Fabian Chia - Project Portfolio

Overview

My team and I were tasked with enhancing AddressBook3 - a given CLI (Command Line Interface) application into a better product. Through the ideation phase, we decided to morph the application into the Njoy Teaching Assistant. We aim to improve the quality of life for teachers by assisting them with daily administrative tasks that have been traditionally resolved for years. In particular, the Njoy Teaching Assistant enables teachers to maintain student records; set questions and quizzes for revision; while keeping track of their schedules with an interactive timetable.

This is what our project looks like:

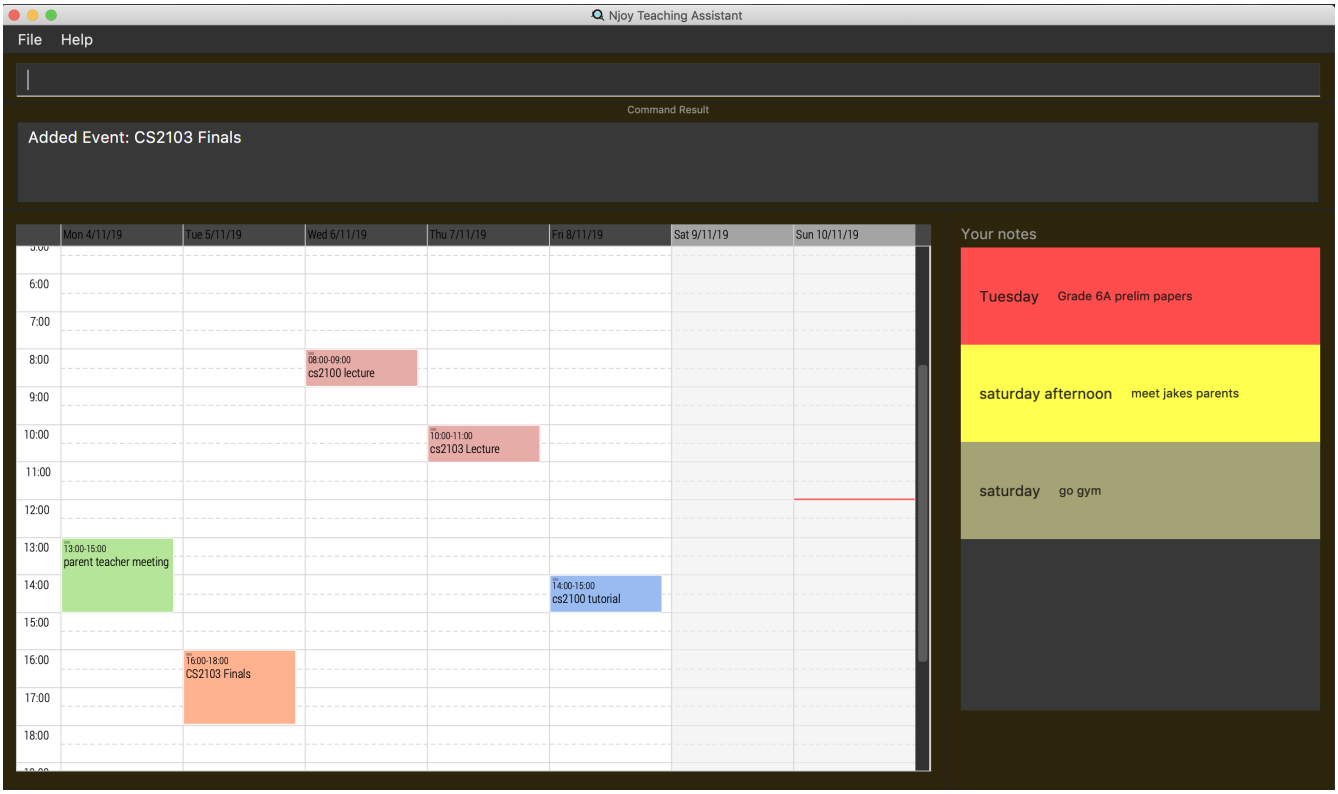

Figure 1. The graphical user interface for Njoy.

My role was to design and write code for the quiz features. The rest of the sections will cover the summary of my quiz specific contributions to the codebase, the user guide and the developer guide.

The following are icons and symbols that I will be using for the Project Portfolio:

NOTE	This symbol indicates a note to the user.
TIP	This symbol indicates a user friendly tip.

IMPORTANT	This symbol indicates something important.

`This indicates a component, class or object in the architecture of the application.`

This indicates important text.

Summary of contributions

This section entails a summary of my specific enhancements, code contributions and other helpful increments towards the Njoy Teaching Assistant.

I implemented the quiz specific commands which include:

- **Creating a quiz automatically**
- **Creating a quiz manually**
- **Adding a question to a quiz**
- **Deleting a question from a quiz**
- **Listing the questions and answers for a quiz**
- **Showing only the questions or answers for a quiz**
- **Exporting a quiz to a formatted HTML file**

I'll be sectioning the functionality into the following format:

1. **Creation**
2. **Editing**
3. **Display**

Enhancements

Quiz Creation

- **What it does:** Users can either create quizzes automatically or manually. Regardless of either option, users will also have to specify a quiz identifier for further storage and future operations. Automatic-wise, users have to input the number of questions they want to be added to the quiz and also, the type of questions to be added to the quiz. Users can choose to add mcq, open ended or both types of questions to their quiz. Manual-wise, users have to input the specific question indexes of questions in the question bank that they want to be added to the quiz.
- **Justification:** Teachers have an inherent need to prepare homework, revision papers and exam papers for their students. As such, the NJoy Assistant provides a way for them to continuously store questions in a database and generate quizzes based off that. They can either automate and randomise the process by specifying the number of questions, or even create tailored quizzes by specifying the question number indexes.
- **Highlights:** This enhancement works with existing as well as future commands. The implementation for this feature was relatively challenging because of the automation option

when creating quizzes. Users have to specify the number of questions they want to be included in a quiz and if the question bank does not contain enough questions then the command would not go through and would prompt the user with an error stating that there are insufficient questions in the question bank for the number that was specified. The hardest part was to make sure that no repeated questions were added to the quiz which involved keeping track of what questions have already been added.

Quiz Editing

- **What it does:** Users can choose to edit their quizzes after creation. This includes the addition of specific questions in the question bank, or even the deletion of specific questions from the quiz. All the user has to do is to specify the correct indexes of the questions to add or remove and the application will handle the rest. For the adding of questions, users can also choose to specify the quiz question index that the question should be added to. For example, if I had a quiz with question 1, 2 and 3, and I want to add a question to question 2 - I can specify just that. Question 1 would remain in the same index, question 2 would be the new question that was added, and the old question 2 and 3 would then be shifted down to the new question 3 and 4 indexes.
- **Justification:** Even after creating a quiz, teachers might still want to add new questions or even delete questions that they later realise were incorrect or were not supposed to be in the quiz. This is because though the NJoy Assistant already accounts for repeated questions or wrongful answers (an mcq answer not being present in the options), it does not ensure the correctness of questions and answers. For example, a question could be: What is 1 + 1? And the answer specified by the teacher could be 3. The Njoy Assistant would have no measures to prevent the question from being created and hence from being added to the quiz. As such, upon the detection of such errors, the teacher might then choose to delete the question from the quiz, then deleting it from the question bank itself.
- **Highlights:** This enhancement works with existing as well as future commands. The implementation for this feature was relatively challenging because of the adding questions feature. This is because we ensure that no repeated questions can be present in a quiz - that makes no sense. A quiz should not have a repeated question, because that would attribute to more marks given or lost depending on whether the student could have answered the question. Before adding a question to a quiz, we make sure that it does not match any existing question in the quiz. This ensures that the question will not be added to the quiz if it was indeed a repeat.

Quiz Display

- **What it does:** Users can choose to display their quizzes on the user interface after creation. There are several display options provided: showing both questions and answers, showing only the questions and showing only the answers. Beyond that, there is a defining feature whereby teachers can choose to display their quizzes in an exported HTML file. Of course, the answers will not be shown on the quiz because this HTML file is meant as a quiz for students. The questions will be formatted properly for display - mcq questions will have radio buttons for the 4 different options, and open ended questions will have blanks for the students to fill in. Just by specifying the quiz identifier, the user will be able to display their quizzes appropriately.
- **Justification:** The display functions serve as a confirmation of what the teacher currently has in their quizzes. Teachers can ensure that the correct questions and answers have been allocated to their specific quizzes before exporting them. As mentioned earlier, teachers are no stranger

to the preparation of homework, assignments and revision papers. After confirming the quiz is set up correctly, the key feature of the quiz component - the export feature creates a HTML file in an appropriate format for their students to fill in.

- **Highlights:** This enhancement works with existing as well as future commands. The implementation for this feature was relatively challenging because of the export quiz feature. In particular, the formatting of text for HTML was foreign to me which is why I spent a fair bit of time understanding how to write the correct syntax for HTML files.

Code contributed

Please click these links to see a sample of my code.

[Functional code] [Test code] *{Links to samples of my code.}*

Other contributions

- Project management:
  - Managed release v1.3.1 on GitHub, out of the 2 releases.
  - Resolved the issues found by others related to my feature on Github.
- Enhancements to existing features:
  - Updated README aka the main page, user guide and developer with NJoy Assistant related pictures: #177
  - Wrote additional tests for existing features to increase coverage from 38% to 44% (Pull requests #177, #184)
- Documentation:
  - Made cosmetic tweaks to the existing contents of the User Guide: #80, #103, #177, #179
  - Made relevant changes for the Developer Guide: #80, #103, #177
- Community:
  - Reviewed Pull Requests: #18, #20, #33, #79, #91, #106, #121, #175, #178, #182, #183, #188, #189
- Tools:
  - Set up an online continuous integration tool (`Travis`) for the team's repository on Github.
  - Integrated `coveralls.io` with the team's repository on Github for coverage updates.

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. The following are specific portions of the NJoy Assistant's User Guide that I have selected. This is because some features are repetitive, and as mentioned `above` I will be grouping them into Creation, Editing and Display once again. Therefore, I'll only show one example for each of the main features that I have written about in the user guide.

NOTE	The following is an example of **Quiz Editing**:

Quizzes - `quiz`

After taking the effort to record different questions, a teacher might be wondering what other features could the NJoy Teaching Assistant actually provide? What better way to make use of your predefined questions to create quizzes for homework, revision or even assignments?

NOTE	Unfortunately, since majority of the quiz commands rely on question and quiz question indexes, it is entirely up to the user to ensure the correct indexes of the questions or quiz questions have been entered.

You may refer to creating a question for examples on how to add questions and listing the questions on how to view the current list of questions in the databank.

NOTE	For the following examples, we will be operating under the assumption that these sample questions are in the question bank.

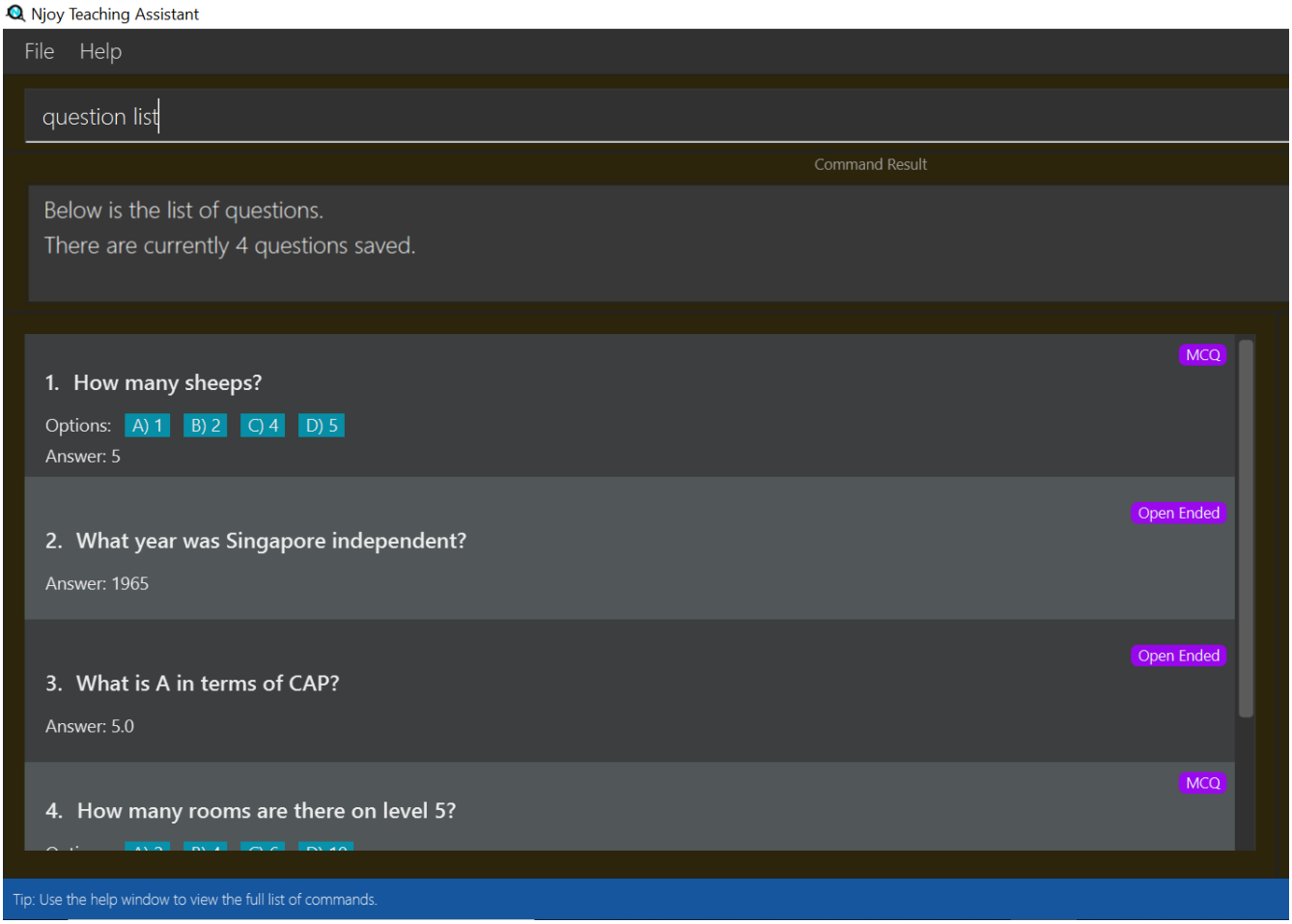

Figure 1. Question list

Creating a Quiz manually: `manual`

> **NOTE** The following is an example of **Quiz Display**:

The questions chosen are randomised from the questions that you have previously added. As such, the user has to ensure enough questions are available in storage for quiz creation.

Format: `quiz auto quizID/… numQuestions/… type/… (Type can either be mcq/open/all)`

The format supported by this feature includes:

Keyword	Description
Quiz ID	The label of the quiz.
Number of Questions	The number of questions you want added to the quiz.
Question Type	The type of questions you want added to the quiz: mcq, open, all.

Examples:

- `quiz auto quizID/CS2103T numQuestions/2 type/mcq`
  Adds 2 questions of type mcq to the quiz labelled CS2103T.
- `quiz auto quizID/CS2103T numQuestions/1 type/open`
  Adds 1 question of type open-ended to the quiz labelled CS2103T.
- `quiz auto quizID/CS2103T numQuestions/3 type/all`
  Adds 3 questions of any type to the quiz labelled CS2103T.

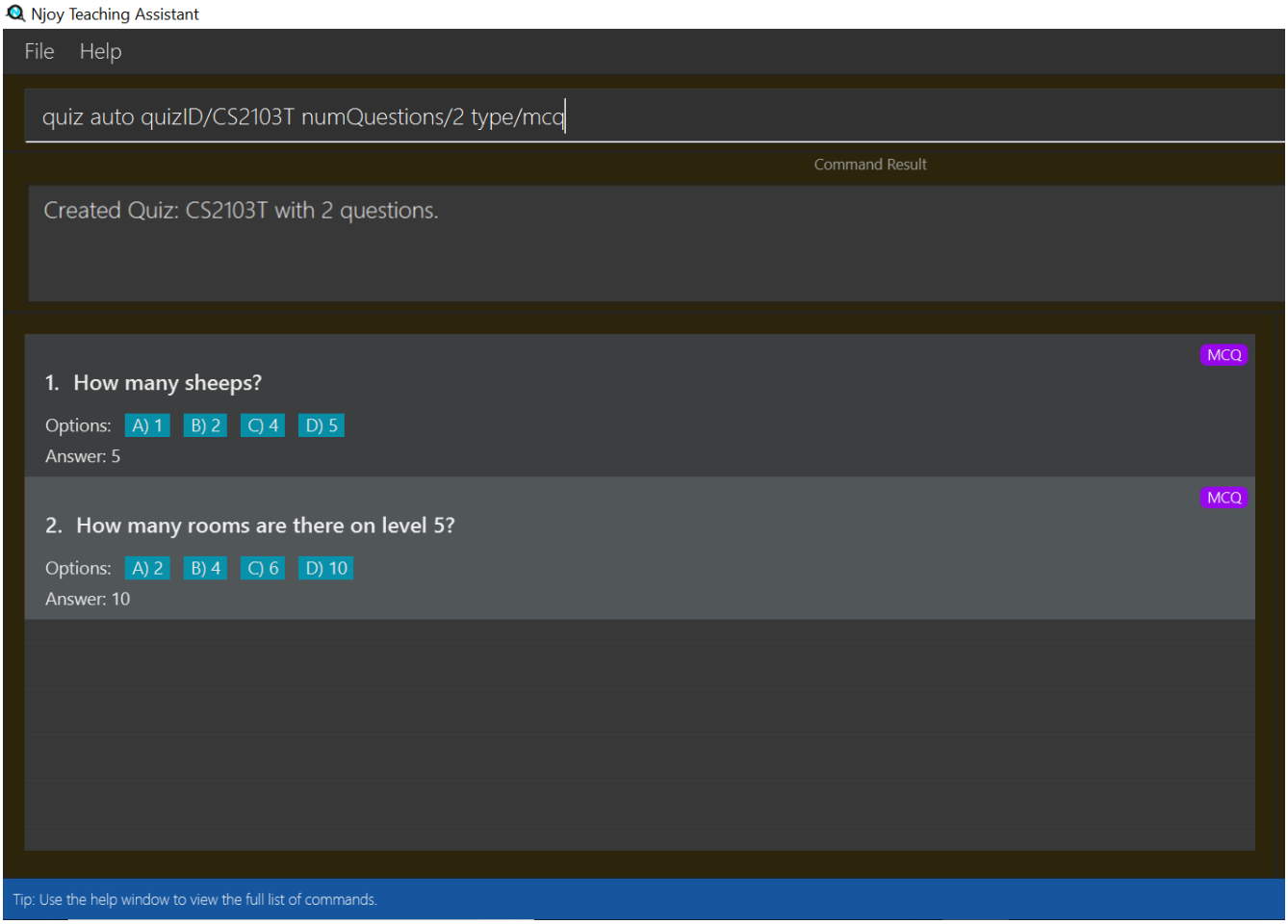

Figure 2. Results of the first create quiz automatically command.

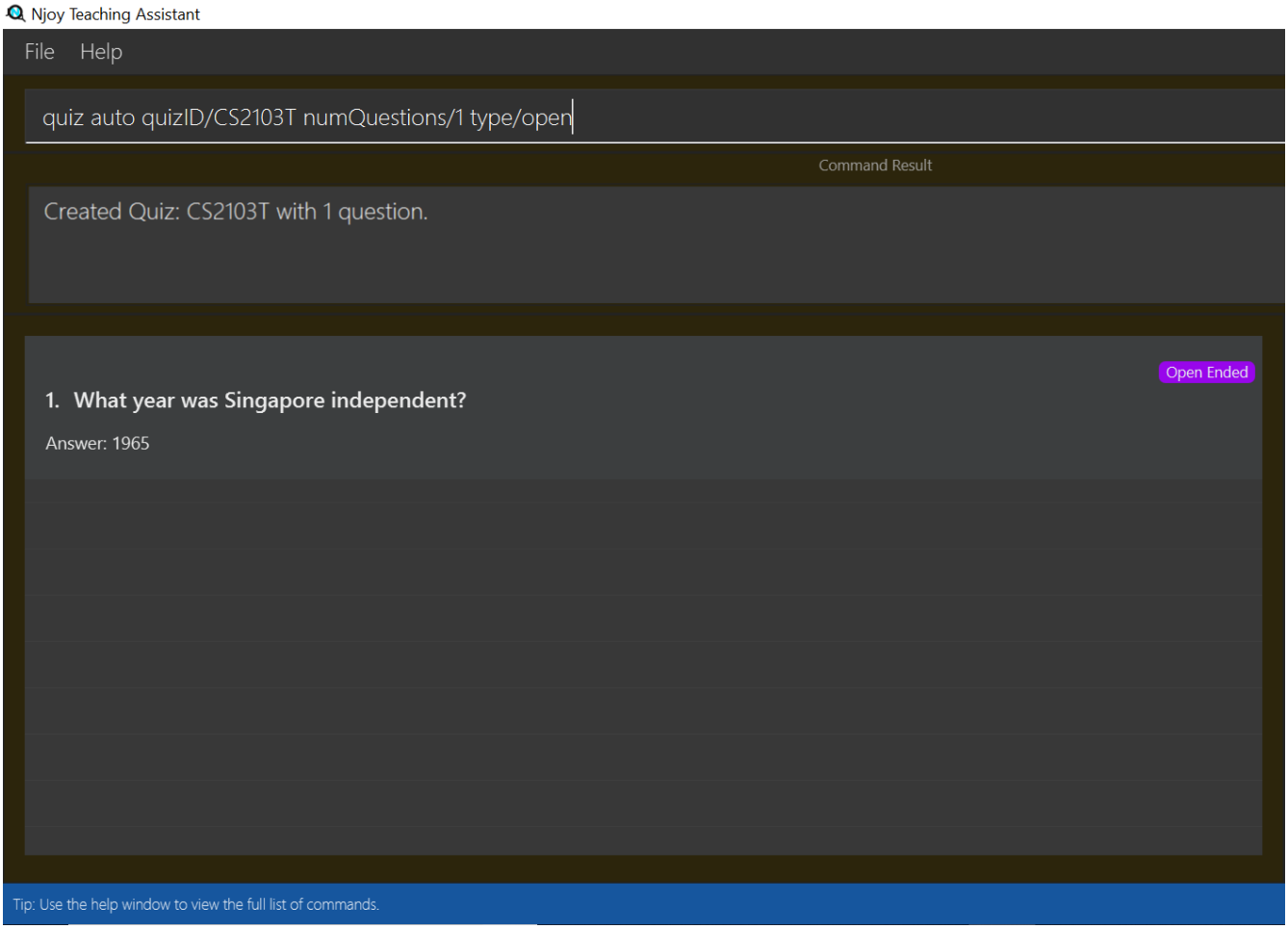

Figure 3. Results of the second create quiz automatically command.

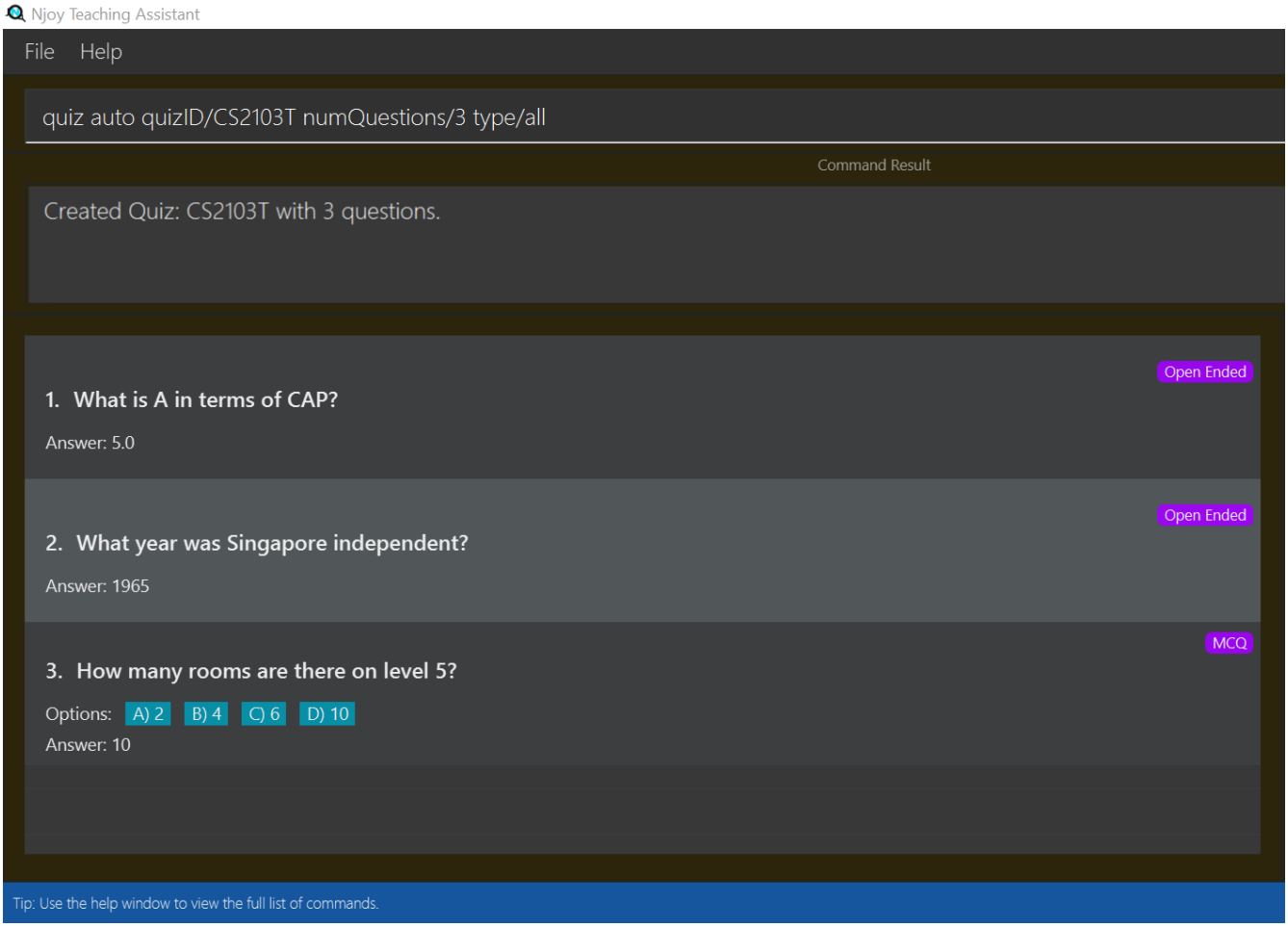

Figure 4. Results of the third create quiz automatically command.

Adding a Question to a Quiz: `add`

Allows a user to add a question to a quiz.

Format: `quiz add quizID/… questionNumber/… quizQuestionNumber/…`

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. Again, I'm only going to include the most relevant portions of the guide as mentioned in the `above` citing. Also, some of the command hyperlinks will not work because I have omitted them for brevity.

Quiz feature

Overview

The quiz feature utilises the questions implemented and stored in the `QuestionBank#questions` observable list. The quiz feature utilises the `QuizCommandParser` class to parse the user command input into the different command types and validates the input. Quizzes are then added into the `QuizBank#quizzes` observable list. The quiz feature also relies heavily on the `QuizManager` class for

handling commands from `QuizCommand#execute`. This is done to hide the implementation logic from the `ModelManager` class.

NOTE The following is an example of **Quiz Creation**:

When the user inputs the `quiz manual` command in the command line, the following chain of operations occur:

1. The `NjoyParser` will delegate the parsing of the command to `QuizCommandParser`.
2. `QuizCommandParser#parse()` will take in a `String` input consisting of the arguments.
3. The arguments will be tokenized and the respective models for each argument are created.
4. If the parsing of all arguments are successful, a new `QuizCreateManuallyCommand` is returned back to `LogicManager`.
5. The `LogicManager` executes `QuizCreateManuallyCommand#execute()`. This in turn executes `model#createQuizManually()`.
6. The `ModelManager` defers the operations to `SavedQuizzes#createQuizManually()`.
7. Finally, this delegates the actual operations to `QuizManager#createQuizManually()`.
8. The newly created `Quiz` object is added to the `QuizBank` in `SavedQuizzes` for storage and further use.

Creating Quiz Automatically

The create quiz automatically feature allows the user to create a quiz in Njoy. This feature is facilitated by `CreateQuizAutomaticallyCommand`, `QuizCommandParser`, `NjoyParser`, `SavedQuizzes`, `QuizBank` and `QuizManager`. The arguments supported by this feature includes:

- `Quiz ID`
- `Number of Questions` (1…*)
- `Question Type` (Mcq, Open ended, All)

Example of a possible command: `quiz auto quizID/CS2103T numQuestions/2 type/mcq` This randomly adds 2 mcq questions to the quiz named CS2103T.

Implementation

NOTE The following is an example of **Quiz Display**:

When the user inputs the `quiz delete` command in the command line, the following chain of operations occur:

1. The `NjoyParser` will delegate the parsing of the command to `QuizCommandParser`.

2. `QuizCommandParser#parse()` will take in a `String` input consisting of the arguments.
3. The arguments will be tokenized and the respective models for each argument are created.
4. If the parsing of all arguments are successful, a new `QuizDeleteQuestionCommand` is returned back to `LogicManager`.
5. The `LogicManager` executes `QuizDeleteQuestionCommand#execute()`. This in turn executes `model#deleteQuizQuestion()`.
6. The `ModelManager` defers the operations to `SavedQuizzes#deleteQuizQuestion()`.
7. Finally, this delegates the actual operations to `QuizManager#deleteQuizQuestion()`.
8. The `Question` object is deleted from the chosen `Quiz` object in the `QuizBank` in `SavedQuizzes` for storage and further use.

Exporting a Quiz to HTML

The quiz export feature allows the user to export a quiz to HTML in Njoy. This feature is facilitated by `QuizExportCommand`, `QuizCommandParser`, `NjoyParser`, `SavedQuizzes`, `QuizBank` and `QuizManager`. The arguments supported by this feature includes:

- `Quiz ID`

Example of a possible command: `quiz export quizID/CS2103T` This exports the quiz named CS2103T to a HTML file in the user's directory.

Implementation

When the user inputs the `quiz export` command in the command line, the following chain of operations occur:

1. The `NjoyParser` will delegate the parsing of the command to `QuizCommandParser`.
2. `QuizCommandParser#parse()` will take in a `String` input consisting of the arguments.