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

Project 2: React CRUD

Assessment Overview

In this **individual** assessment, you will develop a **CRUD** application using **React**. This application will consume the **REST API** you developed in the **Project 1: Node.js REST 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**). In addition, marks will be allocated for code elegance, 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 Outcomes
Practical: Node.js REST API Testing	20%	11-09-2023 (Monday at 04.59 PM)	1
Project 1: Node.js REST API	40%	11-09-2023 (Monday at 04.59 PM)	1
Project 2: React CRUD	40%	13-11-2023 (Monday at 04.59 PM)	1

Conditions of Assessment

You will complete this assessment during your learner-managed time. However, there will be time to discuss the requirements and your assessment progress during the teaching sessions. This assessment will need to be completed by **Monday, 13 November 2023 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 program files via **GitHub Classroom**. Here is the URL to the repository you will use for your submission – https://classroom.github.com/a/wJ4pC7Y7. Create a **.gitignore** and add the ignored files in this resource - https://raw.githubusercontent.com/github/gitignore/main/Node.gitignore. The latest program files in the **master** or **main** branch will be used to mark against the **Functionality** criterion. Please test your **master** or **main** branch application 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 a **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 a **AI generative tool** to help you get unstuck, permitting you acknowledge that you have used **AI generative tool**. 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.

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 and 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 and achieved a **D grade (40-49%)**. The maximum grade awarded for resubmission will be **C-**.

Resits

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

Instructions

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

Functionality - Learning Outcome 1 (40%)

- **CRUD:**
  - Request **REST API** data from at four three **API** resource groups using **Axios**.
  - Create new **REST API** data via a button and form.
  - View **REST API** data in a table.
  - Update **REST API** data via a button and form.
  - Delete **REST API** data via a button. Prompt the user for deletion. You **can** use the in-built **confirm() JavaScript** function.
  - Incorrectly formatted form field values handled gracefully using validation error messages.
  - UI is visually attractive with a coherent graphical theme and style.
- **Scripts:**
  - Formatting your code using **Prettier**.

Code Elegance - Learning Outcome 1 (45%)

- A **Node.js .gitignore** file is used.
- Environment variables' key is stored in the **env.example** file.
- Appropriate naming of files, variables, functions and components.
- Idiomatic use of control flow, data structures and in-built functions.
- Efficient algorithmic approach.
- Sufficient modularity.
- Each **component** file **must** have a **JSDoc** header comment located immediately before the **import** statements.
- In-line comments where required. It should be for code that needs further explanation.
- Code is formatted using **Prettier**.
- **Prettier** is installed as a **development dependency**.
- No dead or unused code.

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

- Provide the following in your repository **README.md** file:
  - How do you setup the environment, i.e., after the repository is cloned?
  - How do you format your code?
- 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

- **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.