**Name:  
Section & Year:  
Date:**

## **Objective:**

This project aims to reinforce the understanding of data validation, error handling, and ensuring unique data integrity by working with Node.js API endpoints. Students will learn to validate input, handle errors gracefully, and ensure that email addresses remain unique in a simple registration system.

### **Requirements:**

1. **Data Validation**: Learn to validate user inputs to ensure data integrity and prevent incorrect or harmful data from entering the system.
2. **Error Handling**: Implement proper error handling to manage invalid inputs or unexpected exceptions, ensuring the application doesn’t crash.
3. **Unique Email Addresses**: Introduce a mechanism to verify that each email address registered in the system is unique, maintaining consistent and reliable data.

## **Instructions:**

### **1. Fork and Clone the Repository**

* Navigate to the GitHub repository: [LabActivity03](https://github.com/philliplumod/LabActivity03).
* Click the **Fork** button at the top right corner to create a copy of the repository in your GitHub account.

Open your terminal and run the following command to clone the repository:  
  
git clone https://github.com/your-username/LabActivity03.git

### **2. Open the Project in VSCode**

### **3. Install Dependencies**

Install the necessary dependencies by running the following command in your terminal:  
npm install express body-parser

## **Lab Tasks:**

### **4. Implement Input Validation**

* Ensure the name, email, and password fields are properly validated (e.g., check for required fields, valid email format, and password length).
* Add validation logic in the POST /register route.
* For example, you can validate the email format using a regular expression or a validation library.

### **5. Handle Errors Gracefully**

* Add error handling to manage situations like missing fields or invalid data.
* Return appropriate error messages and status codes (e.g., 400 Bad Request for invalid input).
* Make sure your application doesn’t crash on encountering an exception.

### **6. Ensure Unique Emails**

* Modify the registration logic to check if the email already exists in the system before allowing registration.
* If the email is not unique, return a clear error message (e.g., "Email already exists") and a suitable HTTP status code (e.g., 409 Conflict).

## **Testing the API:**

### **7. Testing the POST /register Endpoint**

Use the following curl command to test the registration endpoint:  
  
curl -X POST http://localhost:3000/register \

-H "Content-Type: application/json" \

-d '{"name": "John Doe", "email": "john.doe@example.com", "password": "123456"}'

### **8. Testing the GET /users Endpoint**

Use the following curl command to test retrieving all users:  
  
curl http://localhost:3000/users

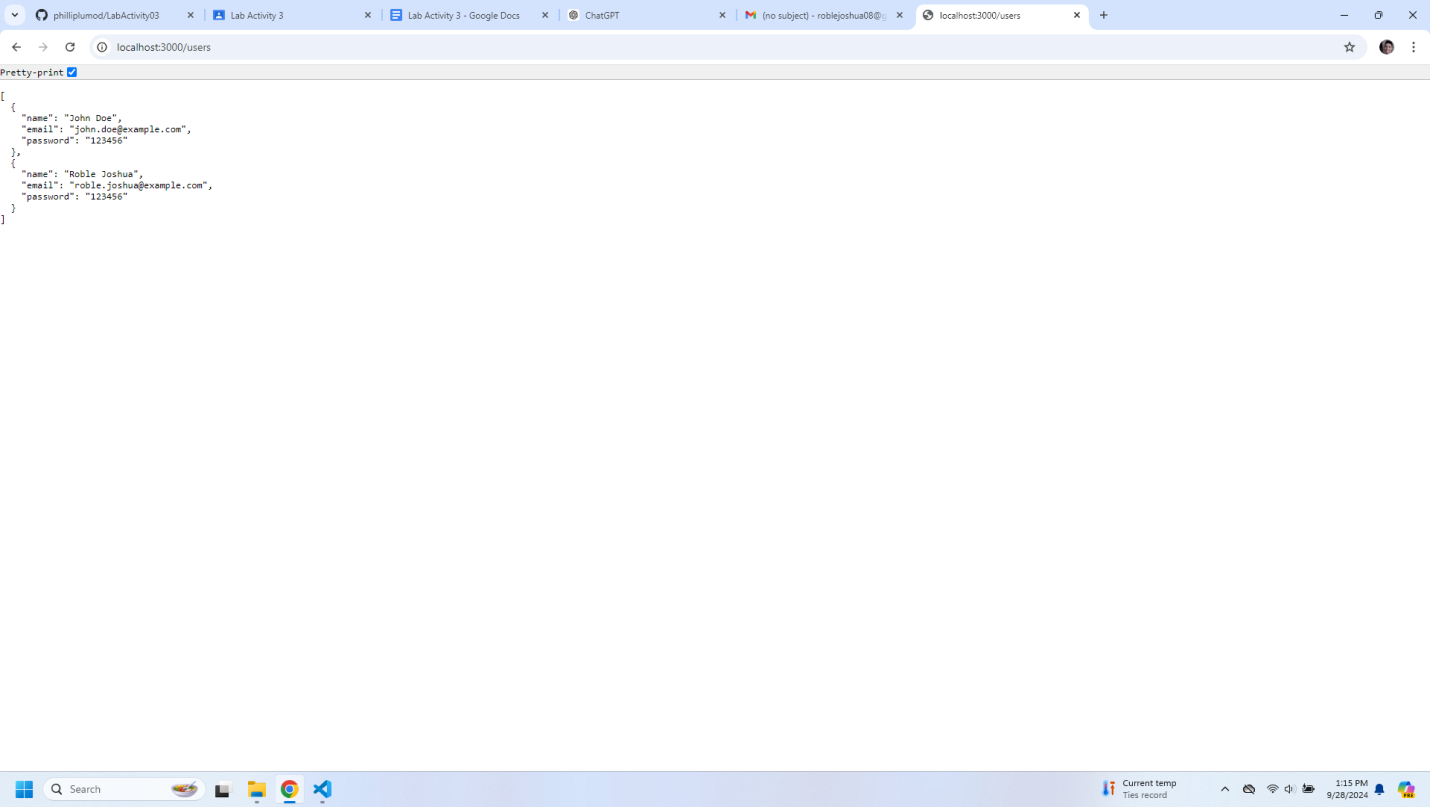
### **9. Document Each Step**

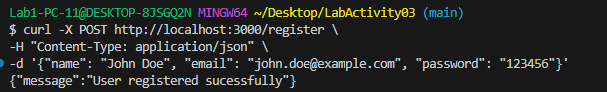
* Take a screenshot of every function implementation and output.

**10. Push Changes to GitHub**

Once you’ve completed the lab, push your changes back to your GitHub repository using the following commands:  
  
git add .

git commit -m "Completed LabActivity03"

git push origin main



Screenshot 2024-09-28 131648Screenshot 2024-09-28 123758Screenshot 2024-09-28 123644