College of Engineering, Construction & Living Sciences
Bachelor of Information Technology
ID608001: Intermediate Application Development Concepts
Level 6, Credits 15

Assessment 2: React Application - The Movie DB

Assessment Overview

In this **individual** assessment, you will replicate a given application using **React**. The main purpose of this assessment is to demonstrate your ability to replicate an existing application using various taught frontend concepts. However, you will be required to independently research & implement more complex concepts. 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. Apply design patterns & programming principles using software development best practices.
2. Design & implement full-stack applications using industry relevant programming languages.

Assessments

Assessment	Weighting	Due Date	Learning Outcomes
Practical: Skills-Based	20%	27-10-2022 (Thur at 10.00 AM)	1
Assessment 1: Node.js RESTful API - Open Trivia DB	45%	17-10-2022 (Mon at 7.59 AM)	1 & 2
Assessment 2: React Application - The Movie DB	35%	09-11-2022 (Wed at 2.59 PM)	1 & 2

Conditions of Assessment

You will complete this assessment during your learner-managed time. However, there will be time during class to discuss the requirements & your progress on this assessment. This assessment will need to be completed by **Wednesday, 09 November 2022** at **2.59 PM**.

Pass Criteria

This assessment is criterion-referenced (CRA) with a cumulative pass mark of **50%** over all assessments in **ID608001: Intermediate Application Development Concepts**.

Authenticity

All parts of your submitted assessment **must** be completely your work. If you use code snippets from **GitHub**, **StackOverflow** or other online resources, 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.

Submission

You **must** submit all project files via **GitHub Classroom**. Here is the URL to the repository you will use for your submission – https://classroom.github.com/a/Ys5M8rKj. Create a **.gitignore** & add the ignored files in this resource - https://raw.githubusercontent.com/github/gitignore/main/Node.gitignore. The latest project files in the **master** or **main** branch will be used to mark against the **Functionality** criterion. Please test before you submit. Partial marks **will not** be given for incomplete functionality. Late submissions will incur a **10% penalty per day**, rolling over at **3.00 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 **ID608001: Intermediate Application Development Concepts**.

Instructions

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

Functionality - Learning Outcomes 1, 2, 3 (50%)

- **Header:**
  - The **Header** component will render six icon components using the **@material-ui/core@4.11.2** & **@material-ui/icons@4.11.2** dependencies. The components are:

    - Home
    - FlashOn
    - LiveTv
    - PersonOutline
    - Search
    - VideoLibrary
  - When the cursor hovers on an icon, it will display the following:
    - Home icon - Home
    - FlashOn icon - Trending
    - LiveTv - Verified
    - PersonOutline - Collections
    - Search - Search
    - VideoLibrary - Account
  - **Resource:** https://www.npmjs.com/package/@material-ui/icons

- **Navigation Bar:**
  - The **Navbar** component will render eight items fetched from the given file - **endpoint.js**. In the **endpoint.js**, refer to the **type** key.
  - The eight items need to be centered.
  - When the cursor hovers on an item, it will change it's colour and font size.

- **Movie Card:**
  - You have been given the incomplete **MovieCard** component file. Using the given comments in the **MovieCard** component file, complete the following functionality:
    - Under the **img** element, render the movie's overview using the **react-text-truncate@0.16.0** dependency.
    - Under the **h1** element, render the movie's media type, release date or first air date & vote count in a **span** element.
  - For styling, the **MovieCard** component uses the **react-jss@10.5.0** dependency.
  - **Resources:**
    - https://www.npmjs.com/package/react-text-truncate
    - https://www.npmjs.com/package/react-jss

- **Dashboard:**
  - The **Dashboard** component will fetch a list of movies from the given file - **endpoint.js**. In the **endpoint.js**, refer to the **url** key.
  - For each movie in the list of movies, render the **MovieCard** component.

- **App:**
  - The **App** component will render the **Header** component, **Navbar** component & **Dashboard** component.

Code Elegance - Learning Outcome 1 (35%)

- Environment variables' key is stored in the **env.example** file.
- Variables, functions & components are named appropriately.
- Idiomatic use of control flow, data structures & in-built functions.
- Sufficient modularity, i.e., UI split into independent reusable pieces.
- File header comment for each component file explaining its purpose using **JSDoc**.
- Code is linted & formatted using **ESLint** & **Prettier**.
- **ESLint**, **Prettier** & **Commitizen** are installed as **development dependencies**.

Documentation & Git/GitHub Usage - Learning Outcomes 2, 3 (15%)

- **GitHub** project board to help you organise & prioritise your work.
- Provide the following in your repository **README.md** file:
  - How do you setup the development environment, i.e., after the repository is cloned, what do you need to do before you run the **React** application?
  - How do you lint & fix your code?
  - How do you format your code?
- Use of **Markdown**, i.e., headings, bold text, code blocks, etc.
- Correct spelling & grammar.
- Your **Git commit messages** should:
  - Reflect the context of each functional requirement change.
  - Be formatted using an appropriate naming convention style using **Commitizen**.

Additional Information

- In this repository, you have been given a directory called **expected-outputs**. In this directory, you are given expected output screenshots for this assessment
- **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.