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

Practical: API Testing Research

Assessment Overview

In this assessment, you will be given a **Laravel API** application to test. You will be required to independently research & write at least 40 API tests using **PHPUnit** which verify the correctness of the given application. This includes **CRUD** functionality, query parameters, status codes & the shape of response data. You will also check the coverage of your tests using **php-code-coverage**. 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: API Testing Research	20%	1	CRA	Cumulative
Project 1: Laravel 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 availability during the teaching sessions to discuss the requirements & your progress of this assessment. This assessment will need to be completed by **Wednesday, 20 October 2021 at 5:00 PM**.

Pass Criteria

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

Authenticity

All parts of your submitted assessment must be completely your work & any references must be cited appropriately. Provide your references in a **README.md** file. 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.

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/0kYlKqW8. 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 **are not** given for functionality in other branches. Late submissions will incur a **10% penalty per day**, rolling over at **5:00 PM**.

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 **IN607: Introductory Application Development Concepts**.

Instructions

You will need to submit an application & documentation that meet the following requirements:

Functionality - Learning Outcome 1 (60%)

- At least 40 API tests verifying the correctness of the following:
  - CRUD (create, read, update & delete) functionality.
  - Validation rules, i.e., checking if field is required, etc.
  - Query parameters, i.e., filtering & sorting 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 column?
- Code covered using **php-code-coverage**.

Code Elegance - Learning Outcome 1 (30%)

- Use of intermediate variables. No method calls as arguments.
- Sufficient modularity, i.e., setup method at the beginning of each test case.
- Idiomatic use of control flow, data structures & in-built functions.
- Adheres to an **OO** architecture, i.e., classes, methods & variables are named appropriately.
- Code files are formatted.
- No dead or unused code.
- Databases configured for testing environment.

Documentation & Git Usage - Learning Outcome 1 (10%)

- Provide the following in your repository **README.md** file:
  - How do you setup the environment for development, i.e., after the repository is cloned, what do you need to run the the tests locally?
  - How do you run the tests?
- Commit messages must reflect the context of each functional requirement change. **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.