College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
ID608001: Intermediate Application Development Concepts
Level 6, Credits 15

Assessment 1: Node.js RESTful API - Open Trivia DB

Assessment Overview

In this **individual** assessment, you will develop a **RESTful API** using **Node.js** & deploy it to **AWS**. The main purpose of this assessment is to demonstrate your ability to develop a **RESTful API** using the various taught concepts. However, you will be required to independently research & implement more complex concepts. In addition, marks will be allocated for code elegance, documentation & **Git** usage.

Due to a nation-wide lockdown, your local pub is no longer able to run their weekly quiz night onsite. Your local pub owners know you are an IT student & ask if you want create an online quiz night application for them. For the first step, the pub owners want a **RESTful API** that provides various functions for registering, logging in, participating in various quizzes & keeping track of scores so that they can give away prizes at the end of each quiz night.

Learning Outcome

At the successful completion of this course, learners will be able to:

1. Apply design patterns & programming principles using software development best practices.
2. Design & implement full-stack applications using industry relevant programming languages.

Assessments

Assessment	Weighting	Due Date	Learning Outcomes
Practical: Skills-Based	20%	27-10-2022 (Wed at 7.59 AM)	1
Assessment 1: Node.js RESTful API - Open Trivia DB	45%	17-10-2022 (Mon at 7.59 AM)	1 & 2
Assessment 2: React - Open Trivia DB	35%	09-11-2022 (Wed at 2.59 PM)	1 & 2

Conditions of Assessment

You will complete this assessment during your learner-managed time. However, there will be time during class to discuss the requirements & your progress on this assessment. This assessment will need to be completed by **Monday, 17 October 2022** at **7.59 AM**.

Pass Criteria

This assessment is criterion-referenced (CRA) with a cumulative pass mark of **50%** over all assessments in **ID608001: Intermediate Application Development Concepts**.

Authenticity

All parts of your submitted assessment **must** be completely your work. If you use code snippets from **GitHub**, **StackOverflow** or other online resources, you **must** reference it appropriately using **APA 7th edition**. Provide your references in the **README.md** file in your repository. Failure to do this will result in a mark of **zero** for this assessment.

Policy on Submissions, Extensions, Resubmissions & Resits

The school's process concerning submissions, extensions, resubmissions & resits complies with **Otago Polytechnic** policies. Learners can view policies on the **Otago Polytechnic** website located at https://www.op.ac.nz/about-us/governance-and-management/policies.

Submission

You **must** submit all project files via **GitHub Classroom**. Here is the URL to the repository you will use for your submission – https://classroom.github.com/a/i4G4NwNS. Create a **.gitignore** & add the ignored files in this resource - https://raw.githubusercontent.com/github/gitignore/main/Node.gitignore. The latest project files in the **master** or **main** branch will be used to mark against the **Functionality** criterion. Please test before you submit. Partial marks **will not** be given for incomplete functionality. Late submissions will incur a **10% penalty per day**, rolling over at **8:00 AM**.

Extensions

Familiarise yourself with the assessment due date. If you need an extension, contact the course lecturer before the due date. If you require more than a week's extension, a medical certificate or support letter from your manager may be needed.

Resubmissions

Learners may be requested to resubmit an assessment following a rework of part/s of the original assessment. Resubmissions are to be completed within a negotiable short time frame & usually **must** be completed within the timing of the course to which the assessment relates. Resubmissions will be available to learners who have made a genuine attempt at the first assessment opportunity & achieved a **D grade (40-49%)**. The maximum grade awarded for resubmission will be **C-**.

Resits

Resits & reassessments **are not** applicable in **ID608001: Intermediate Application Development Concepts**.

Instructions

You will need to submit an application & documentation that meet the following requirements:

Functionality - Learning Outcomes 1, 2, 3 (60%)

- **User:**
  - You will have **three** types of users - **super admin**, **admin** and **basic** user.
  - Each user will have the following information: first name, last name, username, email address, profile picture, password, confirm password and role. The users' profile picture will be fetched from the following **API** - https://avatars.dicebear.com/docs/http-api using Axios.
  - Each user can login, logout, get their information and update their information. A **super admin** user can get all users' information, update all **admin** and **basic** users' information and delete all **admin** and **basic** users. An **admin** user can get all **admin** and **basic** users' information and update all **basic** users' information. A **basic** user can register.
  - When performing a **POST** request for registering a **basic** user, the following error checking must be implemented:
    * First name has a minimum length of two characters, a maximum length of 50 characters and alpha characters only.
    * Last name has the same error checking as first name above.
    * Username is unique, has a minimum length of five characters, maximum length of ten characters and alphanumeric characters only, i.e., johndoe123.
    * Email address is unique, contains the username above, an @ special character and a second-level domain, i.e., johndoe123@email.com.
    * Password has a minimum length of eight characters, maximum length of 16 characters and contains one numeric character and one special charcater.
    * Confirm password is the same as the password above.

    For each error check, a status code and response message is returned, i.e., "First name must have a minimum length of two characters".
  - When performing a **POST** request for logging in a user using either username/password or email address/password, return a status code, a response message, i.e., "<User's username> has successfully logged in" and the user's **JWT**.
  - When performing a **GET** request for logging out a user, return a status code, a response message, i.e., "<User's username> has successfully logged out" and set the user's **JWT** to expired.
  - When performing a **PUT** and **DELETE** request, return a status code and a response message, i.e., "<User's username>'s information has successfully updated" or "<User's username> has successfully deleted".
  - Two **super admin** users are seeded via you. Only you can seed the two **super admin** users. The **admin** users' data will be fetched from a local file and inserted into the **User** table using **Prisma**.
  - Five **admin** users are seeded via a **super admin** user. Only a **super admin** user can seed the five **admin** users. The **admin** users' data will be fetched from a private **GitHub Gist** using **Axios** and inserted into the **User** table using **Prisma**.
  - Five **basic** users are seeded via a **super admin** or an **admin** user. Only a **super admin** or an **admin** user can seed the five **basic** users. The **basic** users' data will be fetched from a private **GitHub Gist** using **Axios** and inserted into the **User** table using **Prisma**.
- Quiz:
  - Each quiz will have the following information: name, start date, end date, category, difficulty, type, number of questions, list of questions, list of correct answers, list of incorrect answers, list of scores, average score, list of ratings, average rating and overall winner. The category, list of questions, list of correct answers and list of incorrect answers will be fetched from the following **API** -

https://opentdb.com/api_config.php. The difficulties will be easy, medium and hard. The types will be multiple choice or true/false.

  - Each user can get all quizzes, get all past quizzes, get all present quizzes, get all future quizzes, get a list of scores and get a list of ratings. A **super admin** and an **admin** user can create a quiz. A **super admin** user can delete a quiz. A **basic** user can participate in a quiz and rate a quiz.
  - When performing a **POST** request for creating a quiz, the following error checking must be implemented:
    * Name has a minimum length of five characters, a maximum length of 30 characters and alpha characters only.
    * Start date has to greater than today's date.
    * End date has to greater than the start date and no longer than five days.
    * Number of questions has to be ten.

    For each error check, a status code and response message is returned, i.e., "Name must have a minimum length of five characters".
  - When performing a **POST** request for a **basic** user who is participating in a quiz, the following error checking must be implemented:
    * Can not participate if today's date is before the start date and after the end date.
    * Answered all ten questions.
  - When performing a **POST** request for a **basic** user who has participated in a quiz, return a status code, a response message, i.e., "<User's username> has successfully participated in <Quiz's name>", user's score and quiz's average score.

- HTTP:
  - When performing a **GET** request for **/api/v1/**, return a response containing all available endpoints in the **RESTful API**.
  - Headers are secured using **Helmet**.
  - Implement **CORS**, **compression**, **caching** and **rate limiting**.
- API Testing:
  - API tests are written using **Mocha** and **Chai**.
  - At least 40 **API**/integration tests verifying the user and quiz functionality.
  - Code coverage using **c8**.
- Deployment:
  - **RESTful API** is deployment to **AWS**.

Code Elegance - Learning Outcome 1 (25%)

- Environment variables' key is stored in the **example.env** file.
- **PostgreSQL** databases configured for development and production environments.
- Variables, functions and resource groups are named appropriately.
- Idiomatic use of control flow, data structures and in-built functions.
- File header comment for each controller and route file explaining its purpose using **JSDoc**.
- Code is linted and formatted using **ESLint** and **Prettier**.
- **Pre-commit hook** for **ESLint**, **Prettier** and **Git** commit message conventions using **Husky**.
- **Mocha**, **Chai**, **ESlint**, **Prettier** and **Husky** are install as **development dependencies**.

- NPM scripts for the following:
  - Opening **Prisma Studio**.
  - Creating a migration using **Prisma**.
  - Linting and fixing your code using **ESLint**.
  - Formatting your code using **Prettier**.
  - Running **API**/integration tests using **Mocha**.
  - Running code coverage using **c8** and **Mocha**.

Documentation & Git/GitHub Usage - Learning Outcomes 2, 3 (15%)

- Project board to help you organise and prioritise your work.
- A **Entity-Relationship** diagram of your **Prisma** schema.
- **GitHub** repository contains a **Node.js .gitignore**.
- Provide the following in your repository **README.md** file:
  - URL to the **RESTful API** on **AWS**.
  - How do you setup the development environment, i.e., after the repository is cloned, what do you need to do before you run the **RESTful API**?
  - How do you deploy the **RESTful API** to **AWS**
  - How do you open **Prisma Studio**?
  - How do you create a migration?
  - How do you lint and fix your code?
  - How do you format your code?
  - How do you run your **API**/integration tests?
  - How do you run your code coverage & output the results to **HTML**?
- Use of **Markdown**, i.e., headings, bold text, code blocks, etc.
- Correct spelling and grammar.
- Your **Git commit messages** should:
  - Reflect the context of each functional requirement change.
  - Be formatted using an appropriate naming convention style.

Additional Information

- **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.