College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
IN607: Introductory Application Development Concepts
Level 6, Credits 15

Project 1: Node.js REST API

Assessment Overview

In this 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, authentication, validation, seeding, caching & rate limits. However, you will be required to independently research and 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 usable, secure & attractive 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, 14 April 2022 at 4.59 PM**.

Pass Criteria

This assessment is criterion-referenced (CRA) with a cumulative pass mark of **50%** across all assessments in **IN607: Introductory Application Development Concepts**.

Submissions

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. The latest program files in the **main** branch will be used to mark against the **Functionality** criterion. Please test your **main** branch application before you submit. Partial marks **will not** be given for functionality in other branches. 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** or other online resource, 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 **IN607: Introductory Application Development Concepts**.

Instructions

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

Functionality - Learning Outcome 1 (40%)

- **REST API** is developed using **Node.js**.
- **REST API** can run locally without modification.
- **Five collections** containing at least **three fields** of data which you can interact with.
- A range of different data types, i.e., all **fields** of data can not be of a single type.
- Each **collection** has a separate **controller** & **route** file.
- A **controller** contains **CRUD** (Create, Read, Update & Delete) operations.
- Each **field** of data has custom validation when creating & updating a **document**.
- Each **collection** is seeded with a **JSON** file. **Note:** this is **only** for testing purposes.
- **Independent Research: REST API** version is set to **v1**. For example, an endpoint should look like **/api/v1/items**
- Return an appropriate status code & message when performing **CRUD** operations. For example, when a **document** is created, return 200 & **document** successfully created.
- **Independent Research:** Return an appropriate message if a query does not return any **REST API data**.
- **Independent Research:** Return an appropriate message if an endpoint does not exist.
- **Independent Research:** 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.
- **Independent Research:** Paginate the **REST API data** so that 25 records are displayed per page.
- **POST**, **PUT** & **DELETE** routes are protected using **JSON Web Tokens (JWT)**.
- **REST API** rate limit is set to 25 requests per minute.
- **REST API** is deployed to **Heroku**. The **REST API** must 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.

Code Elegance - Learning Outcome 1 (45%)

- Use of intermediate variables. No method calls as arguments.
- Idiomatic use of control flow, data structures & in-built functions.
- Sufficient modularity, i.e., reusable base URL.
- 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 verb, i.e., **/api/v1/items** not **/api/v1/item**.
- File header comment explaining the purpose of each **controller** & **route** file.

- In-line comments explaining complex logic.
- **Independent Research:** Code files are formatted using **Prettier**. You **need** to declare a **npm** script in your application's **package.json** file which automates this process. Rules **must** include:
  - Single quote is set to **true**.
  - Semi-colon is set to **false**.
  - Tab-width is set to **2**.
- **Independent Research: Prettier** is installed as a development dependency.
- No dead or unused code.
- Database configured for production environment.

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

- **REST API** is documented using **Postman**.
- 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 environment for development, i.e., after the repository is cloned, what do you need to run the **REST API** locally?
  - How do you deploy the **REST API** to **Heroku**?
- 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
- At least **10** feature branches excluding the **main** branch.
  - Your branches must be prefix with **feature**, for example, **feature-<name of functional requirement>**.
  - For each branch, merge your own pull request to the **main** branch. **Note:** if you want ongoing feedback, assign the **GitHub** user **grayson-orr** to a reviewer.

Additional Information

- Attempt to commit at least **10** times per week. By the end of this assessment, you should have at least **50** commits.
- **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.