



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

## Assessment Overview

In this **individual** assessment, you will test the **REST API** you created in the **Project** assessment. In addition, marks will be allocated for code quality and best practices, documentation and **Git** usage.

## Learning Outcome

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

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

## Assessments

Assessment	Weighting	Due Date	Learning Outcome
Practical	20%	21-06-2024 (Friday at 4.59 PM)	1
Project	80%	21-06-2024 (Friday at 4.59 PM)	1

## Conditions of Assessment

You will complete this assessment during your learner-managed time. However, there will be time during class to discuss the requirements and your progress on this assessment. This assessment will need to be completed by **Friday, 21 June 2024 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 application files via **GitHub Classroom**. Here is the URL to the repository you will use for your submission – <https://classroom.github.com/a/wlZE5yYo>. If you do not have not one, create a **.gitignore** and add the ignored files in this resource - <https://raw.githubusercontent.com/github/gitignore/main/Node.gitignore>. Create a branch called **practical**. The latest application files in the **practical** 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 **5:00 PM**.

## Authenticity

All parts of your submitted assessment **must** be completely your work. Do your best to complete this assessment without using an **AI generative tool**. 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 an **AI generative tool** to help you get unstuck, permitting you to acknowledge that you have used it. In the assessment's repository **README.md** file, please include what prompt(s) you provided to the **AI generative tool** and how you used the response(s) to help you with your work. It also applies to code snippets retrieved from **StackOverflow** and **GitHub**.

Failure to do this may result in a mark of **zero** for this assessment.

## Policy on Submissions, Extensions, Resubmissions and Resits

The school's process concerning submissions, extensions, resubmissions and 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. Extensions will **only** be granted if you are unable to complete the assessment by the due date because of **unforeseen circumstances outside your control**. The length of the extension granted will depend on the circumstances and **must** be negotiated with the course lecturer before the assessment due date. A medical certificate or support letter may be needed. Extensions will not be granted for poor time management or pressure of other assessments.

## Resits

Resits and reassessments are not applicable in **ID607001: Introductory Application Development Concepts**.

## Instructions

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

### Functionality - Learning Outcome 1 (50%)

- **Testing:**
  - **API tests** are written using **Mocha** and **Chai**.
  - **API tests** verifying the correctness for the following:
    - \* **GET one**, **GET all**, **POST**, **PUT** and **DELETE** operations. (30 tests).
    - \* **Index route** displaying all existing **routes**. (one test).
    - \* A **route** that does not exist. (one test).
    - \* Validation for **POST** and **PUT** operations. (12 tests).
- **Scripts:**
  - Seed your database with **Prisma**.
  - Run your **API tests** using **Mocha**.

### Code Quality and Best Practices - Learning Outcome 1 (45%)

- Appropriate naming of files, variables and functions.
- Idiomatic use of control flow, data structures and in-built functions.
- Efficient algorithmic approach.
- Sufficient modularity.
- Each **test** file has a **JSDoc** header comment located at the top of the file.
- Code is formatted using **Prettier**.
- **Mocha** and **Chai** are installed as **development dependencies**.
- No dead or unused code.

### Documentation and Git Usage - Learning Outcome 1 (5%)

- Provide the following in your repository **README.md** file:
  - How to seed your database with **Prisma**?
  - How do you run your **API tests**?
- 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

- You do not need to test the **filtering**, **sorting** and **pagination** operations.
- **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.