Assignment 3

Part1: Core Modules (1.5 Grades)

1. Use a readable stream to read a file in chunks and log each chunk. (0.5 Grade)

• Input Example: "./big.txt"
• Output Example: log each chunk

2. Use readable and writable streams to copy content from one file to another. (0.5 Grade)

• Input Example: "./source.txt", "./dest.txt"
• Output Example: File copied using streams

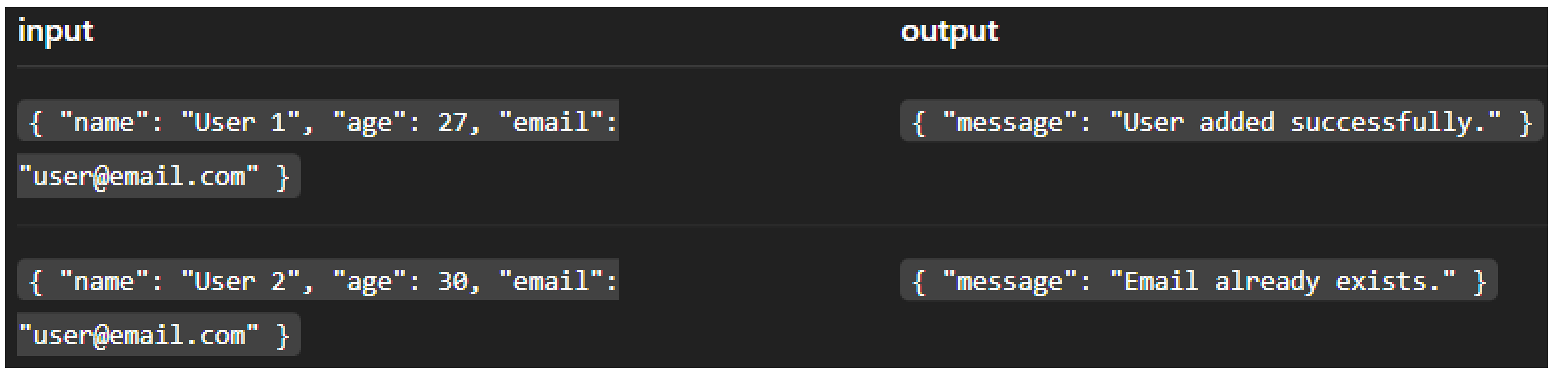
3. Create a pipeline that reads a file, compresses it, and writes it to another file. (0.5 Grade)

• Input Example: "./data.txt", "./data.txt.gz"

Part2: Simple CRUD Operations Using HTTP (5.5 Grades):

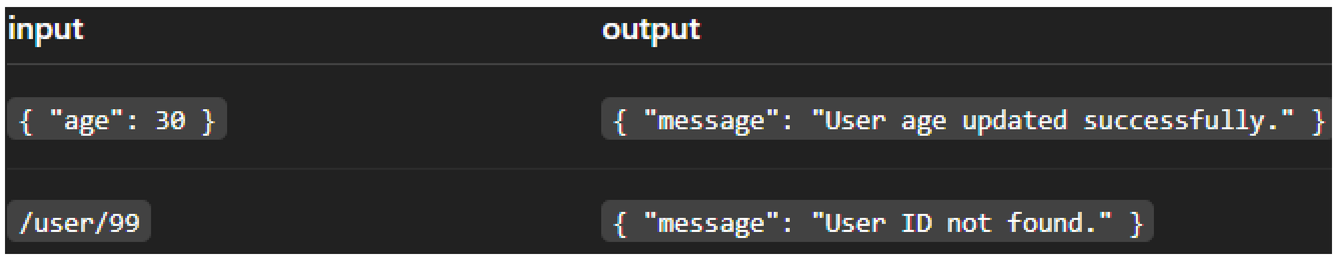
For all the following APIs, you must use the fs module to read and write data from a JSON file (e.g., users.json). Do not store or manage data using arrays (0.5 Grades).

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) (1 Grade)**
   - **URL:** POST /user



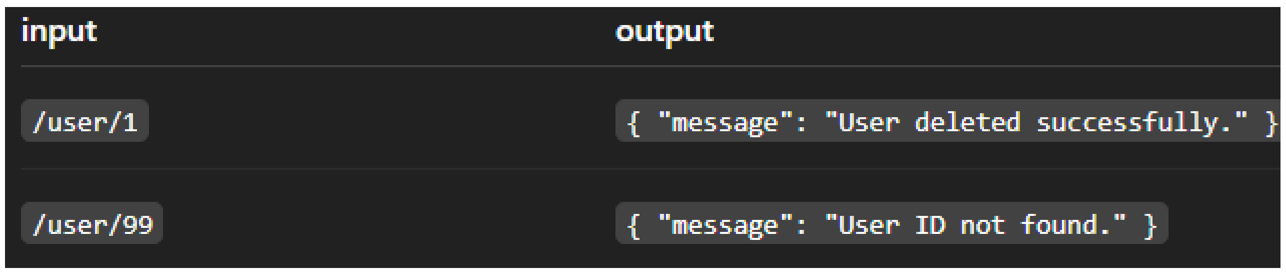
input	output
`{ "name": "User 1", "age": 27, "email": "user@email.com" }`	`{ "message": "User added successfully." }`
`{ "name": "User 2", "age": 30, "email": "user@email.com" }`	`{ "message": "Email already exists." }`

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 **URL (1 Grade)**
   ***Note:* Remember to update the corresponding values in the JSON file**
   - **URL**: PATCH /user/id



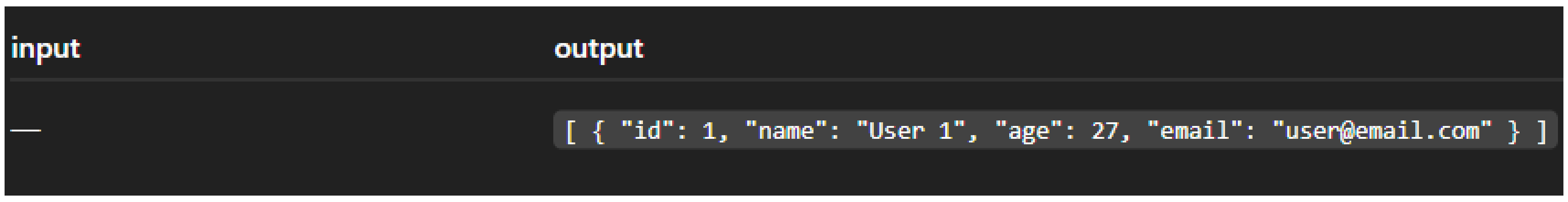
input	output
`{ "age": 30 }`	`{ "message": "User age updated successfully." }`
`/user/99`	`{ "message": "User ID not found." }`

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



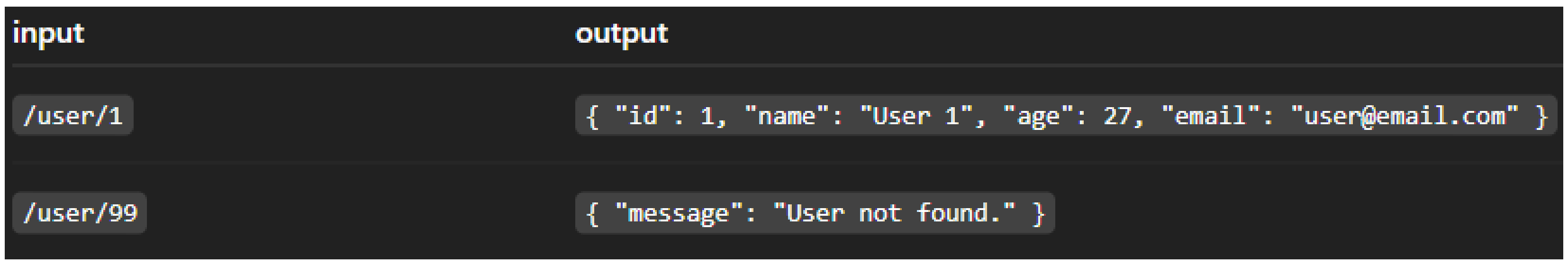
input	output
`/user/1`	`{ "message": "User deleted successfully." }`
`/user/99`	`{ "message": "User ID not found." }`

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



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

5. 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." }`

Assignment 3

Part3: Node Internals (3 Grades):

1. **What is the Node.js Event Loop? (0.5 Grade)**
2. **What is Libuv and What Role Does It Play in Node.js? (0.5 Grade)**
3. **How Does Node.js Handle Asynchronous Operations Under the Hood? (0.5 Grade)**
4. **What is the Difference Between the Call Stack, Event Queue, and Event Loop in Node.js? (0.5 Grade)**
5. **What is the Node.js Thread Pool and How to Set the Thread Pool Size? (0.5 Grade)**
6. **How Does Node.js Handle Blocking and Non-Blocking Code Execution? (0.5 Grade)**

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 Majority Element 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