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

Project 2: React CRUD

Assessment Overview

In this assessment, you will develop a **CRUD** application using **React** & deploy it to **Heroku**. This application will consume the **Laravel API** you created in the **Project 1: Laravel API** assessment. The main purpose of this assessment is not just to build a full-stack application, rather to demonstrate an ability to decouple the presentation layer (**frontend**) from the business logic (**backend**). Also, you will be required to independently research and implement pagination, deployment & automated code formatting. In addition, marks will be allocated for code elegance, documentation & **Git** usage.

Learning Outcomes

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 Outcomes**	**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, 19 November 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/PZJYGNeP. The latest program files in the **main** branch will be used to run your application. 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 Outcomes 1 (40%)

- Request **API** data from at least three **API** resource groups using **Axios**.
- Create new **API** data via a form. You **must** display the form in a modal.
- Incorrectly formatted form field values handled gracefully using validation error messages, i.e., **first name** form field is required.
- View **API** data in a table using an id & a variety of query parameters.
- Update **API** data via a form. Similar to creating **API** data, you **must** display the form in a modal.
- Delete **API** data. Prompt the user for deletion. You **can not** use the in-built **confirm() JavaScript** function.
- Data should automatically re-render, i.e., the user should not have to refresh the browser to see the created, updated and deleted data.
- **Independent Research:** Paginate **API** data across several pages with **next** & **previous** buttons or links.
- Visually attractive user-interface with a coherent graphical theme & style using **Reactstrap**.
- **Independent Research:** Application deployed to **Heroku**.
- Five component tests using **React Testing Library**.

Code Elegance - Learning Outcomes 1 (45%)

- Idiomatic use of control flow, data structures & in-built functions.
- Sufficient code modularity, i.e., UI split into independent reusable pieces.
- Components written as functional, not class.
- Adheres to a client-server architecture, i.e., the presentation layer (frontend application) is separate from the business logic (backend application).
- Header & in-line comments explaining complex logic.
- **Independent Research:** Code files are formatted using **Prettier**. You **must** declare a **npm** script in your application's **package.json** file which automates this process.
- No dead or unused code.

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

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

- 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.