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

Project 1: Laravel API

Assessment Overview

In this assessment, you will develop an **API** using **Laravel** & deploy it to **Heroku**. You will choose the theme of your **API**. This could be on sport, culture, food or something else you are interested in. Your **API** data will be stored in a **MySQL** development database & **Heroku PostgreSQL** production database. The main purpose of this assessment is to demonstrate your ability to develop an **API** using concepts taught in class such as queries, relationships, validation, seeders, resources, caching, observers & rate limits. However, you will be required to independently research and implement more complex concepts such as messaging, filtering, sorting & pagination. 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 **Friday, 17 September 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/c1Wxock6. 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 (40%)

- **API** application can run locally without modification.
- Five **Models** containing at least four columns of data which you can interact with.
  - You must have a range of different data types, i.e., all columns of data can not be of type **string**.
  - **Independent Research:** For updating & deleting, if the child table has a foreign key, you must use an on cascade update & on cascade delete. For example, if data in the parent table is updated or deleted, then data in the child/children table is updated or deleted as well.
- **Controller** for each **Model** which have **CRUD** (create, read, write & delete) functionality.
- Custom validation when creating & updating a record.
- Seed each table using a **Seeder** & **JSON** file.
- **Independent Research:** Return an appropriate status code & message when performing **CRUD** actions. For example, when a record is created, return 200 & record successfully created.
- **Independent Research:** Return an appropriate message if a query does not return any data.
- Return data except for **id**, **created_at** & **updated_at** using **API Resources**.
- **Independent Research:** Filter & sort **API** data on all columns using query parameters. A user should be able to sort **API** data in ascending & descending order.
- **Independent Research:** Paginate the **API** data so that 25 records are displayed per page.
- Store data in the cache when a **GET** request is performed.
- Remove data from the cache when a **POST** request is performed.
- **POST**, **PUT** & **DELETE** routes are protected using **Sanctum**.
- Set the **API** rate limit to 25 requests per minute.
- **API** application deployed to **Heroku**. The application must be usable i.e., a user should be able to perform requests to your **API**.
- **API** data is stored in a **MySQL** development database & **Heroku PostgreSQL** production 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.
- Adheres to an **OO** architecture, i.e., classes, methods & variables are named appropriately.
- Efficient algorithmic approach, i.e., using the appropriate **Eloquent** function when querying your **Models**.
- **API** resource groups named with a plural noun instead of a verb, i.e., **/api/students** not **/api/student**.
- Header comments explain each **CRUD** action in a **Controller**.
- In-line comments explaining complex logic, i.e., an **Eloquent** function that may need additional explanation.
- Code files are formatted.
- No dead or unused code.
- Databases configured for development & production environments.

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

- **API** documented using **Postman**.
- Provide the following in your repository **README.md** file:
  - URL to the **API** application on **Heroku**.
  - URL to the **API** documentation on **Postman**.
  - How do you setup the environment for development, i.e., after the repository is cloned, what do you need to run the **API** application locally?
  - How do you deploy the **API** application to **Heroku**?
- 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.