Tiffany Chong - Project Portfolio for TopDeck

1. About the Project

For our Software Engineering module CS2103T, our team was tasked to modify or enhance the addressbook program into our software of choice. We morphed it into TopDeck, a general purpose flash card application, which allows users store flash cards in decks, and study these flash cards later.

My role was to design the Study Sessions. The following sections will describe my contributions in greater detail, as well as the sections I've added to the user and developer guides with regards to these enhancements.

2. 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 Study View to study a deck of flash cards.

- What it does: The study session allows users to study a deck of cards. This is done by presenting users with random questions from cards in a deck. Once the users have recalled their answers, they can choose to reveal the answers. Cards do not repeat until the whole deck has been viewed.
- Justification: In order for a flash card application to aid users in memory tasks, there has to be a mode where the users can test their own knowledge. This mode allows users to randomise viewing of questions and then choose when to see the answers.
- Highlights: The ability to generate flash cards are found in a separate class, making it easy for smarter study algorithms to be implemented in future. Several designs were considered to ensure that the display would have minimal interactions with the internal state of the program. The feature was challenging to implement because it entails stateful interactions. Another challenge was creating a UI that made it intuitive for users to know whether they were viewing questions or answers.

Code Contributed: Please view our github page to see the code that I have contributed.

Other contributions:

- Project management:
  - There were a total of 4 releases, from version 1.1 to 1.4. I managed releases versions 1.2 to 1.3 (2 releases), together with the mid-versions (mid 1.2 and mid 1.3) on GitHub. This includes compiling the requirements needed at each stage, ensuring that all issues are resolved before each release, as well

as building and releasing the binary.

- I proposed a few meetings in order for our team to settle our roles before the release.

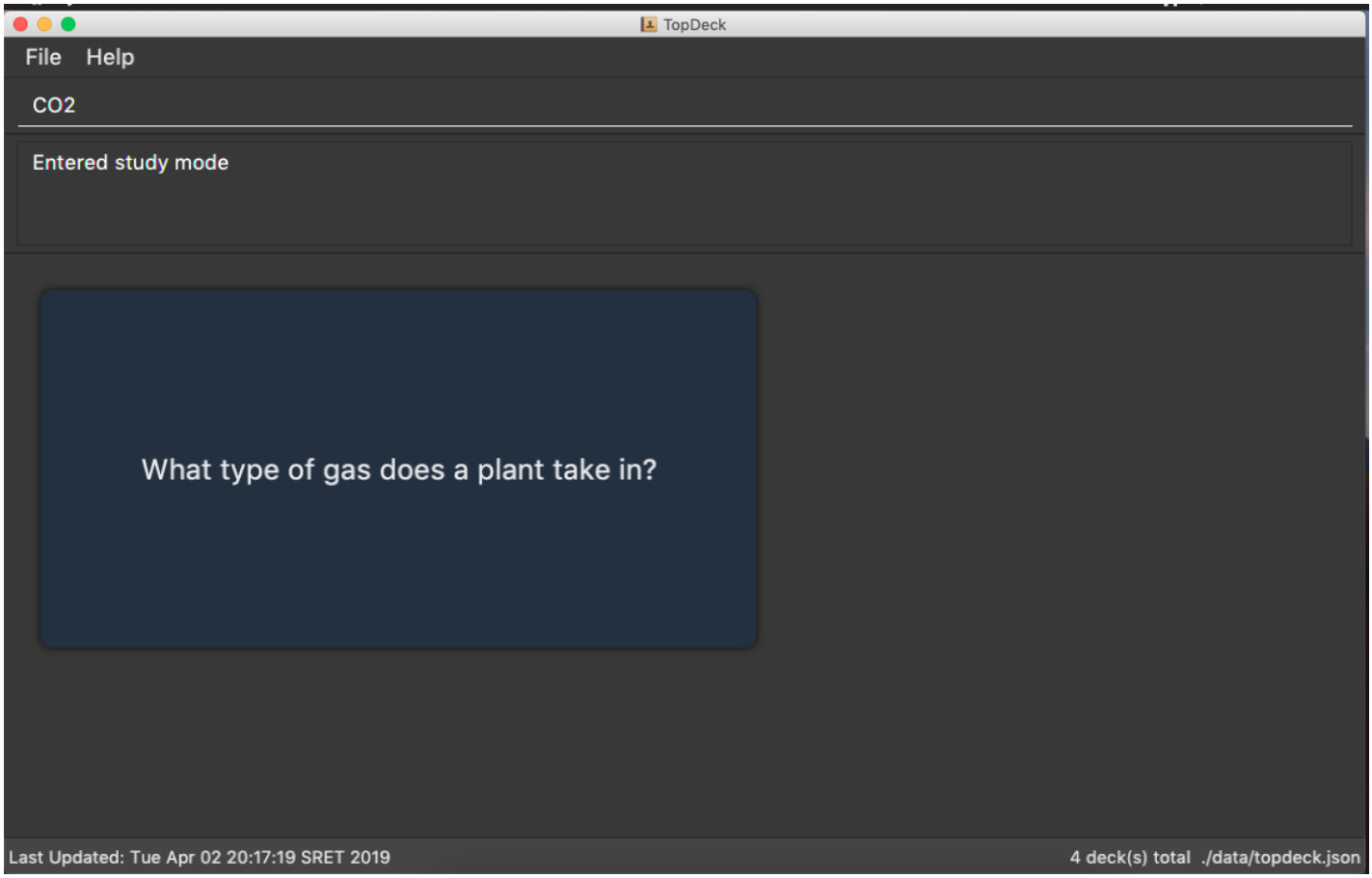
- Enhancements to existing features:
  - I added the ability to switch between Cards/Deck view to Study view and vice versa
  - I resolved checkStyle issues
- Documentation:
  - I made slight changes to the introduction to make it more comprehensible for non-technical users.
  - I alphabetised and split our Command Summary based on relevant categories.
- Community:
  - I reviewed pull requests for deck and card testing, and undoing of Card commands.
  - I contributed to finding the reasons why the Undo functionality was not working for Card Commands.
  - I contributed to planning how the final UI should look like.
  - I put up issues and enhancements to keep our team on track.
  - I helped my friends add screenshots of our UI for our documentation.
  - I gave our team a few tips on the IntelliJ workflow.
- Tools:
  - I made the project RepoSense Compatible

3. Contributions to User Guide

3.1. Study View

In this view you can study a deck of cards.

- Test your knowledge of the cards in your chosen deck
- Rate the difficulty of the cards.



3.2. Study View

These are commands are only available in study view, after `study INDEX` or `study` command is executed.

3.2.1. Returning to decks view: back

Format: `back`

Outcome: Returns to decks view.

3.2.2. Opening the deck in cards view: deck

Format: `deck`

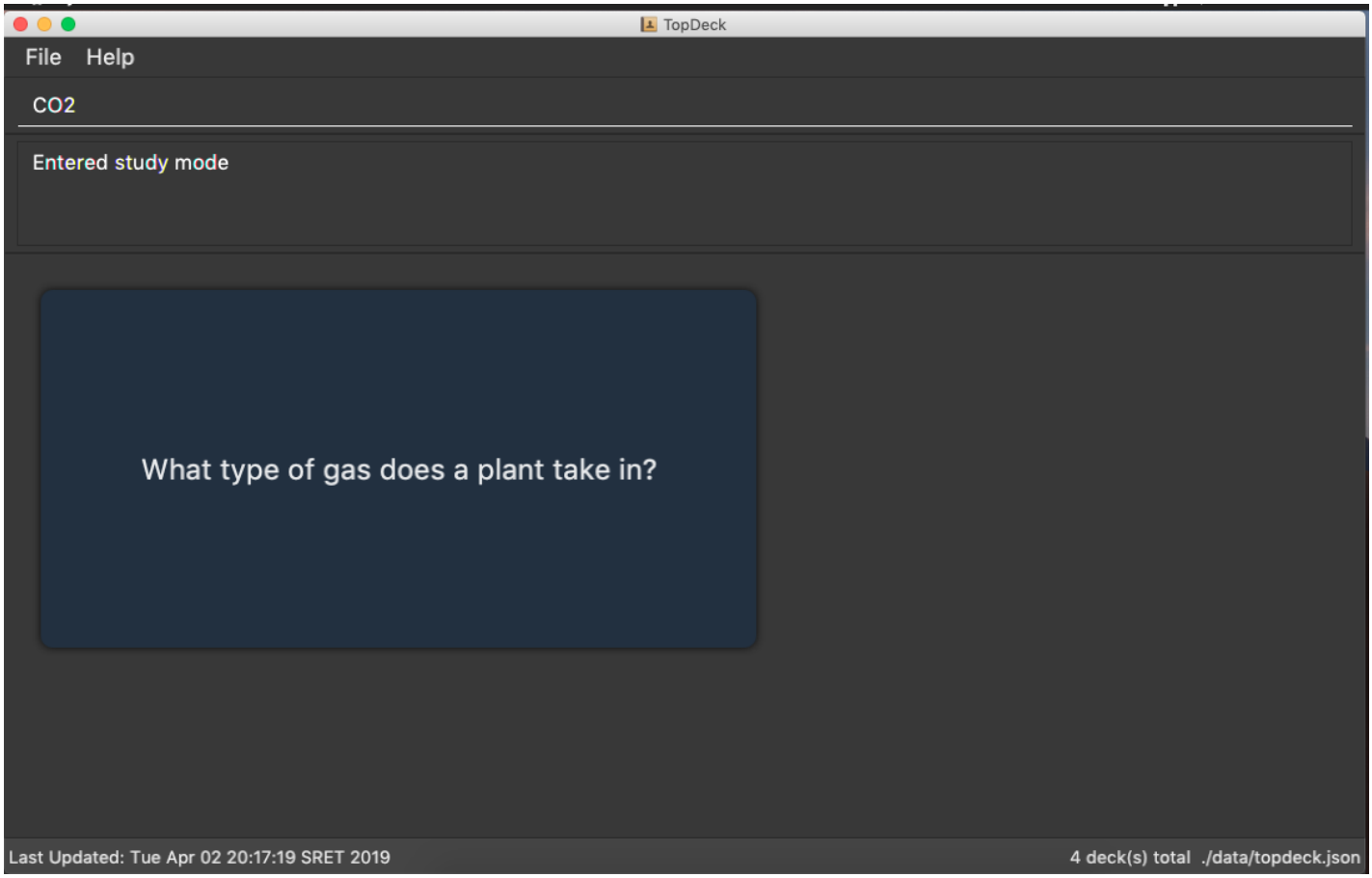
Outcome: Opens the deck in cards view.

3.2.3. Revealing the answer to card: [any attempt]

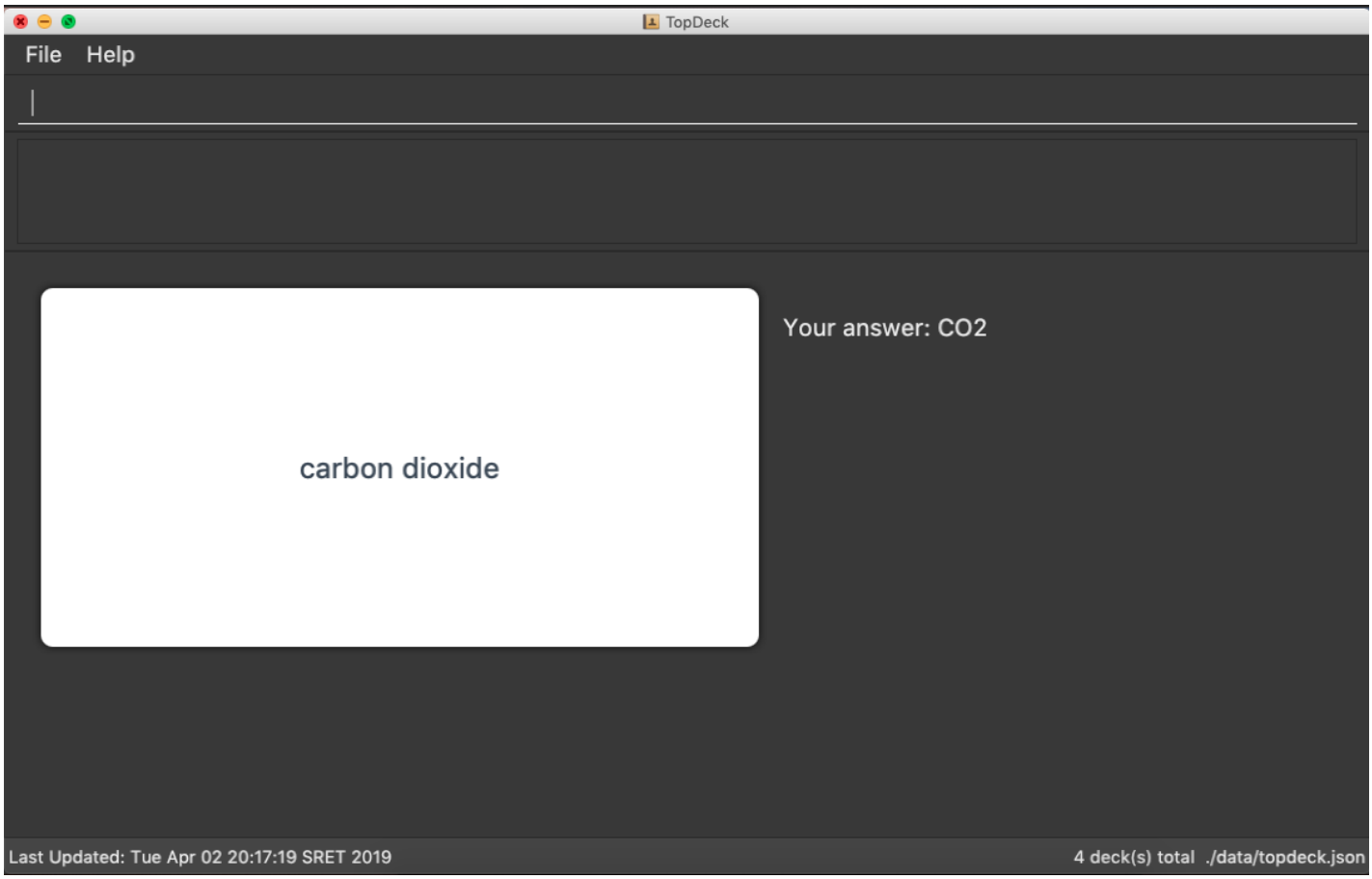
Format: `[any attempt]`

Outcome: Reveals the answer to card, with your attempt shown beside it.

For example, you can type in `CO2` and hit `Enter`



The following image is the result of the command.

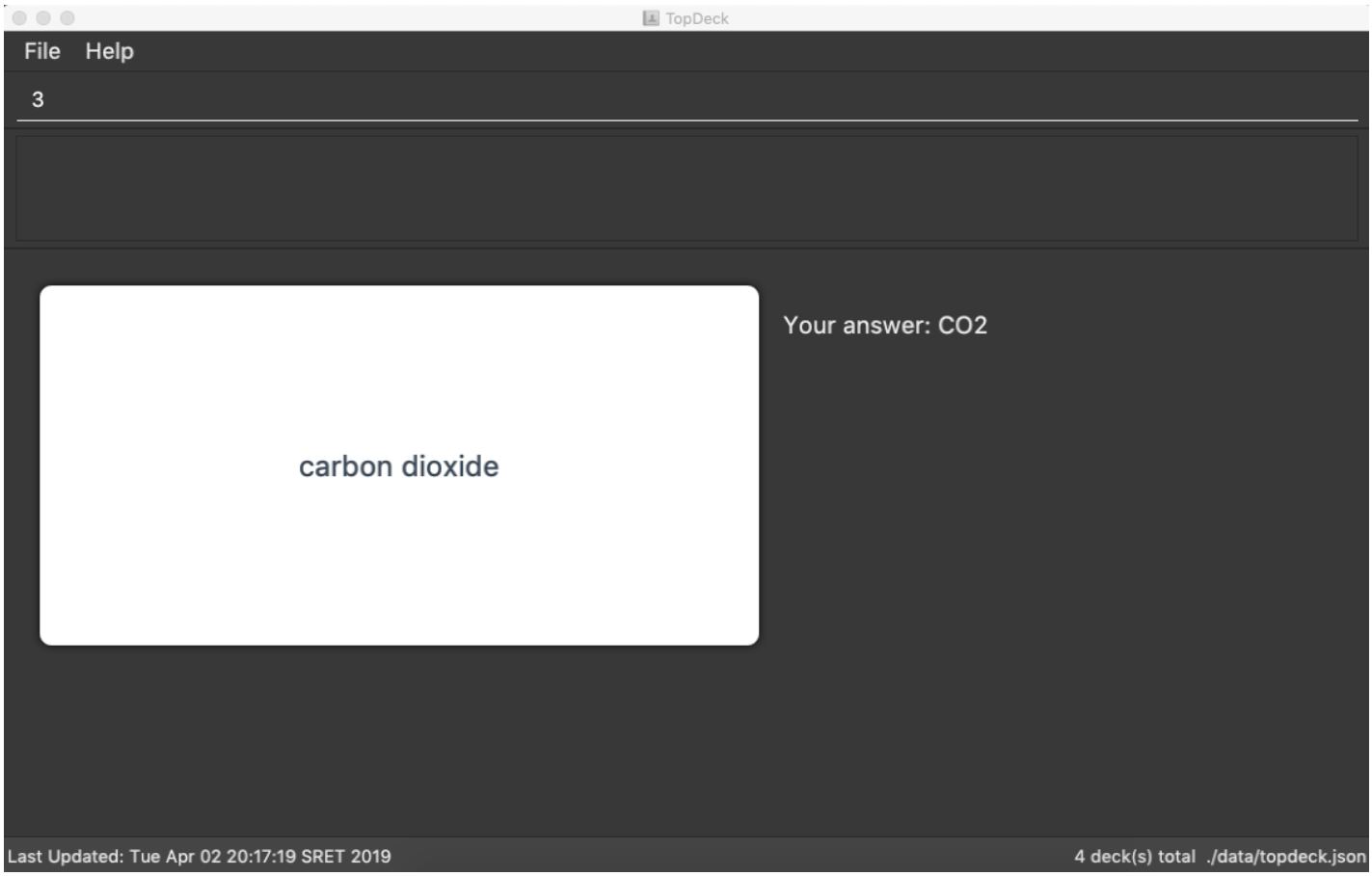


3.2.4. Displaying the next card: `[your difficulty rating]`

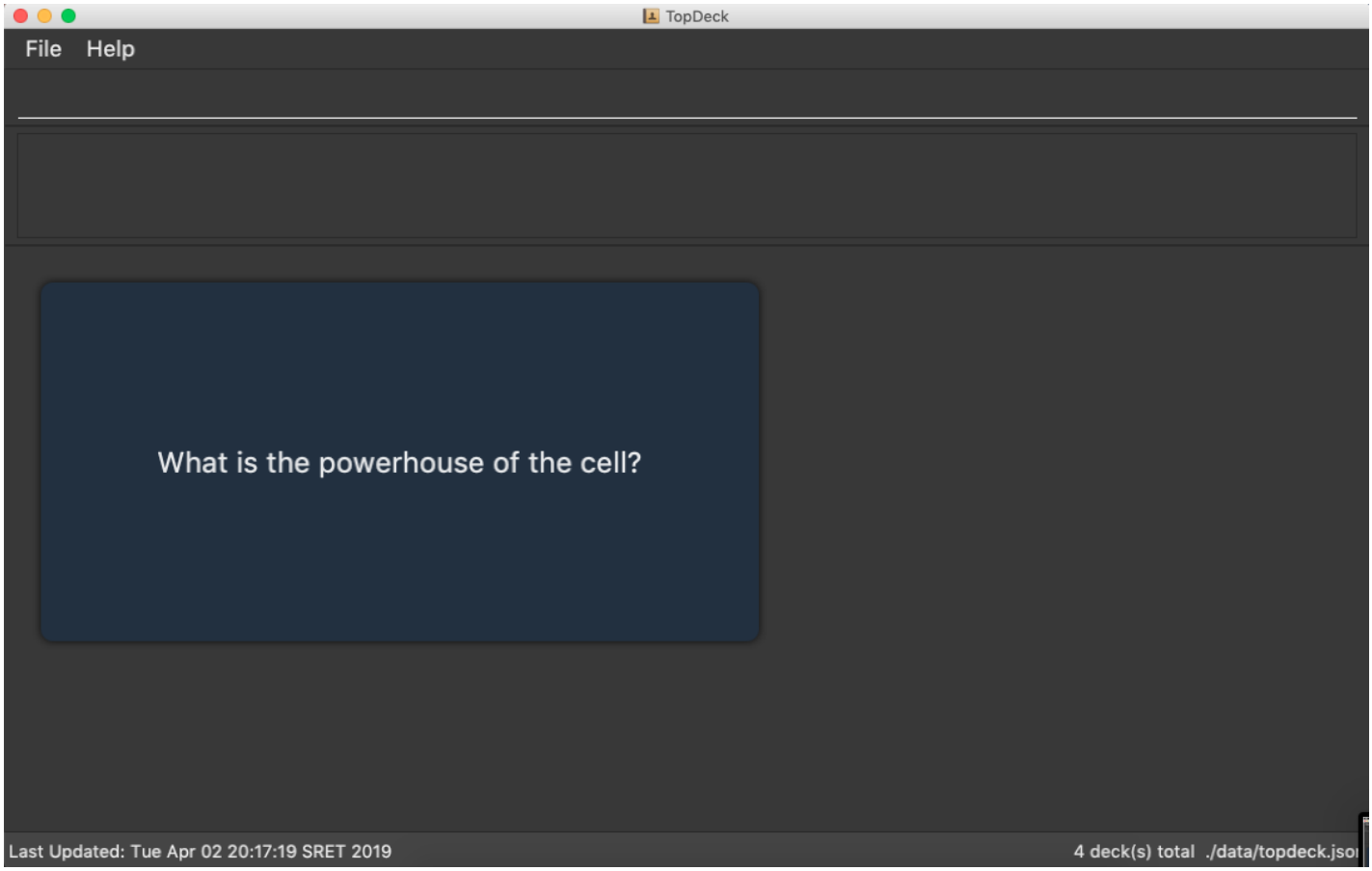
Format: `[your difficulty rating]`

Outcome: Displays another card from the deck. Stores your difficulty rating for the card.

Entering a number 3 indicates that you rate the difficulty as 3.



Submitting this reveals the next question.



4. Contributions to Developer Guide

4.1. Study view

4.1.1. Current implementation

The purpose of the third view, study view, is to help users retain the knowledge of their flash cards.

This is done by consecutively presenting users with random cards from the deck. For each card, users are able to see the questions first, and then reveal the answers afterwards.

The functionality above is supported by the `StudyView` class.

Important Variables

This `StudyView` class two important variables:

- `currentCard` - the card which is currently being shown to the user.
- `currentStudyState` - variable which stores a `studyState` enum. The enum can be either `QUESTION` or `ANSWER`

These two important variables are continuously being altered every time the user interacts with the program, explained in detail later.

Observable Values

Besides these variables, `StudyView` also makes extensive use of `ReadOnlyProperty` wrapper to store variables which the UI has to display. This wrapper is chosen as it implements the `Observable` interface.

4.1.2. Data structure implementation

This section discusses how the two variables, `currentCard` and `studyState` are continuously modified, and also the other variables that change as a result of these two functions.

Given below is an example usage scenario and how the `study` mechanism behaves at each step.

Step 1. User enter study mode using `study 1` command.

A `StudyDeckCommand` is issued.

Upon the execution of this command, Model's `viewState` will now hold a `StudyView` object with 1st deck as active deck.

As part of the execution of `StudyDeckCommand` , step 2 is ran.

Step 2. Generating a question.

In order to present questions to users, these are the functions called by the `GenerateQuestionCommand()` :

- `DeckSuffler#generateCard()` The `DeckShuffler` class has an algorithm (implementation explained later) to select a Card object, and passes this reference to `StudyView` .
- `currentCard` refers now to that card.
- `currentState` is set to QUESTION.
- Consequently, `textShown` property is also modified. In this case, it will contain the question of the chosen card.

Step 2. User executes the `[wildcard]` command to signal that he has come up with the answer.

- User answer, ie whatever the user wrote, is stored as command under the statistics variable.

Step 3. The command generates a `ShowAnswerCommand()`

The follwing are ran upon execution of the show answer command:

- `currentStudyState` to ANSWER.
- Similar to the above, we also change the `textShown` property to show user answer instead.

Step 4. User enters the difficulty rating "command" to rate how difficult they thought it was to recall the answer.

The following things happen:

- Their answer stored under userAnswer variable, which will be displayed later.
- This is perfect as we do not want users to revert statistics by executing undo command.
- `GenerateQuestionCommand()` is executed once again. Back to Step 2.

4.1.3. UI implementation

The UI listens for three things: the `studyState`, `userAnswer`, and `textShown`.

The first is important because depending on whether it's question or answer, we modify the colours of flash cards in the UI. This variables also specifies whether or not we need to display users' answers and whether or not we need to prompt them to rate the difficulty.

The second is important to show the users what they answered. The last is important to show the contents of the flash card.

4.1.4. Design considerations

Aspect: How to store states within study view

- **Alternative 1 (current choice):**
  - Pros: Easy to implement.
  - Cons: UI just changes state based on commands issued.
- **Alternative 2: (current choice):** Use an event listener to see whether the `textShown` has changed. Because it is easy to modify textShown immediately, there is no need for the UI to track this variable as that would be costly.

Last updated 2019-04-02 23:29:57 SGT