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

Project 1: Node.js REST API

Assessment Overview

In this **individual** assessment, you will develop a **REST API** using **Node.js** & deploy it to **Heroku**. You will choose the theme of your **REST API**. It could be on sport, culture, food or something else you are interested in. Your **REST API** data will be stored in a **MongoDB Atlas** database. The main purpose of this assessment is to demonstrate your ability to develop a **REST API** using taught concepts such as queries, relationships, validation & rate limits. However, you will be required to independently research & implement more complex concepts such as filtering, sorting, pagination & automated code formatting. 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.

Assessment Table

Assessment Activity	**Weighting**	**Learning Outcome**	**Assessment Grading Scheme**	**Completion Requirements**
Practical: Node.js REST API Testing Research	20%	1	CRA	Cumulative
Project 1: Node.js REST API	30%	1	CRA	Cumulative
Project 2: React CRUD	50%	1	CRA	Cumulative

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 **Thursday, 15 September 2022 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/hWjmBeNq. 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. If you use code snippets from **GitHub**, **StackOverflow**, your fellow learners, 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.

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 **seven days** 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 **REST API** & documentation that meet the following requirements:

Note: Independent research requirements are highlighted yellow.

Functionality - Learning Outcome 1 (40%)

- **REST API** is developed using **Node.js** & can run locally without modification.
- **Five collections** containing at least **three fields** of data with different types which you can interact with.
- **Three relationships** between **collections**.
- A separate **controller** & **route** file for each collection. Each **controller** file must contain operations for **CRUD** (Create, Read one, Read all, Update & Delete).
- Each **field** of data has custom validation.
- Return success, i.e., true or false & data when performing **CRUD** operations.
- Secure **HTTP** headers using **Helmet**.
- Enable cross-origin **HTTP** requests using **Cors**.
- **REST API** is deployed to **Heroku**. The **REST API should** be usable i.e., a consumer should be able to perform operations on your **REST API**.
- **REST API data** is stored in a **MongoDB Atlas** database.
- **Independent Research:**
  - The **index route**, i.e., **/api** must display all of the available **routes** in the **REST API**.
  - **REST API** version is set to **v1**. For example, an endpoint should look like **/api/v1/items**.
  - Return a success & failure message when performing **CRUD** operations, i.e., **"Successfully created an institution"** or **"Something went wrong while creating an institution"**.
  - Filter & sort **REST API data** using query parameters. A consumer should be able to filter all **fields** of data & sort **fields** of data in ascending & descending order.
  - Return an appropriate message if a request does not return any **REST API data**, i.e., do not display an empty array.
  - Return an appropriate message if an endpoint does not exist.
  - Paginate the **REST API data** so that any number of records can be displayed per page. The default number is 10 records per page.
  - **REST API** rate limit is set to 50 requests per minute. You must display the following message if the request limit exceeds 50 - **"You have exceeded the number of requests per minute: 50. Please try again later."**

Code Elegance - Learning Outcome 1 (40%)

- Use of intermediate variables, idiomatic use of control flow, data structures, in-built functions & sufficient modularity.
- Functions & variables are named appropriately.
- Efficient algorithmic approach, i.e., using the appropriate function(s) when querying your **collections**.
- **REST API** resource groups named with a plural noun instead of a noun or verb, i.e., **/api/v1/items** not **/api/v1/item**.

- File header comments using **JSDoc**. You **need** to explain the purpose of each **controller** & **route** file.
- In-line comments using **JSDoc**. You **need** to explain complex logic that is not obvious.
- Declare a **npm** script in your **package.json** file which seeds the **collections**.
- No dead or unused code.
- Database configured for the development & production environments.
- Environment variables are stored in a **.env** file.
  - Create **example.env** file containing all of the environment variables' key.
  - Do not include the environment variables' value.
- Independent Research:
  - Code files are formatted using **Prettier** & a **.prettierrc** file. You **need** to declare a **npm** script in your **package.json** file which automates this process. Rules **should** include:
    - Single quote is set to **true**.
    - Semi-colon is set to **false**.
    - Tab-width is set to **2**.
  - **Prettier** is installed as a development dependency.

Documentation & Git Usage - Learning Outcome 1 (20%)

- **REST API** is documented using **Postman**.
  - You **should** provide an example for each route. However, you **should** provide **one** example of filtering, sorting & paging.
  - Each example **should** contain a description, request & response.
- Provide the following in your repository **README.md** file:
  - URL to the documented **REST API** on **Postman**.
  - URL to the **REST 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 **REST API**?
  - How do you deploy the **REST API** to **Heroku**?
  - How do you seed the collections?
  - How do you format the code using **Prettier**?
- 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 the naming conventions outlined in the following:
    - **Resource:** https://dev.to/i5han3/git-commit-message-convention-that-you-can-follow-1709

Additional Information

- Attempt to commit at least **10** times per week.
- **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.