Russell Loh - Project Portfolio

Project Title: Njoy Teaching Assistant

Overview

Njoy Teaching Assistant is a command line interface(CLI) desktop application built by a team for 5 computing students for our Software Engineering project. It enables teachers of various disciplines to create a more interactive classroom through the use of slideshow and polling. Outside the classroom, it allows them to manage their classes, timetables, notes, questions and quizzes.

Below is an example image of the application:

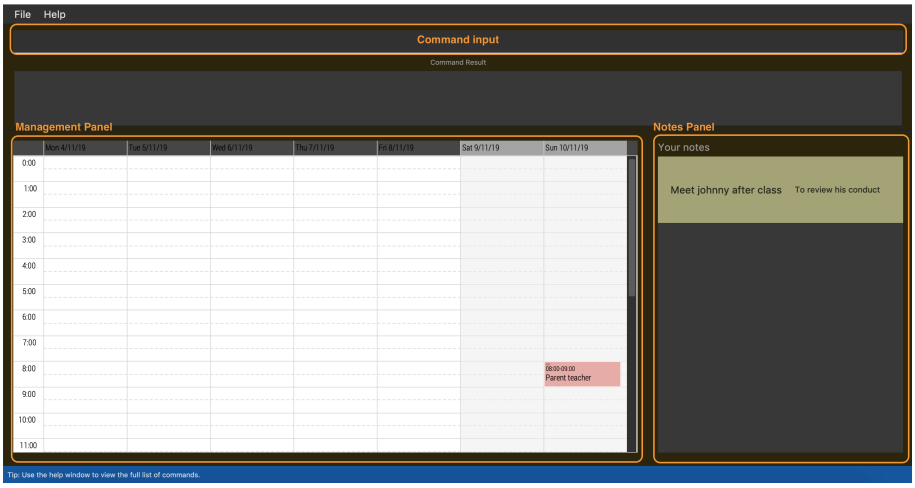

Figure 1. Njoy Teaching Assistant application.

I was tasked to brainstorm a solution that could enhance classroom learning and implement it. Eventually, I came up with the solution of using question prompts through a slideshow that is able to facilitate discussions in the classroom and at the same time, allow teachers to manage these questions more efficiently. The following sections includes my contribution to the project and selected key snippets of documentation that I wrote pertaining to my features for both the user and developer guides.

Certain special formatting are used in this document and their description is as follows:

Format	Description
`command`	Indicates a command, keyword or control for the user guide. For the developer guide, it indicates that it is a class, method or variable name.
command	Indicates a link to a section in the document or to a web page.

Summary of contributions

- **Major enhancement**: Added the ability for teachers to manage their questions.
    - **What it does:** Allows teachers to manage their questions more efficiently by providing a simple interface to create, edit, delete and find questions.
    - **Justification:** With teachers having to constantly prepare their questions before lessons and organise their questions for quizzes or homework, this feature will help to cut down the preparation time needed for such tasks.
    - **Highlights:** This enhancement is the basis for both the slideshow and quiz features and required an in-depth analysis on how teachers will use this feature, and how the questions can be presented in a simple yet intuitive manner. Moreover, it was challenging to adjust this feature such that it can be easily reused by both the slideshow and quiz features as they rely heavily on these questions.
- **Major enhancement**: Added the ability for teachers to start a slideshow
    - **What it does:** Allows teachers to start an in-class activity with their students through a slideshow of questions that has an additional polling feature to garner responses from students.
    - **Justification:** Slideshow reduces the need for teachers to constantly write questions on the whiteboard and at the same time, interact with students through the use of polling to question their understanding about a topic before revealing the answer.
    - **Highlights:** This enhancement adds additional purpose to existing questions and required an in-depth analysis on how typical presentation software works so that using the slideshow feature will not feel foreign to the teachers. It was challenging to ensure that the controls are familiar to the teachers and the layout is simple enough for the students to view.
- **Minor enhancement 1**: Restyled the general application interface.
- **Minor enhancement 2**: Designed the **Njoy** logo and banner.
- **Code contributed**: [Functional and test code]
- **Other contributions**:
    - Project management:
        - Managed milestones and deadlines on Github.
        - Managed issue tracker and enforce feature assignment on Github.
        - Managed milestone version 1.3 release on Github.
    - Enhancements to existing features:
        - Wrote skeleton for commands and related prefix that the team can reuse. (Pull requests #32, #33)
        - Added command summary and adjusted application help guide user interface. (Pull request #121)
        - Refactored and removed address book related commands and parser. (Pull requests #106, #34)
        - Added tests for questions. (Pull requests #185, #189)

- Documentation:
  - Added documentation for features implemented. (Pull requests #185, #121, #106, #74)
  - Added first draft user guide, use cases and UI mockup to the developer guide. (Pull request #18)
- Community:
  - Team pull requests that I reviewed and merged. [Github]
  - Reviews done for other teams (Pull request #122)
- Tools:
  - Integrated a third party library (commons-text by Apache) to the project

---

Contributions to the User Guide

Given below are the sections that I contributed to the User Guide. However, only a few key snippets of my contribution are included, which showcase my ability to write user documentation targeting end-users. The full User Guide can be found here

Creating a question

Create and store questions according to the type specified.

Format: `question question/… answer/… type/… (MCQ OPTIONS IF APPLICABLE)`

> **NOTE** Note that for mcq question type, it is necessary to input keywords `a/`, `b/`, `c/`, `d/`.

The keywords supported by this feature includes:

Keyword	Description
`question`	Question topic.
`answer`	Answer to the question.
`type`	Type of question(`open` or `mcq`).
`a`	Option A for MCQ.
`b`	Option B for MCQ.
`c`	Option C for MCQ.
`d`	Option D for MCQ.

> **NOTE** Note that `answer` is free text and allows any input since answers may be accompanied with explanations. Therefore, for mcq questions, ensure that the answer entered reflects the correct option.

Examples:

- **Open ended question**: `question question/Which year did Singapore gain independence? answer/1965 type/open`
  Creates an open ended question with topic *'Which year did Singapore gain independence?'* with answer *'1965'*.
- **MCQ**: `question question/Which year did Singapore gain independence? answer/1965 type/mcq a/1965 b/1963 c/1968 d/1970`
  Creates an mcq with topic *'Which year did Singapore gain independence?'* with answer *'1965'* and choices *'1965'*, *'1963'*, *'1968'*, *'1970'*.

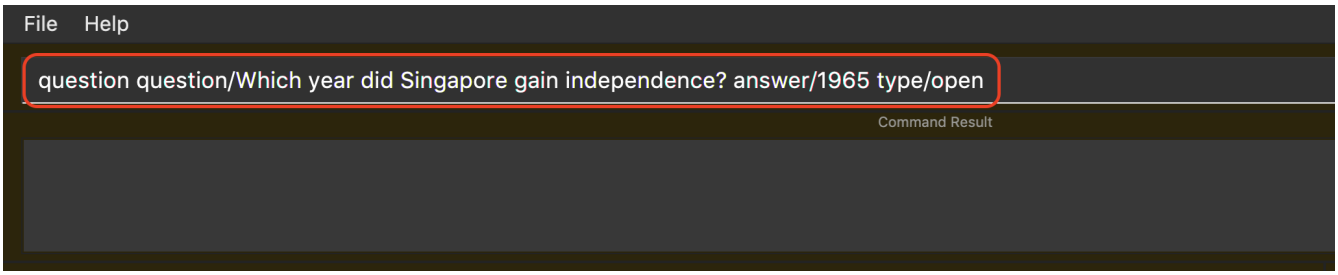

Figure 2. Creating an open ended question.

Starting a Slideshow

Start a slideshow based on the questions selected. The sequence of the questions displayed will follow the ordering of input.

Format: `question slideshow [QUESTIONS INDEX]`

The keywords supported by this feature includes:

Keyword	Description
`[QUESTIONS INDEX]`	Index of questions **separated by a whitespace**. Follows the index as defined in the list and find commands.

The following controls are used to navigate through the slideshow:

Control	Description
`Right arrow(→)`	Go to next question.
`Left arrow(←)`	Go to previous question.
`Esc`	Quit the slideshow.
`Space`	Show or hide the answer to the question.

The following controls are used for polling mcq type questions:

Control	Description
1	Add one response to option A.
2	Add one response to option B.
3	Add one response to option C.
4	Add one response to option D.

Example:

- `question slideshow 1 2 3`
  Starts a slideshow with questions containing index *'1'*, *'2'* and *'3'*.

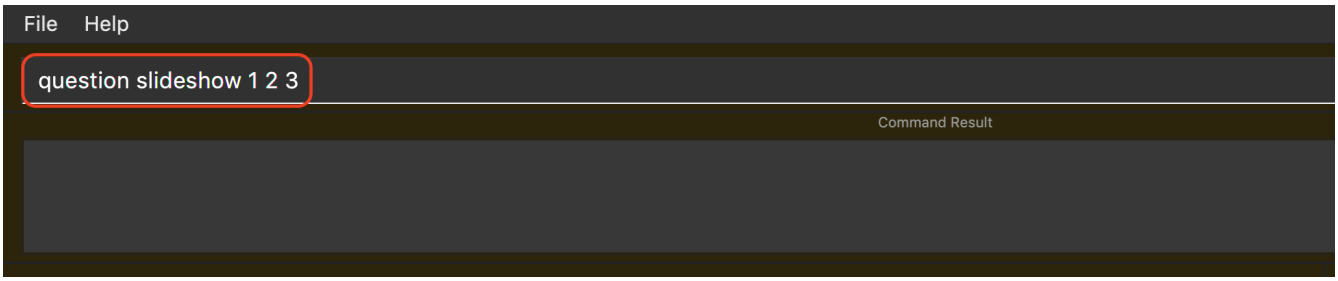

Figure 3. Starting a slideshow.

Below is an example of how to navigate through the slideshow and utilise the polling feature:

Step 1. When the slideshow window appears, scroll between the questions using the `Left arrow(←)` and `Right arrow(→)` keys.

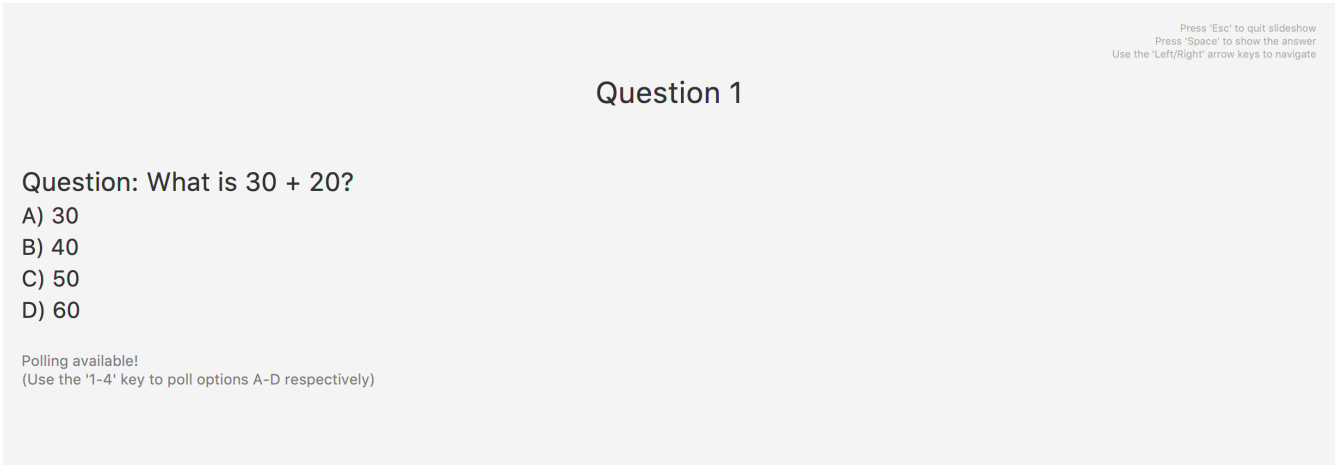

Figure 4. Slideshow window appears.

Step 2. Use the `Space` key to show or hide the answer. For mcq type questions, you may use the `1-4` keys to add to the poll results for options A to D respectively.

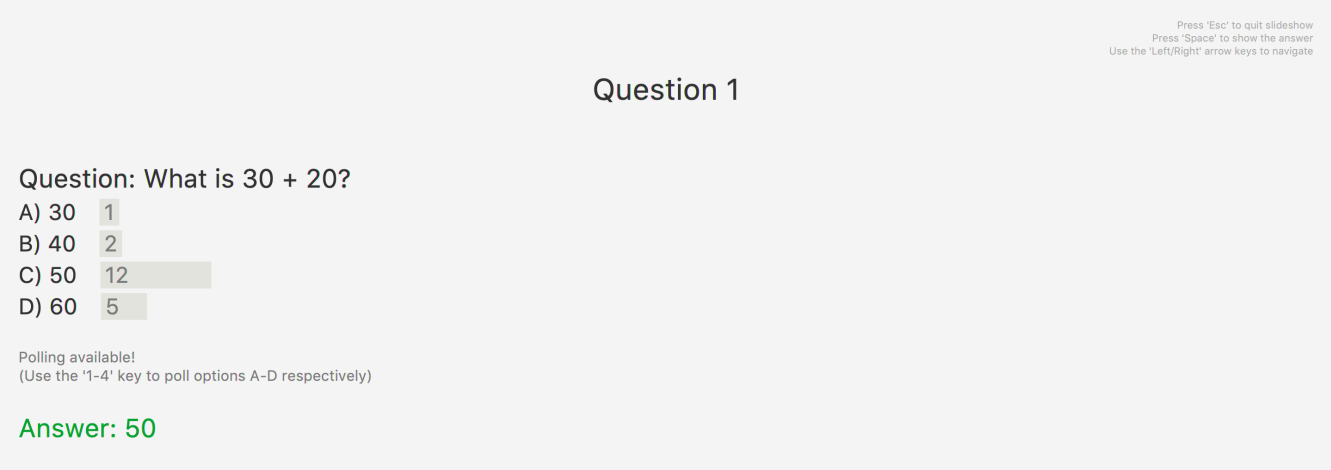

Figure 5. Answer shown with polling results.

Step 3. When you want to end the slideshow, use the `Esc` key to exit or navigate beyond the last question of the slideshow and it will automatically exit.

Here is a summary of the various components of the slideshow feature:

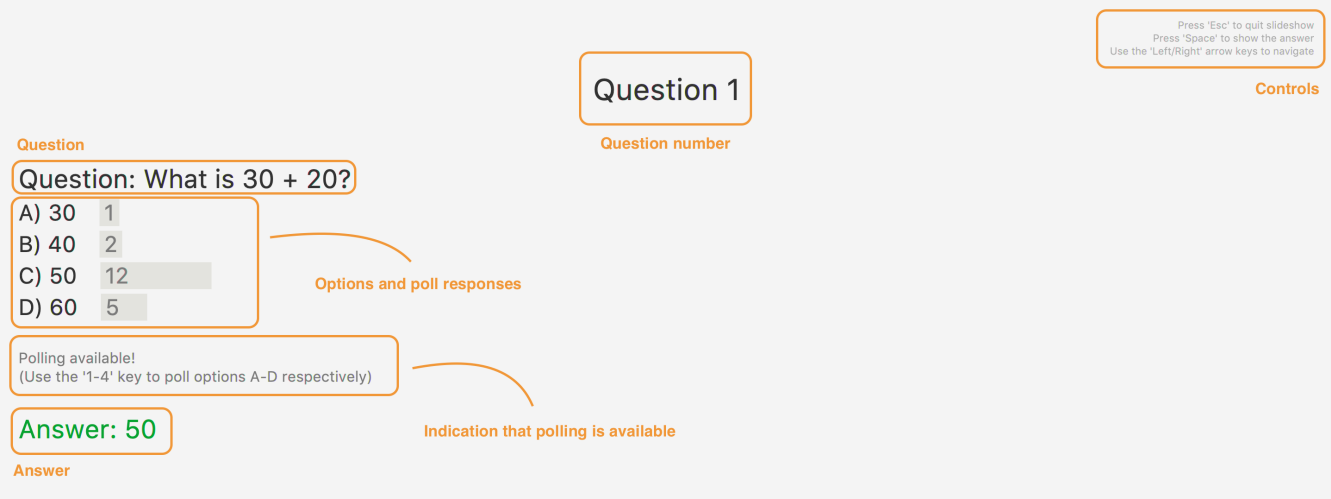

Figure 6. Description of the various components.

Contributions to the Developer Guide

Given below are sections I contributed to the Developer Guide. Similar to the User Guide, only a few key snippets of my contribution are included, which showcase my ability to write detailed and comprehensive technical documentation that can be referenced by fellow developers. The full Developer Guide can be found here

Find command

The following is a detailed explanation of the operations `QuestionFindCommand` performs.

NOTE Note that questions searched using this command has it's own `ObservableList` stored under `QuestionBank#questionsFiltered`.

Step 1. The `QuestionFindCommand#execute(Model model)` method is executed. No validation is necessary here since it does not write to the question list.

Step 2. The method `QuestionBank#searchQuestions(String textToFind)` is then called through the `Model#searchQuestions(String textToFind)` method.

Step 3. The existing `QuestionBank#questionsFiltered` is cleared in case there are existing questions from a previous search. A temporary `ArrayList<Question> similarAl` to store `Question` objects is also created to store similar questions.

Step 4. The `QuestionBank#questions` list is iterated once and the search is performed on the user's search term using 2 levels of searching. Firstly, the question is tested to see if it matches the search term using the `StringUtils.containsIgnoreCase(…)` method. Next, if the search term is not found, we test the question to see if it is similar to the user's search term using the `LevenshteinDistance` method that implements the Levenshtein distance formula with a threshold of 40 percent (See [Feature-Question-Design-Similarity]).

Step 5. The questions are then duplicated with their index appended to the question in order to keep a separate reference from the main `QuestionBank#questions` list. Questions that matches the search term will be added to the `QuestionBank#questionsFiltered` list whereas for similar questions, they will be added to the temporary `similarAl` list instead.

Step 6. Both the `QuestionBank#questionsFiltered` and `similarAl` list is then sorted in ascending order of their question length.

Step 7. The `similarAl` list is then appended to the `QuestionBank#questionsFiltered` list.

Step 8. A new `CommandResult` will be returned with a message stating the search term and the number of results returned.

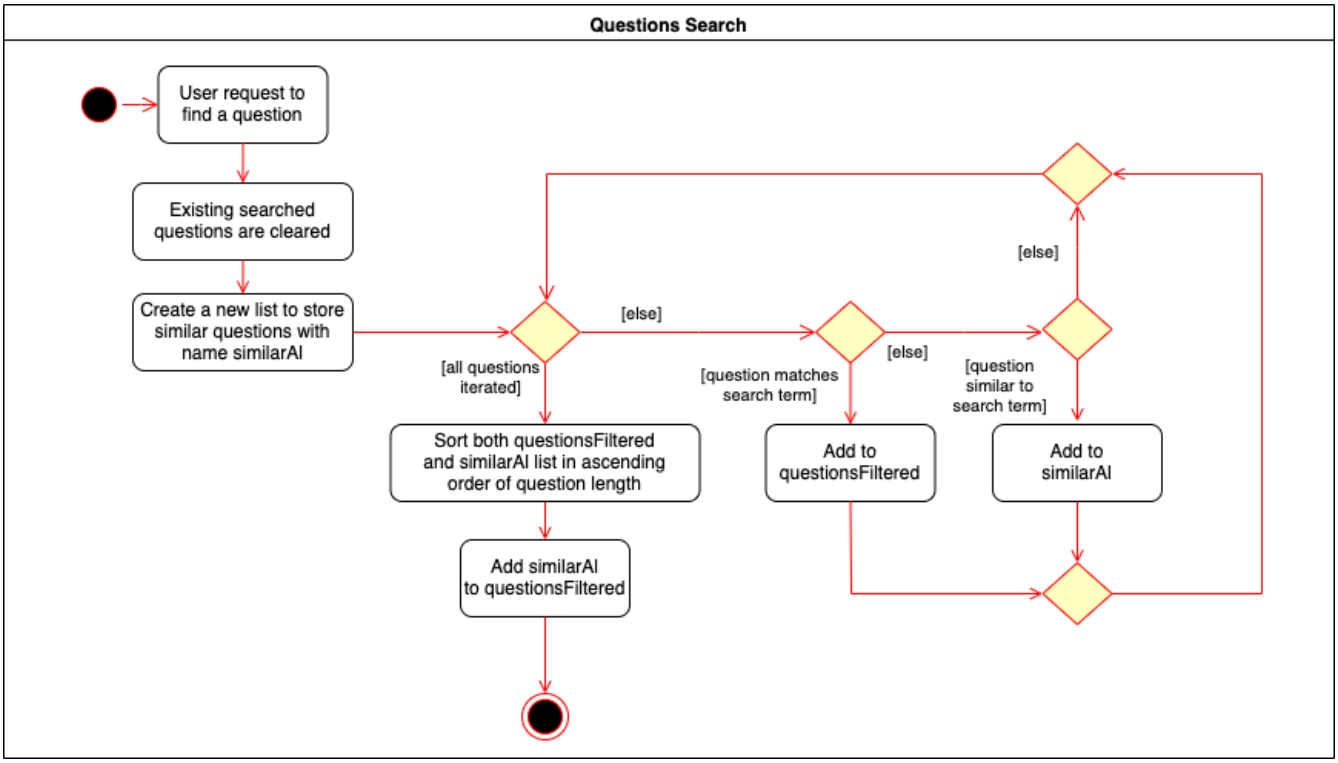

Figure 7. Activity diagram of questions search.

Slideshow feature

The slideshow feature is dependent on the questions added by the `question slideshow [QUESTIONS INDEX]` command and interacts with the `ModelManager` to retrieve the list of questions to be displayed in the slideshow. The logic control for displaying the ui resides in the `SlideshowWindow` class and handles the controls and instantiation of the various `QuestionPanel` that contains each question.

Below is the sequence diagram of the interactions that happen from when the slideshow command is entered, to the corresponding questions displayed in the slideshow.

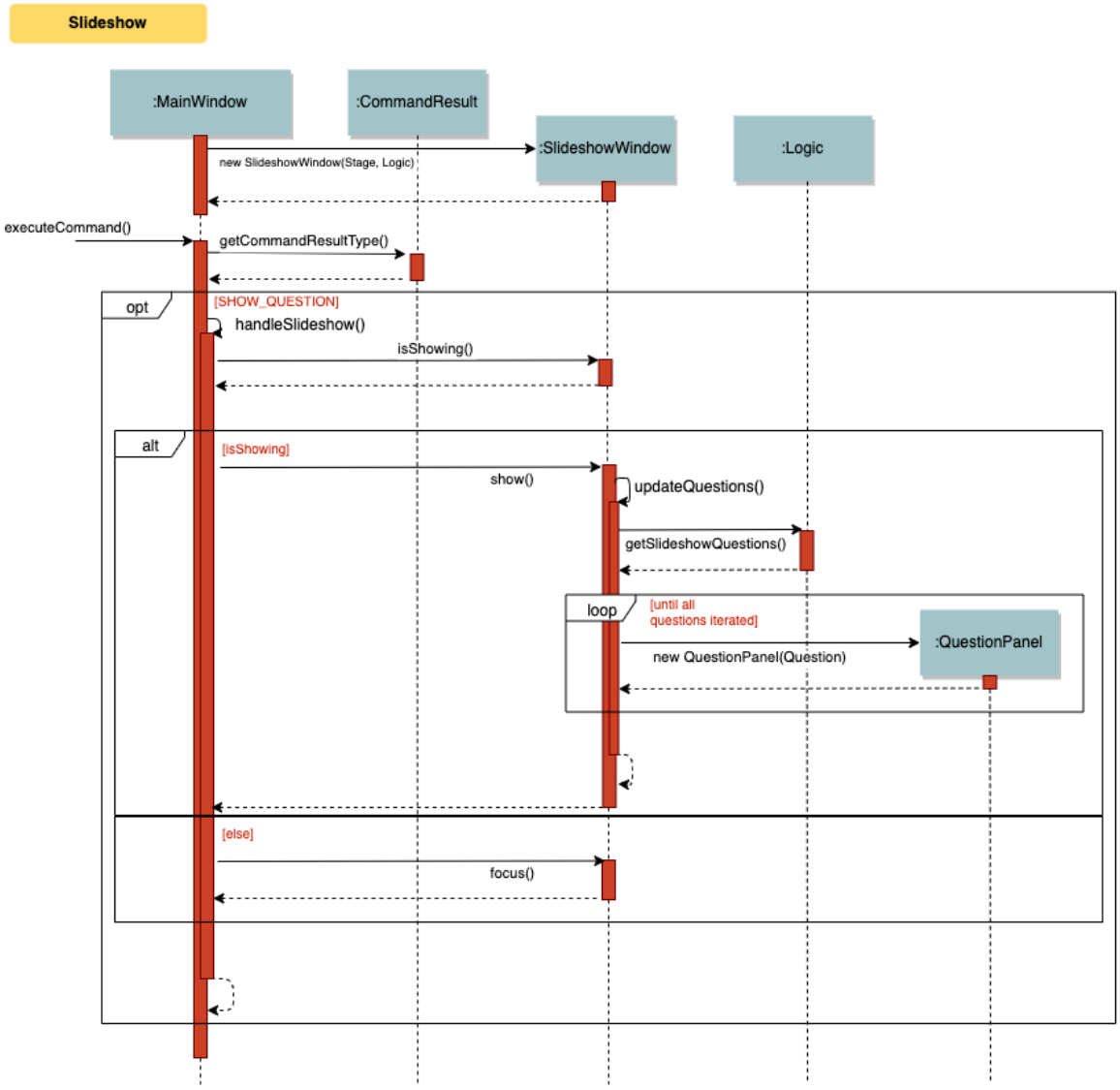

Figure 8. Sequence diagram illustrating the interactions happening.

The following is an example and detailed explanation as to how the questions are fetched and displayed on the slideshow.

Step 1. The user requests to start a slideshow with a selection of questions using the `question slideshow [QUESTIONS INDEX]` command. This will add questions based on the `Index` specified and will be added to the `slideshowQuestions` list under the `SavedQuestions` class. The usage of `Index` here is such that it follows the same convention of when the user edits or deletes a question.

NOTE	The order of the questions displayed on the slideshow will be **based on the input order**.

Step 2. The command is executed and the `MainWindow` calls `CommandResult#isShowSlideshow()` to verify if the command specified is to start a slideshow. The `SlideshowWindow` is then displayed through the `SlideshowWindow#slideShowWindow.show()`.

NOTE	The slideshow window has already been instantiated on application launch and the window is merely being hidden or shown.

Step 3. The window is now visible and existing questions are cleared. The list of slideshow questions is then fetched through `Logic#getSlideshowQuestions()` which in turn calls the `ModelManager#getSlideshowQuestions()` that fetches the `slideshowQuestions` list in `SavedQuestions`.

Step 4. The user will then navigate and control the slideshow using the `Left/Right`, `Space` and `Esc` key as defined by the key listeners in `SlideshowWindow#initialiseKeyboardControls(Stage root)`. The `currQuestionIndex` will be incremented when the user navigates to the next question and decremented when navigating to the previous question.

Step 5. The user exits the slideshow when the `Esc` key event is triggered or when the `currQuestionIndex` exceeds the `questionPanels.size()`. The behaviour of this follows the common procedure that most presentation programs adopt thus, it will not feel foreign to users.

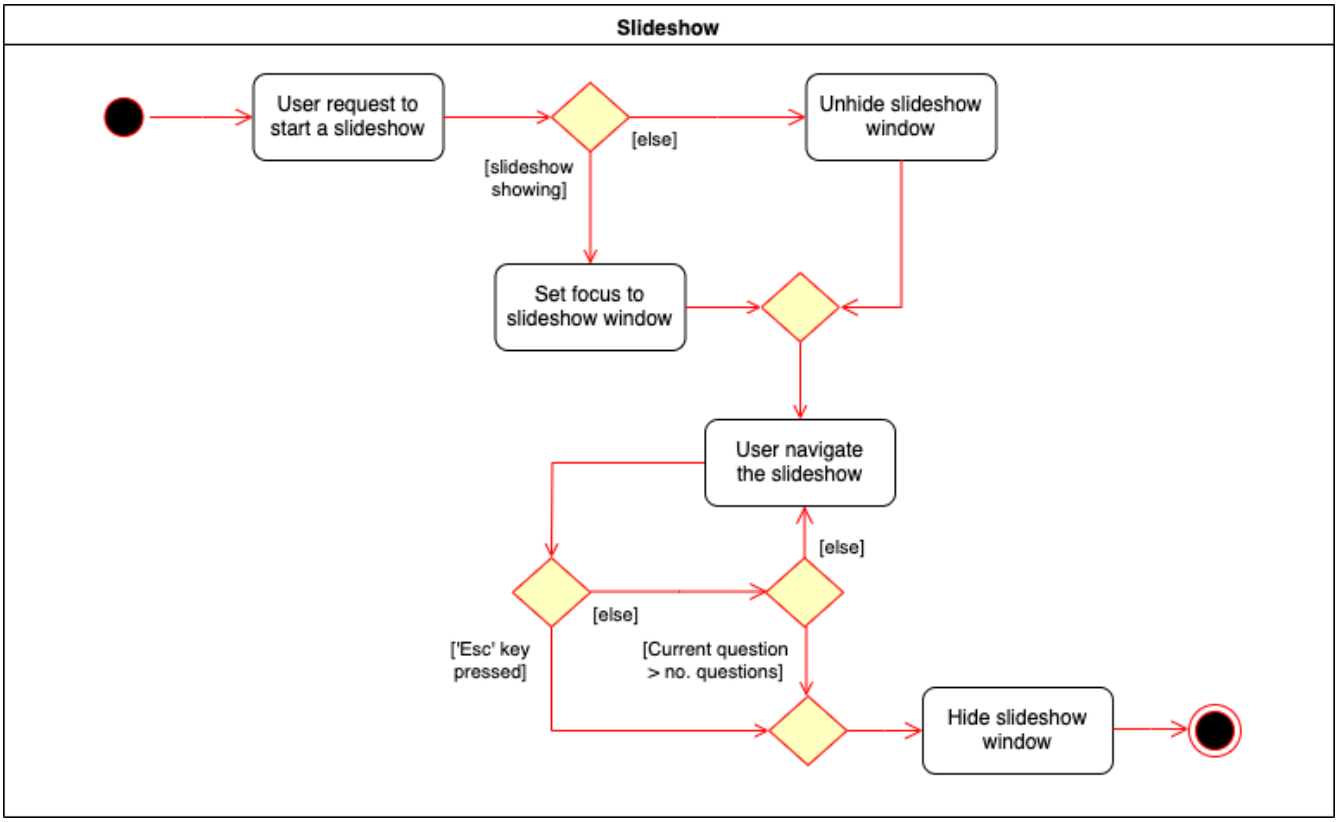

Figure 9. Activity diagram of the actions performed.

Design Considerations

Controls

- **Current Implementation:**
  - The choice of using the arrow keys for navigation and the `Escape` key to quit the slideshow is such that it will feel familiar to users who uses presentation programs often as they have similar controls. The only difference will be the usage of the `Space` key to show answers as it will be something new to the users and is unique to **Njoy**.
- **Alternatives Considered:**
  - Usage of the `A` key to show answers. However, this is not very feasible as it is easily forgotten and not as user-friendly due to the smaller surface of the key as compared to the `Space` key.

Display

- **Current Implementation:**
  - The ordering of the questions is defined based on the user input so it gives flexibility to the user to choose the ordering that they want the questions to be displayed.
  - Placement of the question numbering, topic, options and answer follows the common convention where the question numbering will be at the top followed by the topic, options and then the answer. This is such that viewers will not be confused by the layout.
  - Font sizes are displayed in the following descending order to allow the text for easy viewing:
    - Question Number
    - Question Topic & Options
    - Answer
- **Alternatives Considered:**
  - Having a separate answer format for MCQ such that it will have an arrow beside the correct option *e.g A) 1965* < **(Answer)**. However, this is not feasible as it will break the standard formatting of the answers display since both `open` and `mcq` type questions will have 2 different answer formats and may cause confusion to the user.