



College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
ID608001: Intermediate Application Development Concepts
Level 6, Credits 15
Project 2: Node.js and Express Pub Quiz App

Assessment Overview

In this **individual** assessment, you will develop a **RESTful API** using **Express** and **Node.js**, and deploy it as a **web service** on **Render**. The main purpose of this assessment is to demonstrate your ability to develop a **RESTful API** using various taught concepts. In addition, marks will be allocated for code elegance, documentation and **Git** usage.

Due to a nation-wide lockdown, your local pub is no longer able to run their weekly quiz night onsite. Your local pub owners know you are an IT student and ask if you want create an online quiz night application for them. For the first step, the pub owners want a **RESTful API** that provides various functions for registering, logging in, participating in various quizzes and keeping track of scores so that they can give away prizes at the end of each quiz night.

Learning Outcome

At the successful completion of this course, learners will be able to:

1. Apply design patterns and programming principles using software development best practices.
2. Design and implement full-stack applications using industry relevant programming languages.

Assessments

Assessment	Weighting	Due Date	Learning Outcomes
Practical: Skills-Based	20%	23-08-2023 (Wednesday at 11.59 AM)	1
Project 1: Next.js Hacker News App	30%	11-09-2023 (Monday at 04.59 PM)	1 and 2
Project 2: Node.js and Express Pub Quiz App	50%	10-11-2023 (Friday at 04.59 PM)	1 and 2

Conditions of Assessment

You will complete majority of this assessment during your learner-managed time. However, there will be time during class to discuss the requirements and your progress on this assessment. This assessment will need to be completed by **Friday, 10 November 2023 at 4.59 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. Do your best to complete this assessment without using an **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 an **AI generative tool** to help you get unstuck, permitting you to acknowledge that you have used it. 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>.

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/nqzpoWQ3>. Create a **.gitignore** and 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 **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 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 **ID608001: Intermediate Application Development Concepts**.

Instructions

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

Functionality - Learning Outcomes 1, 2 (50%)

- **User:**

- You will have **two** types of users - **super admin** and **basic** user.
- A **super admin** and **basic** user will have the following information: first name, last name, username, email address, profile picture, password, confirm password and role. The users' profile picture will be from the following **API** - <https://avatars.dicebear.com/docs/http-api>. Each profile picture should be, in most cases, different. I suggest using a random seed when setting the user's profile picture.
- A **super admin** and **basic** user can login, get their information and update their information. A **super admin** user can get all users' information, update all **basic** users' information and delete all **basic** users. A **basic** user can register.
- When performing a **POST** request for registering a **basic** user, the following error checking needs to be implemented using **Joi**:
 - * First name has a minimum length of two characters, a maximum length of 50 characters and alpha characters only.
 - * Last name has the same error checking as first name above.
 - * Username is unique, has a minimum length of five characters, maximum length of ten characters and alphanumeric characters only, i.e., johndoe123.
 - * Email address is unique, contains the username above, an @ special character and a second-level domain, i.e., johndoe123@email.com.
 - * Password has a minimum length of eight characters, maximum length of 16 characters and contains one numeric character and one special character.
 - * Confirm password is the same as the password above. **Note:** Confirm password will not be a field in the **User** table. Rather, it will be used to validate the user's password.

For each error check, a status code and response message is returned, i.e., "First name must have a minimum length of two characters".

- When performing a **POST** request for logging in a user using either username/password or email address/password, return a status code, a response message, i.e., "<User's username> has successfully logged in" and the user's **JWT**.
- When performing a **PUT** and **DELETE** request, return a status code and a response message, i.e., "<User's username>'s information has successfully updated" or "<User's username> has successfully deleted".
- Five **super admin** users are seeded via a **script** in the **package.json** file. The **super admin** users' data will be fetched from a local file and inserted into the **User** table using **Prisma**.
- Five **basic** users are seeded via a **super admin** user. The **basic** users' data will be fetched from a private **GitHub Gist** using **Axios** and inserted into the **User** table using **Prisma**.

- **Quiz:**

- Each quiz will have the following information: name, start date, end date, category, difficulty, type, number of questions, list of questions, list of correct answers, list of incorrect answers, list of scores, average score and overall winner. The category, list of questions, list of correct answers and list of incorrect answers will be fetched from the following **API** - https://opentdb.com/api_config.php. The difficulties will be easy, medium and hard. The types will be multiple choice or true/false.
- A **super admin** and **basic** user can get all quizzes, get all past quizzes, get all present quizzes, get all future quizzes and get a list of scores. A **super admin** user can create and delete a quiz. A **basic** user can participate in a quiz only once.
- When performing a **POST** request for creating a quiz, the following error checking needs to be implemented using **Joi**:
 - * Name has a minimum length of five characters, a maximum length of 30 characters and alpha characters only.
 - * Start date has to be greater than or equal to today's date.
 - * End date has to be greater than the start date and no longer than five days.
 - * Number of questions has to be ten.

For each error check, a status code and response message is returned, i.e., "Name must have a minimum length of five characters".

- When performing a **POST** request for a **basic** user who is participating in a quiz, the following error checking needs to be implemented using **Joi**:
 - * Can not participate if quiz has not started or has ended.
 - * Answered all ten questions.
- When performing a **POST** request for a **basic** user who has participated in a quiz, return a status code, a response message, i.e., "<User's username> has successfully participated in <Quiz's name>", user's score and quiz's average score.

- **HTTP:**

- Headers are secured using **Helmet**.
- Implement **CORS**, **compression**, **caching** and **rate limiting**.

- **Testing:**

- **API/integration tests** are written using **Mocha** and **Chai**.
- 20 **API/integration tests** verifying the **User** and **Quiz** functionality.

- **Scripts:**

- Run your **API/integration tests** using **Mocha**.
- Create and apply a migration using **Prisma**.
- Reset your database using **Prisma**.
- Seed **super admin** users.
- Open **Prisma Studio**.
- Lint and fix your code using **ESLint**.
- Format your code using **Prettier**.

Code Elegance - Learning Outcome 1, 2 (40%)

- A **Node.js .gitignore** file is used.
- Environment variables' key is stored in the **env.example** file.
- Appropriate naming of files, variables, functions and resource groups.
 - Resource groups are named with a plural noun instead of a noun or verb, i.e., **/api/v1/items** not **/api/v1/item**.
- Idiomatic use of control flow, data structures and in-built functions.
- Efficient algorithmic approach.
- Sufficient modularity.
- Each **controller** and **route** file has a **JSDoc** header comment located at the top of the file.
- In-line comments where required. It should be for code that needs further explanation.
- Code is linted and formatted using **ESLint** and **Prettier**.
- **Mocha**, **Chai**, **ESLint**, **Prettier** and **Commitizen** are installed as **development dependencies**.
- No dead or unused code.

Documentation and Git/GitHub Usage - Learning Outcomes 1, 2 (10%)

- A **GitHub** project board to help you organise and prioritise your work.
- Provide the following in your repository **README.md** file:
 - A URL to your **RESTful API** as a **web service** on **Render**.
 - How do you setup the environment, i.e., after the repository is cloned, what do you need to do before you run the **RESTful API**?
 - How do you run your **REST API** locally?
 - How do you run your **API/integration tests**?
 - How do you create and apply a migration?
 - How do you reset your database?
 - How do you seed **super admin** users?
 - How do you open **Prisma Studio**?
 - How do you lint and fix your code?
 - 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 using **Commitizen**.

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.