College of Engineering, Construction & Living Sciences
Bachelor of Information Technology
ID607001: Introductory Application Development Concepts
Level 6, Credits 15

Practical: Node.js REST API Testing

Assessment Overview

In this **individual** assessment, you will test the **Node.js REST API** you created in the **Project 1: Node.js REST API** assessment. In addition, marks will be allocated for code elegance, documentation & **Git** usage.

Learning Outcome

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

1. Design & build secure applications with dynamic database functionality following an appropriate software development methodology.

Assessments

Assessment	Weighting	Due Date	Learning Outcomes
Practical: Node.js REST API Testing	20%	05-05-2023 (Friday at 4.59 PM)	1
Project 1: Node.js REST API	30%	05-05-2023 (Friday at 4.59 PM)	1
Project 2: React CRUD	50%	16-06-2023 (Friday at 4.59 PM)	1

Conditions of Assessment

You will complete this assessment during your learner-managed time. However, there will be time to discuss the requirements & your assessment progress during the teaching sessions. This assessment will need to be completed by **Friday, 05 May 2023 at 4.59 PM**.

Pass Criteria

This assessment is criterion-referenced (CRA) with a cumulative pass mark of **50%** across all assessments in **ID607001: Introductory Application Development Concepts**.

Submission

You must submit all program files via **GitHub Classroom**. Here is the URL to the repository you will use for your submission – https://classroom.github.com/a/fZCB58Sl. Create a **.gitignore** and add the ignored files in this resource - https://raw.githubusercontent.com/github/gitignore/main/Node.gitignore. The latest program files in the **master** or **main** branch will be used to mark against the **Functionality** criterion. Please test your **master** or **main** branch application before you submit. Partial marks **will not** be given for incomplete functionality. Late submissions will incur a **10% penalty per day**, rolling over at **5:00 PM**.

Authenticity

All parts of your submitted assessment **must** be completely your work. Do your best to complete this assessment without **ChatGPT**. You need to demonstrate to the course lecturer that you can meet the learning outcome for this assessment.

However, if you get stuck, you can use **ChatGPT** to help you get unstuck, permitting you acknowledge that you have used **ChatGPT**. In the assessment's repository **README.md** file, please include what prompt(s) you provided to **ChatGPT** & how you used the response(s) to help you with your work. It also applies to code snippets retrieved from **StackOverflow** & **GitHub**. 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 — Te Pūkenga** policies. Learners can view policies on the **Otago Polytechnic — Te Pūkenga** website located at https://www.op.ac.nz/about-us/governance-and-management/policies.

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 **ID607001: Introductory Application Development Concepts**.

Instructions

You will need to submit a **suite of API tests** & documentation that meet the following requirements:

Functionality - Learning Outcome 1 (40%)

- **Testing:**
  - **API tests** are written using **Mocha** & **Chai**.
  - At least **40 API tests** verifying the correctness for the following:
    - CRUD (create, read, update & delete) operations.
    - Validation rules, i.e., checking if field is required, etc.
    - Query parameters, i.e., filtering, sorting & paging data.
    - Status codes, i.e., checking if a response returns 200, 404, etc.
    - Shape of the data, i.e., does the response data contain a specific field?
- **NPM Scripts:**
  - Linting & fixing your code using **ESLint**.
  - Formatting your code using **Prettier**.
  - Running **API** tests using **Mocha**.

Code Elegance - Learning Outcome 1 (45%)

- Environment variables' key is stored in the **env.example** file.
- Database configured for the testing environment.
  - Create a new database called **test.db**. **Note:** Do not use **dev.db**
- Appropriate naming of variables & functions.
- Idiomatic use of control flow, data structures & in-built functions.
- Efficient algorithmic approach.
- Sufficient modularity, i.e., **before()** & **after()** functions.
- Each **test** file **must** have a header comment located immediately before the **import** statements.
- In-line comments where required.
- Code is linted & formatted using **ESLint** & **Prettier**.
- **Mocha**, **Chai**, **ESLint** & **Prettier** are installed as **development dependencies**.
- No dead or unused code.

Documentation & Git Usage - Learning Outcome 1 (15%)

- Provide the following in your repository **README.md** file:
  - How do you setup the testing environment, i.e., after the repository is cloned, what do you need to do **before** you run the **API tests**?
  - How do you lint & fix your code?
  - How do you format your code?
  - How do you run your **API tests**?
- 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.

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.