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 assessment, you will develop a **CRUD** application using **React** & deploy it to **Heroku**. This application will consume an API from either the **Project 1: Node.js REST API** assessment or the **Practical: REST API Testing Research**. 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 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: Node.js REST API Testing Research	20%	1	CRA	Cumulative
Project 1: Node.js REST 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 time to discuss the requirements & your assessment progress during the teaching sessions. This assessment will need to be completed by **Tuesday, 21 June 2022 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/Vq7T0W6E. 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. If you use code snippets from **GitHub**, **StackOverflow** or other online resource, you **must** reference it appropriately using **APA 7th edition**. Provide your references in the **README.md** file in your repository. 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.

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 **seven days** 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 **ID607001: Introductory Application Development Concepts**.

Exemplar

You can find an exemplar here - https://id607001-graysono-frontend.herokuapp.com.

- Email: graysono@op.ac.nz
- Password: P@ssw0rd123

Instructions

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

Functionality (Features) - Learning Outcome 1 (40%)

- Authentication
  - **Independent Research:** Register a new user via a form.
  - Login an existing user via a form.
  - Log out of the application.
- CRUD
  - Request **REST API data** from at least three **API** resource groups using **Axios**.
  - Create new **REST API data** via a form. You can display the form on the page or in a modal.
  - View **REST API data** in a table.
  - **Independent Research:** View **REST API data** in a table using an id. For example, **/institutions/1** would return **REST API data** for that specific **Institutions** object.
  - **Independent Research:** Update **REST API data** via a form. Similar to creating **REST API data**, you can display the form on the page or in a modal.
  - **Independent Research:** Delete **REST API data**. Prompt the user for deletion. You **can** use the in-built **confirm() JavaScript** function.
  - **Independent Research:** Incorrectly formatted form field values handled gracefully using validation error messages, i.e., **first name** form field is required.
- **Independent Research:** Paginate **REST API data** across several pages with **next** & **previous** buttons or links. You can choose the number of **REST API data** per page.
- **Independent Research:** Search **REST API data** via a search bar.
- User-interface is visually attractive with a coherent graphical theme & style using **Reactstrap**.
- Application deployed to **Heroku**.
- End-to-end **Cypress** tests that ensures the register, login and logout functionality is working as expected. You **need** to declare a **npm** script in your application's **package.json** file that automates this process.

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.
- Sufficient modularity, i.e., UI split into independent reusable pieces.
- Functions & variables are named appropriately.
- Components written as functional, not class.
- Adheres to a client-server architecture, i.e., the frontend is separate from the backend.
- File header comment explaining the purpose of each **component** file.
- In-line comments explaining complex logic.
- Code files are formatted using **Prettier**. You **need** to declare a **npm** script in your application's **package.json** file that automates this process. Rules **must** include:
  - Single quote is set to **true**.
  - Semi-colon is set to **false**.
  - Tab-width is set to **2**.
- **Prettier** & **Cypress** are installed as development dependencies.
- No dead or unused code.

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

- Provide the following in your repository **README.md** file:
  - URL to the 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 run the end-to-end **Cypress** tests?
  - How do you deploy the **React** application to **Heroku**?
- Your **Git commit messages** should:
  - Reflect the context of each functional requirement change.
  - Be formatted using the naming conventions outlined in the following:
    - **Resource:** https://dev.to/i5han3/git-commit-message-convention-that-you-can-follow-1709

Additional Information

- Attempt to commit at least **10** times per week.
- **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.