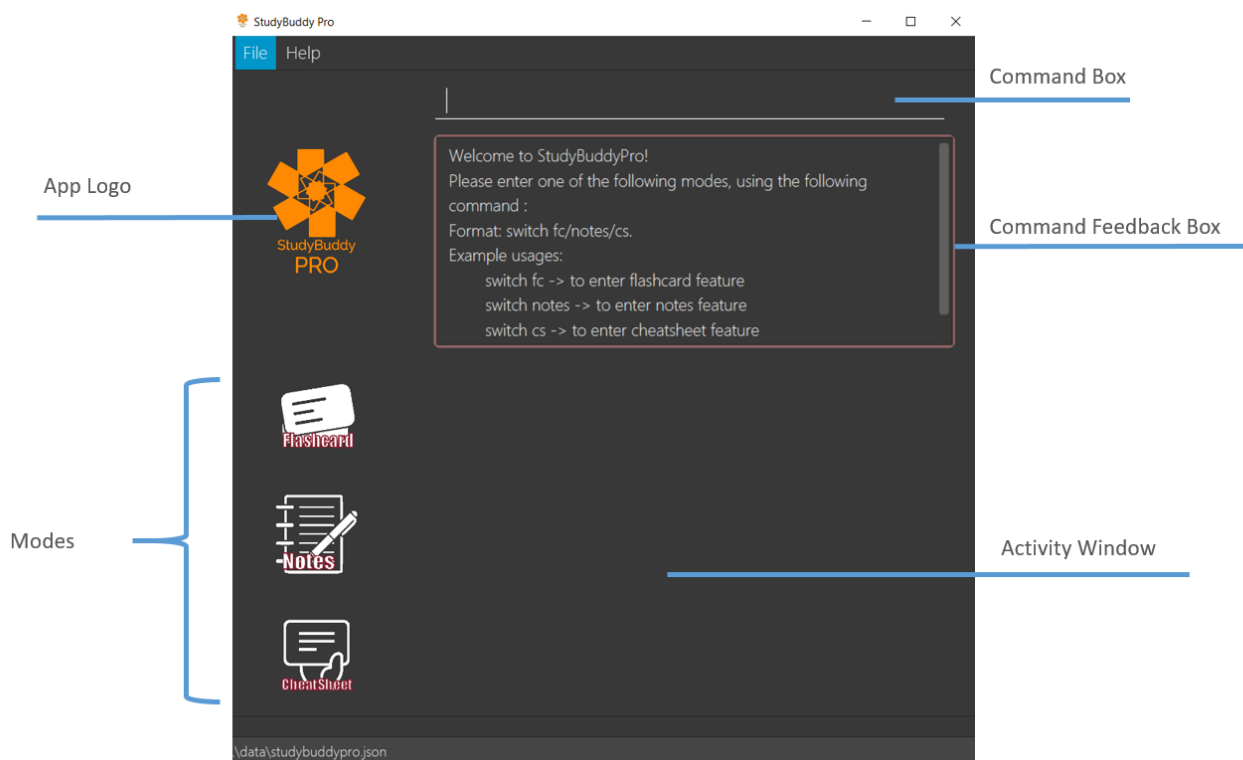


Figure 1. The graphical user interface for **StudyBuddyPro**.

Syntax

Please do take a look at the syntax notations below as they are used throughout the document.

syntax 1 — For commands or class objects

syntax 2 — For figures, tables, functionality or variables

Callouts Signs

Please do refer to the signs below as they are used throughout the document.

WARNING

Indicates information that are to be adhere as potential problems may be encountered if you are not careful.

IMPORTANT

Indicates information that are crucial to understand so that you will be able to follow the flow of the document.

NOTE

Indicates information that are note-worthy. Do read them for more information and better understandings.

Summary of contributions

- **Major enhancement:** added **ability to start a time trial of flashcards**.
 - What it does: The `timetrial` command initializes a test whereby the user will be tested a sequence of flashcards.
 - Justification: Pushes the user to test his understanding / familiarity of a certain topic.
 - Highlights: This enhancement works with existing as well as future commands. An in-depth analysis of design alternatives was necessary to implement the feature in a way that doesn't disrupt other features. The implementation was also challenging because there was a need to read up on an existing API that I did not have prior experience in.
 - Credits: Usage of the timeline api and linking the timer to a label was inspired by: <https://asgteach.com/2011/10/javafx-animation-and-binding-simple-countdown-timer-2/>
- **Minor enhancement:** Designed the overall GUI of the StudyBuddyPro application.
- **Code contributed:** [[Functional code](#)] [[Test code](#)] {give links to collated code files}
- **Other contributions:**
- Project management:
 - Facilitated discussion of the overall direction of the application in v1.1.
- Developing existing features:
 - Implemented the switching of modes and hooking up the activity window to the respective modes([#162](#)).
 - Implemented the viewing and showing of flashcards([#133](#))([#200](#)).
 - Implemented the time trial feature of the Flashcard([#198](#)).
- Enhancements to existing features:
 - Designed the overall GUI of the StudyBuddyPro([#123](#))
 - Enhanced delete commands to prompt the user once to confirm his/her deletion([#244](#)).
 - Wrote additional tests for existing features to increase the test coverage.

- Documentation:
 - Added the command summary for the team([#338](#)).
- Community:
 - Reported bugs and offered suggestions for the other team during the Practical Exam dry run.

Contributions to the User Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

Time Trial: `timetrial`

Starts a time trial for flashcards with specified tags <TAG>.

Format: `timetrial TAG...`

Example usage: `timetrial cs2103tum1 hard`

Expected output: Time trial started

- At least one tag must be specified.
- If more than one tag is specified, selects all flashcards that contains all of the specified tags.
- Default <TIME> will be 5 seconds.
- Answer will be flashed for 3 seconds.
- If a flashcard command (other than `show`) is inputted during the time trial, the time trial will be terminated and the inputted command will be executed.
- Executing `show` will reveal the answer of the flashcard in advance, but will not terminate the time trial.

Command Summary

Global Commands (Can be executed in any mode)

- **Switch** : `switch MODE`
e.g `switch fc`
- **Filter All** : `filterall tag/TAG...`
e.g `filterall tag/cs2103tum1 tag/difficult`

- **List tags** : `taglist`
- **Help** : `help`
- **List** : `list`
- **Exit** : `exit`

Flashcard Commands

- **Add** : `add q/QUESTION a/ANSWER t/TITLE [tag/TAG]...`
e.g. `add q/What is 100 Binary in its Decimal form? a/4 t/Binary Stuff tag/CS2100`
- **Delete** : `delete INDEX` e.g `delete 1`
- **Filter** : `filter tag/TAG...`
e.g `filter cs2103uml`
- **Time Trial** : `timetrial TAG...`
e.g `timetrial cs2103t uml`
- **View** : `view INDEX`
e.g `view 1`
- **List** : `list`
- **Show** : `show`
- **Remind** : `remind`

Note Commands

- **Add** : `add t/TITLE c/CONTENT tag/TAG...`
e.g. `add t/Pipelining Definition c/Pipelining is a process where a processor executes multiple processes simultaneously. tag/cs2100`
- **Delete** : `delete INDEX` e.g `delete 1`
- **View** : `view INDEX`
e.g `view 1`
- **Viewing a raw note** : `viewraw INDEX`
e.g `viewraw 3`
- **Filter** : `filter tag/TAG...`
e.g `filter tag/hard tag/cs2100`
- **List** : `list`

CheatSheet Commands

- **Add** : `add t/TITLE [tag/TAG]...`
e.g. `add t/CS2100 Midterm CheatSheet tag/cs2100midterm`
- **Delete** : `delete INDEX` e.g `delete 1`
- **Edit** : `edit INDEX t/TITLE tag/TAG...`

e.g `edit 8 t/cs2100 final cheatsheet tag/formula`

- **Show** : `show INDEX`
e.g `show 4`
- **View** : `view INDEX`
e.g `view 1`
- **Filter** : `filter tag/TAG...`
e.g `filter tag/hard tag/cs2100`
- **List** : `list`

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.

Flashcards Time Trial Feature

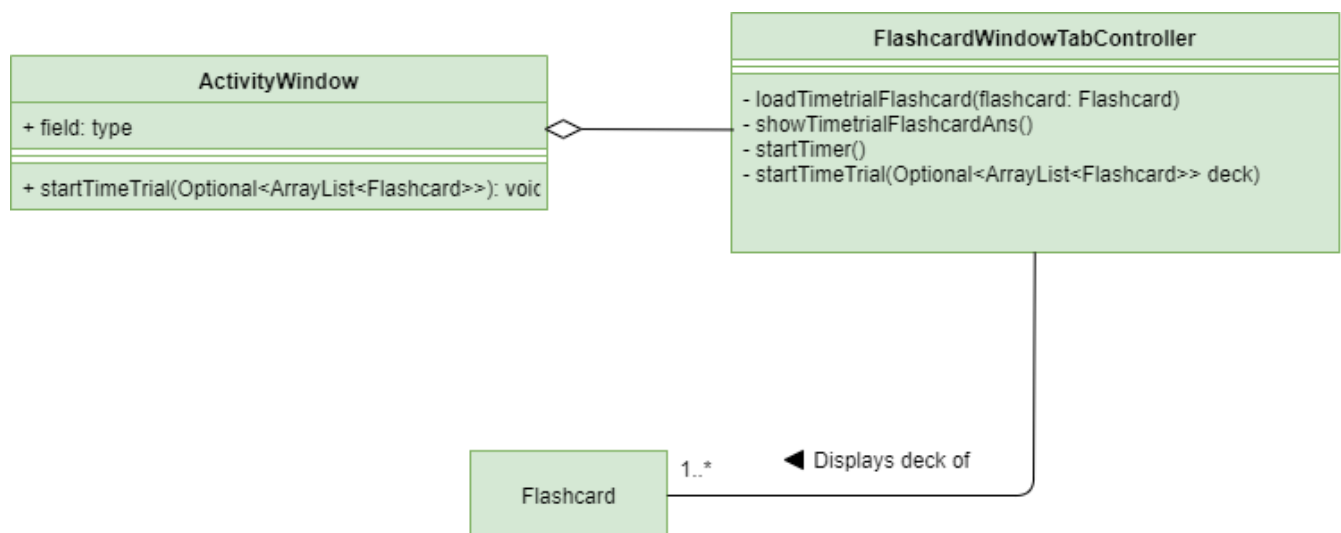
IMPORTANT

The following commands assume that the user is in the *flashcard* mode.

Implementation

1. The time trial mechanism is facilitated by the `FlashcardTabWindowController`, and mainly uses the `Timeline`, `KeyFrame` and `KeyValue` class from the JavaFX package to support its functionality.

The following *figure* shows a class diagram of the relevant classes of the time trial feature.



Class diagram when the `timetrial` command is executed

1. Given below is an example usage scenario and how the time trial mechanism behaves at each step.
2. Upon initialization of the StudyBuddy and switching to the Flashcard window, the `StudyBuddyParser`'s function enum will be set to parse `Flashcard` commands.
3. The user executes (`timetrial cs2100`), and the `StartTimeTrialCommand` retrieves a List of flashcards

with the associated `Tag` through the `Model#getTaggedFlashcards`, which is then passed into the `FlashcardTabWindowController`.

4. The `FlashcardTabWindowController` then calls the `FlashcardTabWindowController#startTimeTrial`, which in turns construct a `Timeline` with the following added for 3 flashcards:
 1. A `KeyFrame` to call the `FlashcardTabWindowController#loadTimeTrial` method, which displays the question of the flashcard on the window, with a `KeyValue` that starts the timer on the screen.
 2. A `KeyFrame` to call the `FlashcardTabWindowController#showFlashcardAns` method, which hides the Timer and flashes the answer of the flashcard for a set period of time.
 3. A `KeyFrame` is then added to the timeline to call the `FlashcardTabWindowController#resetViews` method, which in turn empties the `qnsTextArea` and `ansTextArea`. [TO BE REFORMATTED]

The following *figure* shows the sequence diagram of when the command `timetrial cs2103t` is executed:

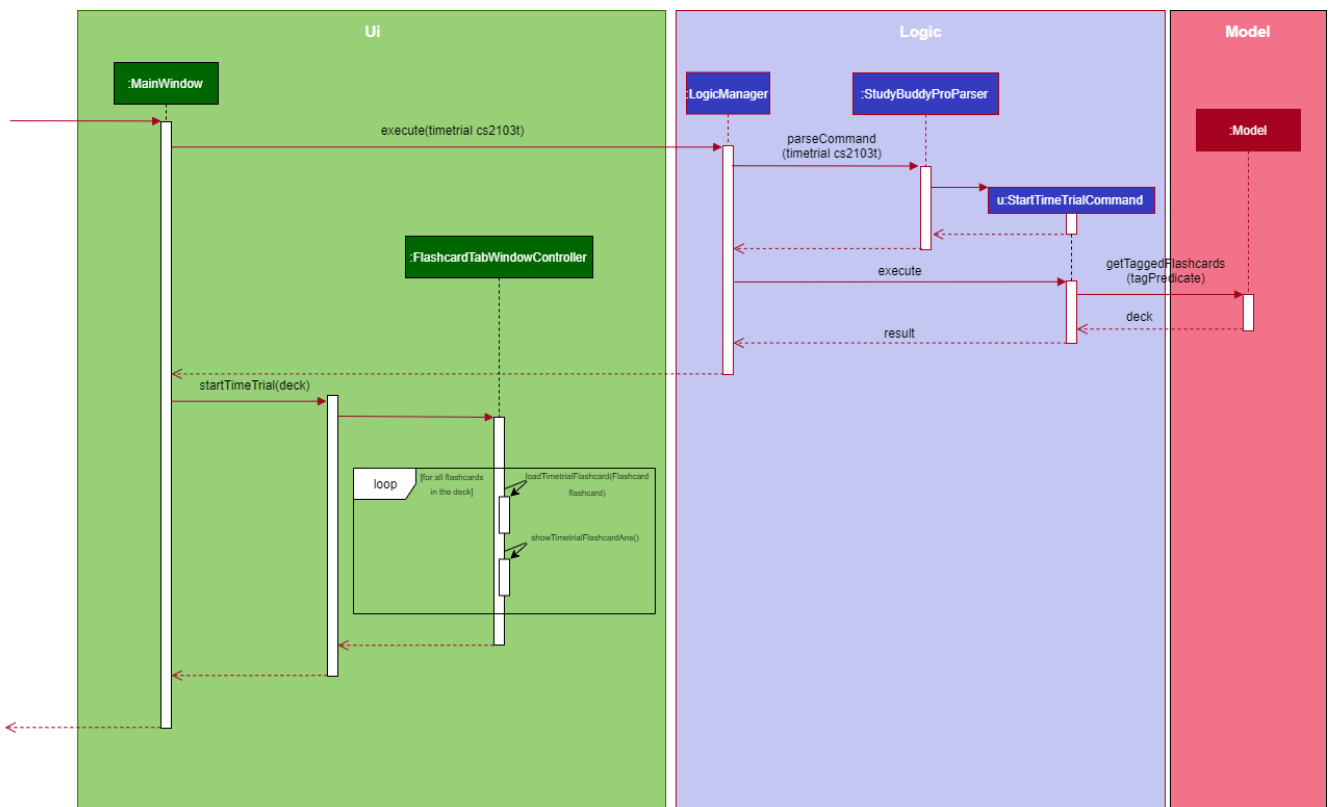


Figure 1. Sequence diagram when the `timetrial` command is executed

The following *figure* is an activity diagram that summarizes the flow of events when a user attempts to start a time trial as described above:

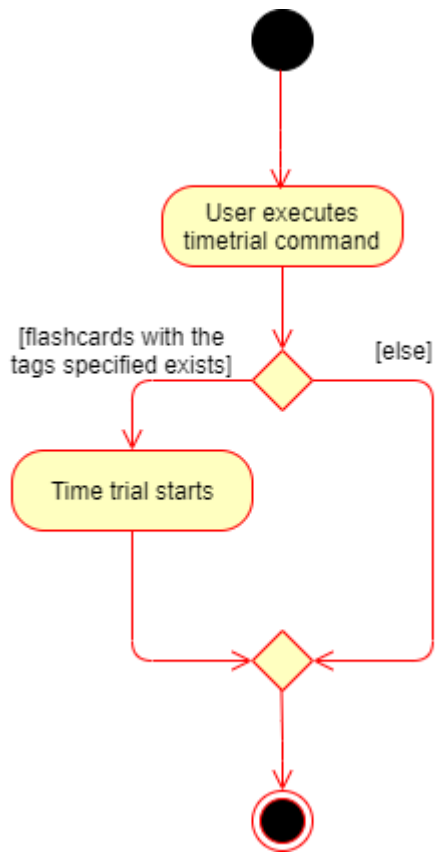


Figure 2. Activity diagram when the timetrial command is executed

Design Considerations

Aspect: How the timetrial is implemented

- **Alternative 1 (current choice):** 1. Using the `TimeLine` class to set the timer object.
 - Pros: Tidier and easier to understand.
 - Cons: Have to read up on the API and learn about the relevant classes such as `KeyFrame` and `KeyValue`
- **Alternative 2:** Looping `Thread.sleep()` to set the timer
 - Pros: Easier to implement
 - Cons: Code will be messier and harder to read

Aspect: How to continue the time trial

- **Alternative 1 (current choice):** Each flashcard and its' respective answer is displayed for a set period of time before the next flashcard
 - Pros: Easier to implement
 - Cons: Inflexible as user can only view the answer for a set amount of time
- **Alternative 2:** Allowing users to input commands to display the flashcard answer / move on to the next flashcard
 - Pros: Better flow of time trial feature and improved user experience
 - Cons: Hard to implement

[Proposed] Future improvements

- Allowing users to set their own time limit for each flashcard in the time trial mode
 - Command will be inputted to set the duration of the timer for each flashcard
- Allowing users to decide when to move on to the next flashcard
 - Question will still be shown for a fixed period of time, but a command will be required to move on to the next flashcard instead of just flashing the answer for a set amount of time

Testing **timetrial** command

NOTE

The following commands have been provided for ease of testing and can be inputted to test some of the test cases:

1. add q/Question 1 a/Answer 1 t/q1 tag/test
2. add q/Question 2 a/Answer 2 t/q2 tag/test

• Test case 1

a. Prerequisites

- i. Currently in flashcard mode with the Flashcard icon highlighted.
- ii. There exist flashcards with the tags that you would like to specify.

b. Input command: **timetrial** [tag]

c. Expected:

- i. Feedback box outputs:

Time trial started

- ii. The question of the flashcard is displayed together with a timer for 5 seconds.
- iii. Once the time hits 0, the answer is flashed for 3 seconds.
- iv. Repeat 2 and 3 until all of the flashcards of the specified tags have been displayed.

• Test case 2

a. Prerequisites

- i. Currently in flashcard mode with the Flashcard icon highlighted.
- ii. There exist flashcards with the tags that you would like to specify.

b. Input command: **timetrial** [tag].

c. Input command: **show**.

d. Expected:

- i. Feedback box outputs:

Flashcard answer loaded

- ii. The answer of the flashcard is loaded, and after a while the time trial continues until all the flashcards have been loaded.

- Test case 3

- a. Prerequisites

- i. Currently in flashcard mode with the Flashcard icon highlighted.
 - ii. There exist flashcards with the tags that you would like to specify.

- b. Input command: `timetrial [tag]`.

- c. Input command: `view 3` (Can be any other flashcard command).

- d. Expected:

- i. Feedback box outputs:

```
Viewing flashcard: Title: (Flashcard title)
Tags: [tags]
```

- ii. Time trial does not continue and the latest command inputted is executed.

- Test case 4

- a. Prerequisites

- i. Currently in flashcard mode with the Flashcard icon highlighted.
 - ii. There are no flashcards with the tags that you would like to specify.

- b. Input command: `timetrial [tag]`.

- c. Expected:

- i. Feedback box outputs:

```
There are no flashcards with the tags specified!
```

PROJECT: StudyBuddyPro
