RMIT UNIVERSITY

School of Computing Technologies

Final Project - COSC2675 Rapid Application Development

	Assessment Type: **Individual assignment** Submit online via Canvas Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums.
	Due date: 24 hours after release; Please check Canvas → Assignments → Final Project for the most up to date information. As this is a major assessment replacing the exam. Late submission will NOT be accepted unless there is a granted special cosideration request or ELP.
	Weighting: 40 marks /100

1. Overview

The objective of this project is to assess high level web development skill including task analysis, fast development and deployment.

If there are questions, you must ask via online forums or similar communication channels. You should phrase your question in a general manner. The teaching team is not supposed to debug for you. Do not post your source code or your proposed solutions.

2. Assessment details

You are asked to implement a quiz website, called "**Quiz+**", which is based on an existing quiz API, **`quizapi.io`**. Quiz API is a simple REST API for technical quizzes covering Linux, DevOps, Networking, Cloud and so on. You need read the documentation on **`quizapi.io`** to understand how to get authentication, how to get quizzes/answers, what are the parameters etc etc.

Note, when you sign up with Quiz API, you may get a HTTP Error 500. You can ignore that and log in with your signup email and password.

Under a Unix-like terminal, e.g. Terminal on Mac, you can try the following:

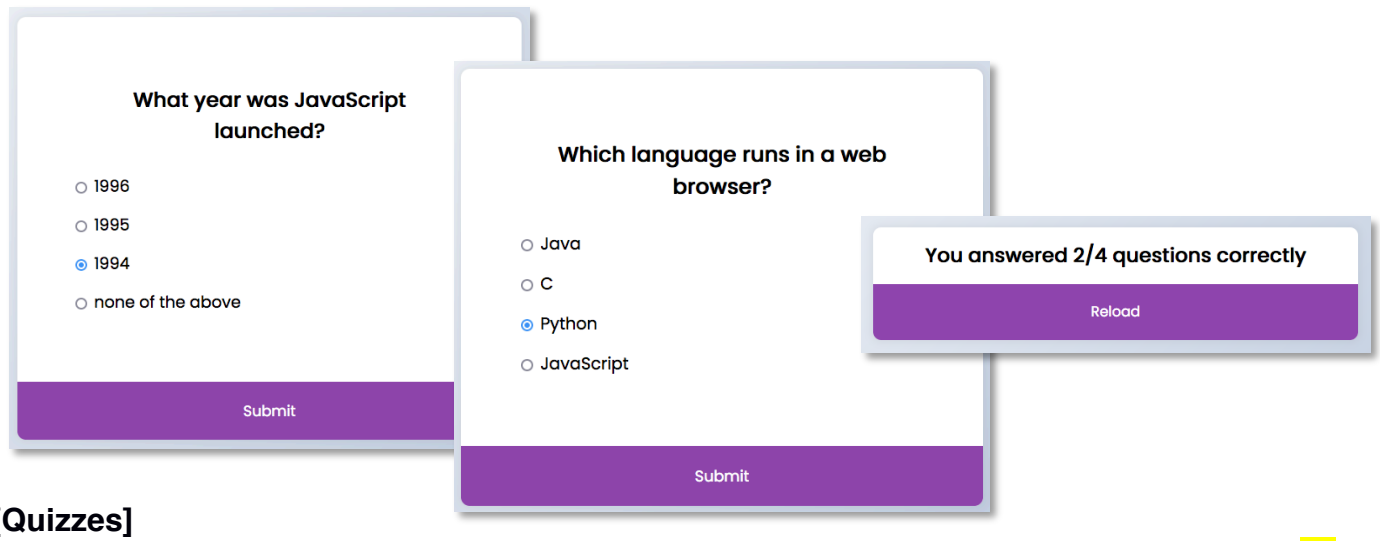
```
curl https://quizapi.io/api/v1/questions -G \
-d apiKey=YOUR_API_KEY \


curl https://quizapi.io/api/v1/questions -G \
-d apiKey=YOUR_API_KEY \
-d limit=10 \
-d category=Linux
-d difficulty=easy
```

Below is an example of api response in JSON format.


```
[
  {
    "id": 1,
    "question": "How to delete a directory in Linux?",
    "description": "delete folder",
    "answers": {
      "answer_a": "ls",
      "answer_b": "delete",
      "answer_c": "remove",
      "answer_d": "rmdir",
      "answer_e": null,
      "answer_f": null
    },
    "multiple_correct_answers": "false",
    "correct_answers": {
      "answer_a_correct": "false",
      "answer_b_correct": "false",
      "answer_c_correct": "false",
      "answer_d_correct": "true",
      "answer_e_correct": "false",
      "answer_f_correct": "false"
    },
    "explanation": "rmdir deletes an empty directory",
    "tip": null,
    "tags": [],
    "category": "linux",
    "difficulty": "Easy"
  }
]
```


========== PASS LEVEL ==========



[Quizzes]

At this level, your Rails application should have a question database containing no less than 10 questions from QuizAPI. When a user visits your application, he/she should see at least 4 different questions one by one which are picked from the question bank.

After the visitor answered all the questions, the result should show, for example, out of 4 questions, you got 2 correct. At this point, your app should allow the user to re-attempt the quizzes (the Reload button in the screenshot above). There is no limits on the number of re-try.

At this level, you can assume all questions are “Easy” and have only one correct answer.

Note: The quiz data should be loaded from your appliaction’s top directory, e.g. where Gemfile is, from a file called “**`quiz.json`**”. During marking, we may use a different file (same file name but different data from QuizAPI) to test your app.

Each question may have different number of options, e.g. 3 options, 4 options, 5 options …

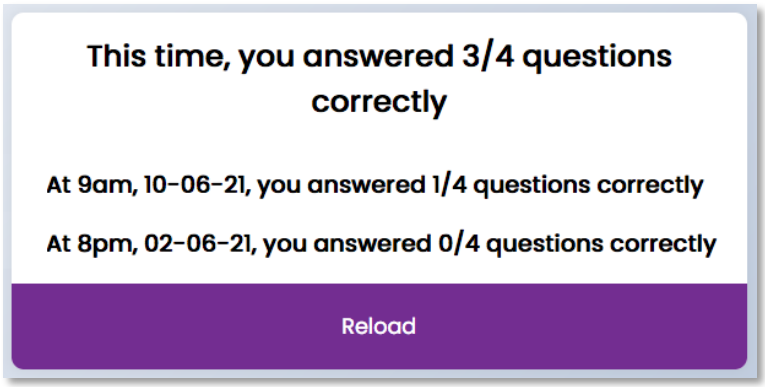
[Header]
Your application should have a header. It shows **your name** and **your student email address**.

[Footer]
Your application has a footer, showing **the highest level you achieved**, links to Facebook, Twitter and LinkedIn. You can link to actual accounts or simply the home page of these social media sites.

========== Credit LEVEL ==========

NOTE: You should only attempt this level if you have completed PASS level.

[History]
At this level, your Rails application ‘remembers’ the early attempts of the same user (from the same device). The previous results will be shown with the current result. See the screenshot below. Also time and date of each early attempt should appear as well. No more than 5 records will show.



[Variable Quizzes]
The questions appear at each attempt should be different, e.g. randomly picked from the quiz database. No question should appear twice in the same attempt.

Also the total number of questions of each attempt can be adjusted at the beginning of the attempt, through a selection ranged from **4 - 8**.

At this level, you can also assume all questions are “Easy” and have only one correct answer.

[Missing answers]
All question must be answered. Your app would not allow the user to continue if the current question is not answered.

RMIT UNIVERSITY

========== Distinction LEVEL ==========

NOTE: You should only attempt this level if you have completed CREDIT level.

[Dynamic Quizzes]
From this level, your app is now capable of directly working with the API, meaning the questions are dynamically loaded from the API. If Quiz API is offline or authentication is unsuccessful, then your app will use "**`quiz.json`**" as specified in the PA/CR levels.

[Categories]
At the beginning of each quiz, your app would allow users to adjust the number of questions (same as that in CR) and pick categories. At least four categories should be supported, e.g Linux, DevOps, Networking and Programming. Visitors can pick any combination of categories, from one to all. The questions appeared in the subsequent quiz would be draw from the selected category/categories.

[Remember Settings]
At the end of each quiz, user can reload the quiz as that in PA/CR. However all the settings of the previous quiz are remembered, e.g. the number of questions and the categories. The user can keep the same setting for the new attempt or adjust them based on the previous settings.

At this level, you can pick all questions from the "Easy" level. All PA/CR features should be supported, e.g. no duplicate questions in each attempt, all questions are randomly picked etc.

========== 80+ LEVEL ==========

NOTE: You should only attempt this level if you have completed DISTINCTION level.

[Difficulty Level]
On top of adjusting number of questions (CR), categories (DI), your app at this level allows users to pick difficulty level at the beginning. Difficulty levels are mutually exclusive, either Easy, or Medium or Hard, meaning the questions in one quiz will all be at the same difficulty level.

[Multiple Answers]
"Hard" questions may have multiple correct answers. Your app at this level should be able to handle that and allow user to pick multiple answers. The answer is considered wrong if the answer from the user is different to the standard answer. There is no partially correct in this app.

Note, you should NOT rely on the field `"multiple_correct_answers": "false"` (See Line 14 on Page 2) as some times Quiz API gives incorrect value for that tag. Instead you need to check whether there are indeed multiple correct answers.

[Redo]
At the end of each quiz, other than reloading a new quiz, the user has one more option – repeat the same quiz. All questions in the new attempts will be identical to the previous attempt, appearing in the same order.

At this level all PA/CR/DI features should be supported, e.g. no duplicate questions in each attempt, all questions are randomly picked, remembering settings etc.

RMIT UNIVERSITY

========== 90+ LEVEL ==========

NOTE: You should only attempt this level if you have completed 80+ level.

[Full History]
You app now supports viewing of the full quiz history via this URL **<Your app domain>/history**

It shows ALL attempts regardless of user or device, sorted by time, descending from the current time. The IDs of questions of each quiz are also shown.

```
Date/Time: 9am 10-06-21
Topic: Linux, DevOps
Difficulty: Hard
Questions: 2, 3, 45, 122, 345, 23, 67, 1201
Result: 4 out of 8

Date/Time: 8pm 02-06-21
Topic: Linux
Difficulty: Easy
Questions: 64, 13, 15, 122, 223
Result: 3 out of 5
```

[Previous and Next]
As an advanced feature, your app allows user to go backwards and forwards before submission, similar to the quizzes on Canvas. The first question only has a "**Next**" button and the last question has a "**Previous**" and "**Submission**" button. The rest of questions all have "**Previous**" and "**Next**" buttons at the bottom. User can change answer before hitting the submission button at the end. Once the submission is click, the result will show as specified in the CR level on Page 3.

Warning, 90+ level features are challenging. You should attempt them only after making sure features at lower levels are all correctly implemented and can run successfully locally and on Heroku.

5. Referencing guidelines

If you have used sources of information other than the contents directly under Canvas→Modules, you must give acknowledge the sources and give references. Add a code comment near the work to be referenced and include the reference.

6. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of others without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to the University website.

7. Assessment declaration

When you submit work electronically, you agree to the assessment declaration.

8. Submission

8.1 Submission on Canvas -> Assignments -> Final Project -> 

* Upload the zipped source file onto Canvas (the first tab from left). **Click Submit**
* Copy n paste README.md in the Text Entry (the second tab). **Click Resubmit.**
* Copy n paste your Heroku URL into the third tab (Website URL). **Click Resubmit.**

8.2 The submission must be **individual**.

8.3 You are free to use any GEM.

8.4 Your application must be product ready and deployed on **Heroku**.

8.5 There is no need for your application to look exactly like the above example. You are recommended to use bootstrap or something similar for your app.

8.6 In the **README.md** file of your application, you should state

- Your quizapi `apiKey`
- Heroku deployment URL and the last Heroku deployment log

8.7 Include your Heroku URL and your quizapi `apiKey` in the comments on Canvas.

Submission failed to run on Heroku and locally would not be marked.

Rubric

Assessment Task	**Marks**
Pass Level	50 – 59%
Credit Level	60 – 69%
Distinction Level	70 – 79%
80+ Level	80 – 89%
90+ Level	90 – 100%