💻 Exam 4: Node, npm, and Express API

> This exam evaluates your ability to:
>
> - Create and manage a Node.js project
> - Use npm (including installing and using a package)
> - Create an Express server with multiple routes
> - Consume your API in a browser using Axios
> - Push a clean project to GitHub
>
> There are **10 tasks**, each worth **10 points**, for a total of 100 points.

📁 Project Folder Requirement

All work for this exam must be done in **one project folder**:
Name your folder: `exam4_lastname` (replace `lastname` with your actual last name).
At the end, you will:

- Push this folder to a GitHub repository
- Submit the **GitHub repository URL** in Canvas

🧩 Exam Tasks (10 points each)

1️⃣ Create the Node Project

Initialize a new Node.js project inside your **exam4_lastname** folder. Your package.json must reflect the correct project name.
✅ **Requirement:** A `package.json` file exists in your folder. The `name` field includes **exam4** and `author` includes your first and and your last name.

2️⃣ Task 2 (10 pts) – Install and Use `chalk`

In this task you will install **chalk** and use the documentation on **npmjs.com** to print a colored message.

1. Create a file named `chalkDemo.js`.
2. Use the documentation on **npmjs.com** for `chalk` to:
    - Implement chalk in your project
    - Print the text **Hello world** (any chalk style/color is fine)

✅ **Requirement:** Running `node chalkDemo.js` prints **Hello world** using chalk.

3️⃣ Add a Custom npm Script

Add a custom script to your package.json named **dev**. When run, it must execute the file you created in Task 2.
✅ **Requirement:** `npm run dev` from the terminal successfully runs `chalkDemo.js` and prints your chalk “Hello world” message.

4️⃣ Create a GIT repo

- Initialize the project as a git repo called **exam4**
- Add the file and code necessary to ensure the node_modules file doesn't get included

✅ Requirements: The repository contains package.json and your source files. The repository does not contain a node_modules folder.

5️⃣ Create an Express Server

Create an Express server in a file named **server.js**. Your server must run on port **3000** and, when started from the terminal, it must display: **"Server is running on http://localhost:3000"**
✅ **Requirement:** Running `node server.js`from the terminal displays the message above and the server is ready to receive requests at **http://localhost:3000**.

6️⃣ Add a String Route

Add a route to your Express server named **/course**. This route must return a simple string (“CIS 131 - Exam 4”).
Start your server and verify the route in the browser by visiting:
http://localhost:3000/course
✅ **Requirement:** Visiting **/course** in the browser displays the string response from your server.

7️⃣ Create an index.html That Consumes Your API

1. Write JavaScript in a `<script>` tag that:
    - Calls `http://localhost:3000/course`
    - Displays the course strong on the page.
2. Open **index.html** in the browser while your server is running and verify it shows the data from the API.

✅ **Requirement:** When you open **index.html**, it successfully fetches from `/course` and displays "CIS 131 - Exam 4".

8️⃣ Task 7 (10 pts) – Array of Objects Route (`/pets`)

In **server.js**, create a small array of objects and return it from a route.

1. Create an array in **server.js**, for example:

```
const pets = [
          { id: 1, type: "dog", name: "Diesel" },
          { id: 2, type: "cat", name: "Milo" },
          { id: 3, type: "bird", name: "Sky" }
        ];
```

6. Create a route:
   `GET /pets` that returns this array as JSON.
7. Verify in the browser at:
   `http://localhost:3000/pets`

✅ **Requirement:** The `/pets` route returns the array of objects as JSON.

9️⃣ Task 9 (10 pts) – Route Using a URL Parameter (`/pets/:id`)

In **server.js**, add a route that uses a URL parameter to return a single pet by its `id`.

1. Create a route:
   `GET /pets/:id`
2. Return that single pet as JSON.
3. Test in the browser with a URL like:
   `http://localhost:3000/pets/2`

✅ **Requirement:** Visiting `/pets/2` returns just the pet whose `id` is 2.

🔟 Task 10 (10 pts) – Route Using a Query String (`/pets/search`)

Add a new route to your Express server that uses a **query string** to filter your existing `pets` array.

1. Create a route with the path:
   `GET /pets/search`
2. The route must read a query string value (for example `type` or `name`) using `req.query`.
3. Use that query value to filter your `pets` array (for example, only return pets that match the requested type or name).
4. Return the **filtered results** as JSON.
5. Test in the browser with a URL like:
   `http://localhost:3000/pets/search?type=dog`
   or
   `http://localhost:3000/pets/search?name=Diesel`

✅ **Requirement:** Visiting `/pets/search` with a query string (such as `?type=dog` or `?name=Diesel`) returns a JSON array containing only the pets that match the filter.

📤 Submission (README.md Upload)

When you complete all 10 tasks, you will create a **README.md** file and push everything to GitHub.

Your **README.md** (stored inside your project folder) must include:

- Your **first and last name**
- The **URL to your public GitHub repository** that contains your `exam4_lastname` project
- A brief statement confirming that your server runs on `http://localhost:3000`
- Any additional notes you feel are relevant (optional)

After adding your README.md:

- Commit all work
- **Push your final code to GitHub**
- Ensure your GitHub repository is set to **public** so it can be accessed

✅ **Final Requirement:** Upload **only your README.md** file to the exam portal.
All code will be graded directly from your public GitHub repository.