CoderAcademy Ruby Assessment

QUIKQUIZ - CLI Quiz Application

QuikQuiz is a CLI quiz application written in Ruby for my first assessment at CoderAcademy.

The application functions as a quick and simple quiz generator, with the capability to both quiz students with randomly generated questions, and display test results to their teachers. The application stores previously written questions within a JSON file, from which questions are retrieved at random and generated into a set of questions when using the 'student' portal. Once the quiz is completed, the student name, score and a timestamp are stored within another JSON file, which can be retrieved via the 'teacher' portal and displayed in a human-readable format. The application also has the ability to generate new questions, via a terminal command.

Many quiz applications available, for example Kahoot!, require teachers to write new questions each session, limiting the range of questions available and the scope of assessment. QuikQuiz allows teachers to write any number of questions within a file, from which questions will be randomly selected and presented to students. This will allow educators to broaden the scope of their assessment, and with repeated use of the application, ensure that students have a strong grasp of a broader range of learning outcomes.

QuikQuiz is targeted towards students and educators at any stage of education due to its ease of use, simple interface and completely modular question generation. Teachers of subjects requiring repeated excercises, such as mathematics and programming, could find a simple and intuitive teaching tool within the application, as it allows many small problems to be presented and resolved by students. The readability of results allows teachers to see, in a simple format, how well each of their students is faring with the current material.

The application is controlled via text inputs in the terminal, with an initial option of whether to access the student or teacher portal. Students are required to enter their name for record-keeping purposes, after which they will be presented with a previously generated set of questions. Once completed, students will receive their grade and the option to take the quiz again or exit. Teachers are required to enter a password for security purposes, after which they will be presented with a readable list of all previously completed tests, featuring student names, scores and timestamps. Teachers will have the option to exit the application once they are finished checking the data.

Development Plan

Features

Generation of Question Set

- Parse information from questions.JSON *(class.rb, line 35)*
- Converts JSON data to an array within the student portal, from which 10 questions and answers are selected using a loop and stored for display during the students quiz. The generation removes each hash as it is selected from the array, to ensure questions are not selected multiple times. *(index.rb, line 48)*

Storage of student results and details

- Parse information from results.JSON *(class.rb, line 18)*
- Converts JSON data to an array within the **test**, which then has the students **test** results *pushed* to the array. This is then converted back to JSON and stored within results.JSON for retrieval. *(index.rb, line 138)*

Display Student Results

- Parse information from results.JSON
- Converts the JSON to a variable, which is then looped through, printing each students name, score & date with well-formatted strings. *(index.rb, line 184)*

Error Handling - Backup Data Storage & Empty-Input checks

- If corrupted/empty, JSON files will be re-initialized for use. *(class.rb, lines 20 & 37)*
- All input fields outside of the quiz section require a text input - If none is found, the application will loop back and request again.

User Interaction Outline

1. Landing page - user must state whether they are a teacher or student.
    - If a student, the user will be asked to enter their name.
    - If a teacher, the user will be asked to enter the required password.
        - If the input is incorrect/blank for either of these requests, the user will be taken to the previous prompt to try again.
2. Within the teacher portal, a complete history of student results stored within results.JSON will be presented in a readable format. If the results file is corrupted, a notice will inform the teacher that it has been re-created for future use. The teacher will then have the option to exit the application.
3. Within the student portal, the quiz will begin automatically once a name has been submitted. Each question will display a question number, question and the three available responses. The user will input their response (a, b or c) and be taken to the next question.
4. Once a student has completed all of the questions, they will receive a prompt informing them of their grade, and the option to re-try the quiz or to exit the application.

Control Flow Diagram

Control Flow

Develop an implementation plan which:

- outlines how each feature will be implemented and a checklist of tasks for each feature
- prioritise the implementation of different features, or checklist items within a feature
- provide a deadline, duration or other time indicator for each feature or checklist/checklist-item