**COMP229 – Web Application Development**

Assignment 2

# Portfolio Application – Node.js, Express REST APIs & MongoDB

Due Week #8 20th October 2025 @ 11:59 pm

Portfolio – Node.js, Express REST APIs & MongoDB **Maximum Mark: 100**

**Overview**: Create the Node.js Express exports REST APIs that interacts with MongoDB Database using Mongoose ODM for your Portfolio application. (Note: The Front-end of the application is already created in Assignment1).

# Instructions:

The Portfolio Application:

**PART I – Reference (Week2 & Week3 slides and Assignment1)**

1. Configure the Backend of your portfolio Application, you already have the frontend in Assignment1 – **(15 marks)**

Steps:

* 1. Using the power point slide and the final zip file provided from week3 – Configuring the backend and frontend of your application or week6 final zip folder in your course shell, make a copy of the folder (Ensure to still keep a copy for the purpose of our class use case) rename the folder you copied from mern\_skeleton as MyPortfolio.
  2. From Assignment 1 folder copy the client folder and replace it with the client folder in the new MyPortfolio folder in Assignment2.

NOTE1: You will have to delete the client folder in the MyPortfolio folder before pasting to avoid duplication.

NOTE2: At the terminal change the directory to client folder run the application by running the command **yarn dev or npm run dev.**

You will notice only the Frontend is running on port 5173. <http://localhost:5173/>

A screenshot of a web page

AI-generated content may be incorrect.

Open a new terminal, at the root run the command node server.js

You will notice the Backend running on port 5000 or 3000 or the port number you have used.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. Stop the server from running and also stop the frontend from running.
  2. To ensure the frontend and Backend run concurrently open a new terminal.
  3. At the terminal change the directory to client folder and install nodemon and concurrently to be able to run it concurrently at the same time.

yarn add --dev nodemon.

* 1. Open the package.json in the client in the scripts section update the “dev”:”vite” with  "dev": "concurrently 'vite' \"nodemon ../server.js\"",
  2. Open a new terminal, cd client, then run **yarn dev**

A screen shot of a computer

AI-generated content may be incorrect.

**PART II – Reference (Week4 slides)**

1. Using MongoDB database, create:**(25 Marks):**
   1. A database by name Portfolio**.**
   2. Create the following collections with their respective property. (5 Marks: Functionality).
2. **contacts**

firstname: string

lastname: string

email: string

1. **projects**

title: string

firstname: string

lastname: string

email: string

completion: Date

description: string

1. **educations or qualifications**

title: string

firstname: string

lastname: string

email: string

completion: Date

description: string

1. **users**

name: string

email: string

password: string

created: Date

updated: Date

* 1. Obtain your connection string ( url or uri)
  2. Provide the screen snapshot of your MongoDB database showing the above steps from 1a – c.

1. Configure the Backend of your portfolio Application by creating the server.js file in the root folder for your server and a server folder or backend folder in the root folder for all your backend code.
   1. Update the server.js file i.e the server with code to display the message shown in the snapshot below when you run the app and provide a screen snapshot of it running in the browser as follows:

A screenshot of a computer

Description automatically generated

1. Create the web server after creating the Express web server next: **(30 Marks)**
   1. Add the configuration for the MongoDB database, ensure your application is connected to the database and give the screen snapshot of the connection from your console.
   2. Create contact, education, project and user model with Mongoose.

**PART III – Reference (Week5 slides)**

* 1. Write the controller for contact, education, project and user.
  2. Define the routes for handling all CRUD operations for the contact, education, project and users api.

Below is an overview of the REST APIs that will be exported:

**contacts**

You already have the fields from your contact from in Assignment 1.

The fields are as follows

**Field Data type**

**firstname string**

**lastname string**

**email string**

|  |  |  |
| --- | --- | --- |
| **Methods** | **Urls** | **Actions** |
| GET | api/contacts | get all contacts |
| GET | api/contacts/:id | get contacts by id |
| POST | api/contacts | add new contact |
| PUT | api/contacts/:id | update contact by id |
| DELETE | api/contacts/:id | remove contact by id |
| DELETE | api/contacts | remove all contacts |

**projects**

**The fields are as follows**

**Field Data type**

**title string**

**firstname string**

**lastname string**

**email string**

**completion Date**

**description string**

|  |  |  |
| --- | --- | --- |
| **Methods** | **Urls** | **Actions** |
| GET | api/projects | get all projects |
| GET | api/projects/:id | get projects by id |
| POST | api/projects | add new project |
| PUT | api/projects/:id | update project by id |
| DELETE | api/projects/:id | remove project by id |
| DELETE | api/projects | remove all projects |

**For Education you can call your collection qualifications**

**qualifications**

**The fields are as follows**

**Field Data type**

**title string**

**firstname string**

**lastname string**

**email string**

**completion Date**

**description string**

|  |  |  |
| --- | --- | --- |
| **Methods** | **Urls** | **Actions** |
| GET | api/qualifications | get all qualifications |
| GET | api/qualifications/:id | get qualifications by id |
| POST | api/qualifications | add new qualification |
| PUT | api/qualifications/:id | update qualification by id |
| DELETE | api/qualifications/:id | remove qualification by id |
| DELETE | api/qualifications | remove all qualifications |

**users**

**The fields are as follows**

**Field Data type**

**name string**

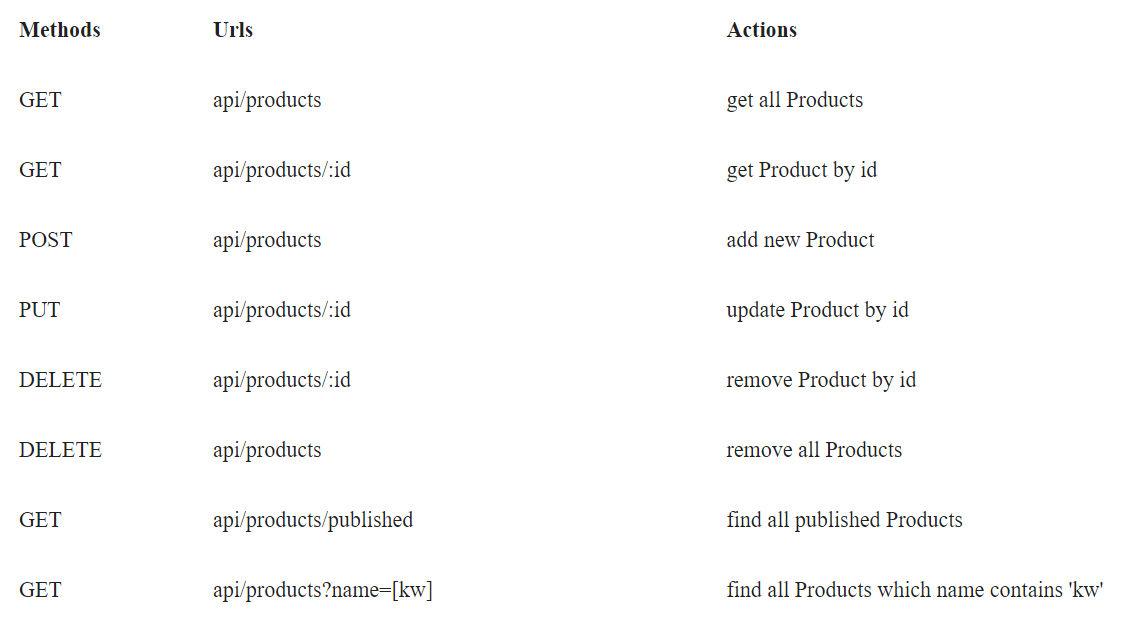
**email string**

**password string**

**created Date**

**updated Date**

|  |  |  |
| --- | --- | --- |
| **Methods** | **Urls** | **Actions** |
| GET | api/users | get all users |
| GET | api/users/:id | get users by id |
| POST | api/users | add new user |
| PUT | api/users/:id | update user by id |
| DELETE | api/users/:id | remove user by id |
| DELETE | api/users | remove all users |



1. a) Test the REST APIs above using Postman, Thunder client or any tool you are familiar with. e.t.c.

b) Provide the screen snapshot of the test. **(20 Marks)**

**PART IV – Authentication Backend code – Reference (Week6 slides)**

1. a) use week6 slide – Authentication.ppt to complete the backend authentication code
2. b) Create the route, corresponding controller function and authentication backend code for user to sign in and sign out using JWT token.

c) create protected routes. etc. **(10 Marks)**

**SUBMITTING YOUR WORK**

Your submission should include:

1. A zip archive of your Portfolio Project files. Do not use .rar
2. A link to GitHub
3. A word doc. Showing the snapshots

This assignment is weighted **10%** of your total mark for this course. Late submissions:20% deducted for each day late.