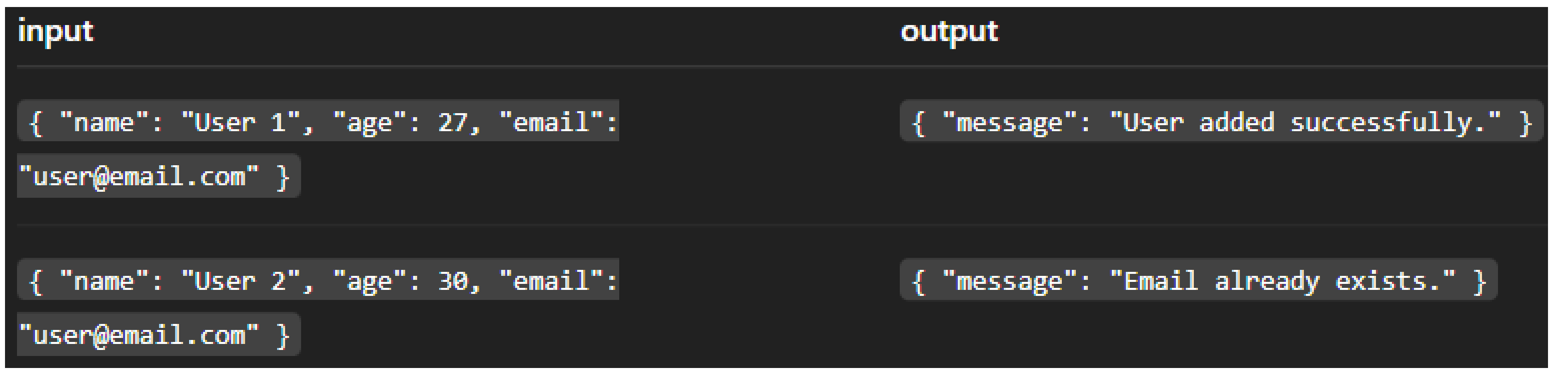
Assignment4 Part 1

Simple CRUD Operations Using Express.js:

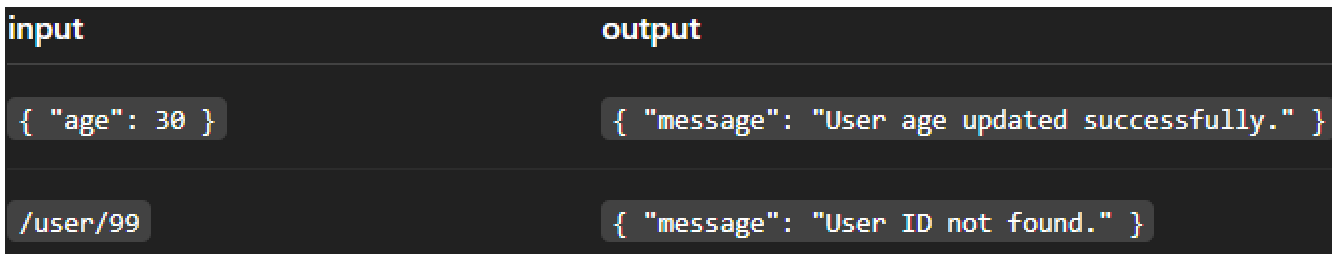
⚠️ **For all the following tasks, you must use the fs module to read and write data from a JSON file (e.g., users.json). Do not store or manage data using arrays. (2 Grades)**

1. Create an API that adds a new user to your users stored in a JSON file. **(ensure that the email of the new user doesn't exist before)(2 Grades)**
   - **URL:** POST /user



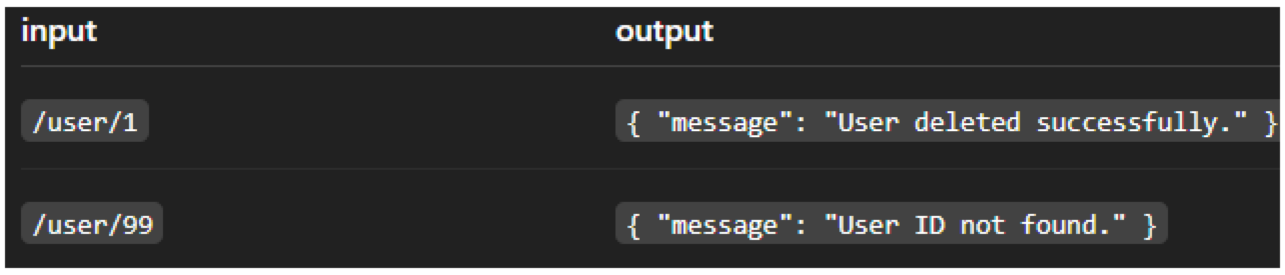
input	output
`{ "name": "User 1", "age": 27, "email": "user@email.com" }`	`{ "message": "User added successfully." }`
`{ "name": "User 2", "age": 30, "email": "user@email.com" }`	`{ "message": "Email already exists." }`

2. Create an API that updates an existing user's name, age, or email by their ID. The user ID should be retrieved from the **params**. **(1 Grade)**
   Note: **Remember to update the corresponding values in the JSON file**
   - **URL**: PATCH /user/:id



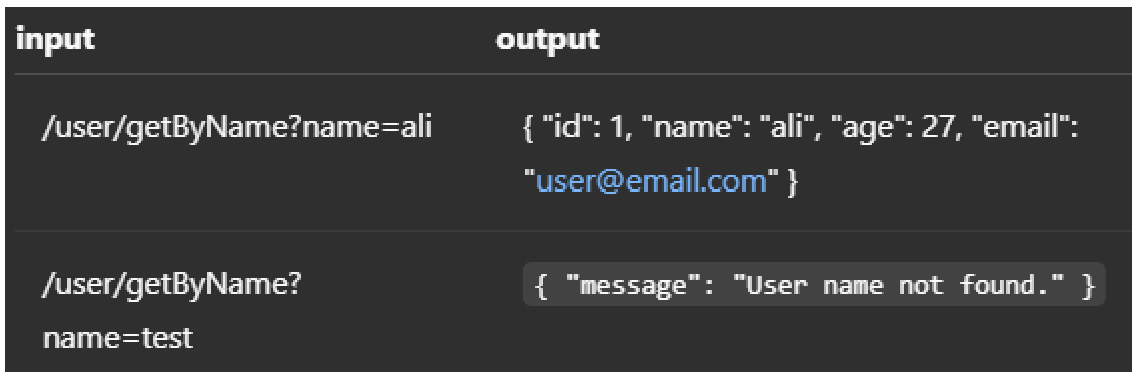
input	output
`{ "age": 30 }`	`{ "message": "User age updated successfully." }`
`/user/99`	`{ "message": "User ID not found." }`

3. Create an API that deletes a User by ID. The user id should be retrieved from either the **request body or optional params. (1 Grade)**
   Note: **Remember to delete the user from the file**
   - **URL:** DELETE /user{/:id}



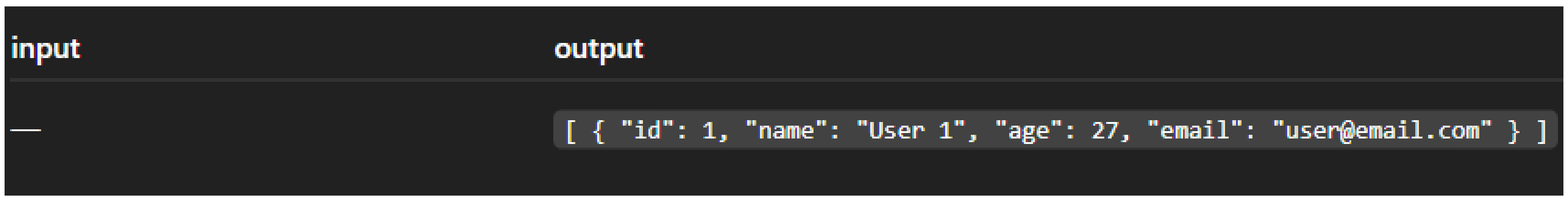
input	output
`/user/1`	`{ "message": "User deleted successfully." }`
`/user/99`	`{ "message": "User ID not found." }`

4. Create an API that gets a user by their name. The name will be provided as a **query parameter**. **(1 Grade)**
   - **URL:** GET /user/getByName



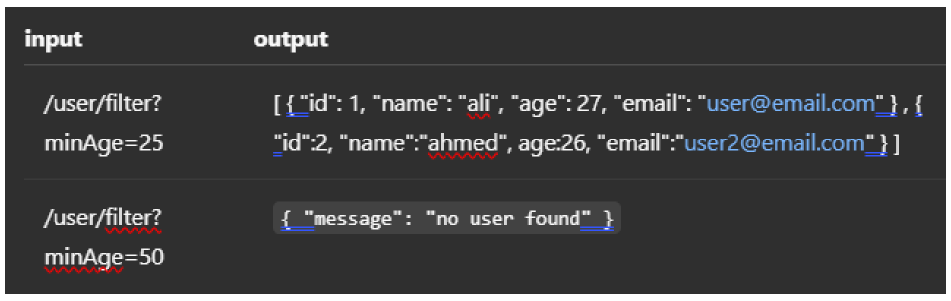
input	output
/user/getByName?name=ali	{ "id": 1, "name": "ali", "age": 27, "email": "user@email.com" }
/user/getByName?name=test	`{ "message": "User name not found." }`

5. Create an API that gets all users from the JSON file. **(1 Grade)**
   - **URL:** GET /user



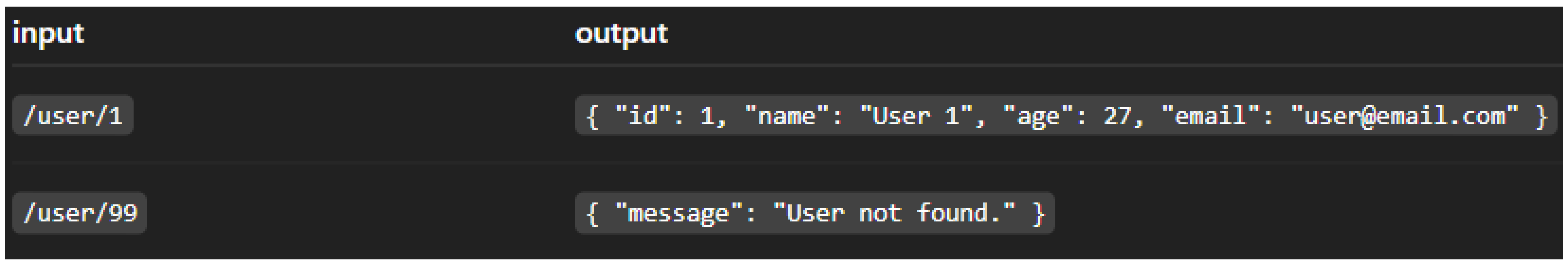
input	output
—	`[ { "id": 1, "name": "User 1", "age": 27, "email": "user@email.com" } ]`

6. Create an API that filters users by minimum age. **(1 Grade)**
   - **URL:** GET /user/filter



input	output
/user/filter?minAge=25	[{ "id": 1, "name": "ali", "age": 27, "email": "user@email.com" }, { "id":2, "name":"ahmed", age:26, "email":"user2@email.com" }]
/user/filter?minAge=50	`{ "message": "no user found" }`

7. Create an API that gets User by ID. **(1 Grade)**
   - **URL:** GET /user/:id
   - **Output:**



input	output
`/user/1`	`{ "id": 1, "name": "User 1", "age": 27, "email": "user@email.com" }`
`/user/99`	`{ "message": "User not found." }`

⚠️ Important Notes about postman

1. **Name the endpoint with a meaningful name like 'Add User', not dummy names.**
2. **Save your changes on each request(ctrl+s).**
3. **Include the Postman collection link (export your Postman collection) in the email with your assignment link**

Bonus (2 Grades)

How to deliver the bonus?

1- Solve the problem [Longest Common Prefix] on **LeetCode**

2- Inside your assignment folder, create a **SEPARATE FILE** and name it “bonus.js”

3- Copy the code that you have submitted on the website inside ”bonus.js” file