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

Assessment: Node.js RESTFul API, AZ-900: Microsoft Azure Fundamentals & SC-900: Microsoft Security, Compliance, and Identity Fundamentals

Assessment Overview

In this **individual** assessment, you will develop a **RESTful API** using **Node.js** & deploy it to **Heroku**. Also, study & sit the **AZ-900: Microsoft Azure Fundamentals** & **SC-900: Microsoft Security, Compliance, and Identity Fundamentals** exams.

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
AZ-900: Microsoft Azure Fundamentals	20%	17-02-2023	1
SC-900: Microsoft Security, Compliance, and Identity Fundamentals	20%	17-02-2023	1
Node.js RESTful API	60%	17-02-2023	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 **Friday, 17 February 2022** at **12.00 PM**.

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**.

Node.js RESTful API (60% towards your final mark)

Overview

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.

Functionality - Learning Outcomes 1, 2, 3 (50%)

- **User:**
  - You will have **two** types of users - **admin** & **basic** user.
  - Each user will have the following information: first name, last name, username, email address, profile picture, password, confirm password & role. The users' profile picture will be from the following **API** - https://avatars.dicebear.com/docs/http-api. Each profile picture should be, in most cases, different. I suggest using a random seed when setting the user's profile picture.
  - Each user can login, get their information & update their information. An **admin** user can get all **admin** & **basic** users' information & 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 & 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 & alphanumeric characters only, i.e., johndoe123.
    - Email address is unique, contains the username above, an @ special character & a second-level domain, i.e., johndoe123@email.com.
    - Password has a minimum length of eight characters, maximum length of 16 characters & contains one numeric character & one special character.
    - Confirm password is the same as the password above. **Note:** Confirm password will not be a field in the **User** table. Rather, it will be used to validate the user's password.

    For each error check, a status code & 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" & the user's **JWT**.
  - When performing a **PUT** & **DELETE** request, return a status code & a response message, i.e., "<User's username>'s information has successfully updated" or "<User's username> has successfully deleted".
  - Seed two **admin** users. The **admin** users' data will be fetched from a private **GitHub Gist** using **Axios** & inserted into the **User** table using **Prisma**.
  - Two **basic** users are seeded via a an **admin** user. Only an **admin** user can seed the five **basic** users. The **basic** users' data will be fetched from a private **GitHub Gist** using **Axios** & 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 & overall winner. The category, list of questions, list of correct answers & list of incorrect answers will be fetched from the following **API** - https://opentdb.com/api_config.php. The difficulties will be easy, medium & hard. The types will be multiple choice or true/false.
  - Each user can get a list of scores. An **admin** user can create & delete a quiz. A **basic** user can participate in 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 & alpha characters only.
    - Start date has to be greater than today's date.
    - End date has to be greater than the start date & no longer than five days.
    - Number of questions has to be ten.

    For each error check, a status code & 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 & 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 & 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** & **rate limiting**.

- **Testing:**
  - API tests are written using **Mocha** & **Chai**.
  - At least 15 **API/integration** tests verifying the user & quiz functionality.
  - Code coverage using **c8**.

- **Deployment:**
  - **RESTful API** is deployed to **Heroku**.

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

Code Elegance - Learning Outcome 1 (35%)

- Environment variables' key is stored in the **env.example** file.
- **PostgreSQL** databases configured for development & production environments.
- Variables, functions & resource groups are named appropriately.
- Idiomatic use of control flow, data structures & in-built functions.
- File header comment for each controller & route file explaining its purpose using **JSDoc**.
- Code is linted & formatted using **ESLint** & **Prettier**.
- **Pre-commit hook** for **ESLint** & **Prettier** using **Husky**.
- **Mocha**, **Chai**, **c8**, **ESLint**, **Prettier**, **Husky** & **Commitizen** are installed as **development dependencies**.

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

- **GitHub** project board to help you organise & prioritise your work.
- Provide the following in your repository **README.md** file:
  - A **Entity-Relationship** diagram of your **Prisma** schema.
  - URL to the **RESTful API** on **Heroku**.
  - 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 **Heroku**
  - How do you open **Prisma Studio**?
  - How do you create a migration?
  - How do you lint & 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 & grammar.
- Your **Git commit messages** should:
  - Reflect the context of each functional requirement change.
  - Be formatted using an appropriate naming convention style using **Commitizen**.

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.

AZ-900: Microsoft Azure Fundamentals (20% towards your final mark)

You are required to study & sit the **AZ-900: Microsoft Azure Fundamentals** exam, & reflect on your experience.

SC-900: Microsoft Security, Compliance, and Identity Fundamentals (20% towards your final mark)

You are required to study & sit the **SC-900: Microsoft Security, Compliance, and Identity Fundamentals** exam, & reflect on your experience.