

**Name: Patel Shivam S.**

**Enroll no: 21162101019**

**Sem: 5      Batch: 51      Branch: CBA**

**Sub: Microservices**

## **Practical 5**

**AIM:** www.abc.com website owner wants to manage user's records in JSON files. Save visitor details like name, password, id, occupation, etc.

Admin wants to perform the following task on that file:

1. list out all users.
  2. Add a new user with said detail
  3. Create Rest API to work with JSON
- covering the following endpoints:

**/user/add-** to add a new user(Check if any data is missing in the request or the user that you are trying to add already exists).

**/user/list-** To get the list of all the existing users in the file

**/user/update/:username-** To update the user's data by finding the user using the name and do it through the patch method. (Check if the user exists or not).

**/user/delete/:username-** delete the user with the help of username(Check if the user exists or not).

**GITHUB LINK:**

**[https://github.com/Shivam3783/microservice\\_practicals/tree/main/prac5](https://github.com/Shivam3783/microservice_practicals/tree/main/prac5)**

## CODE: 1)

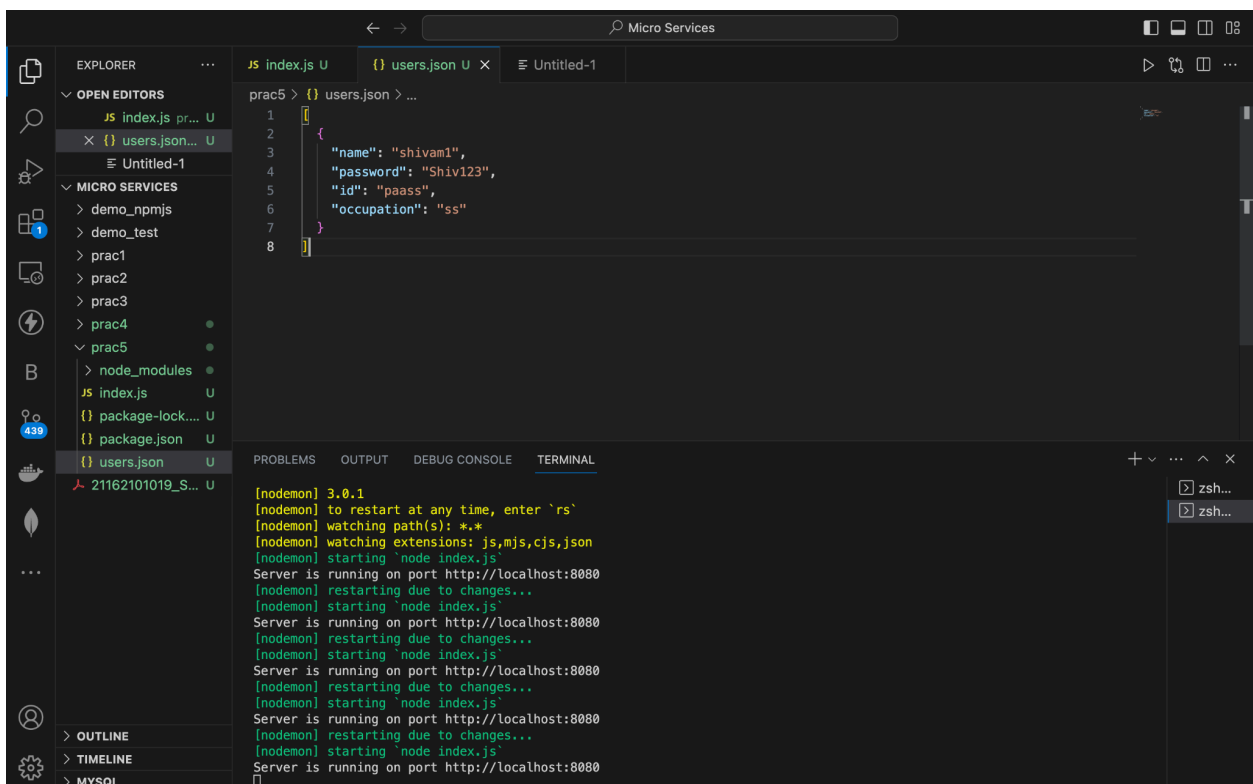
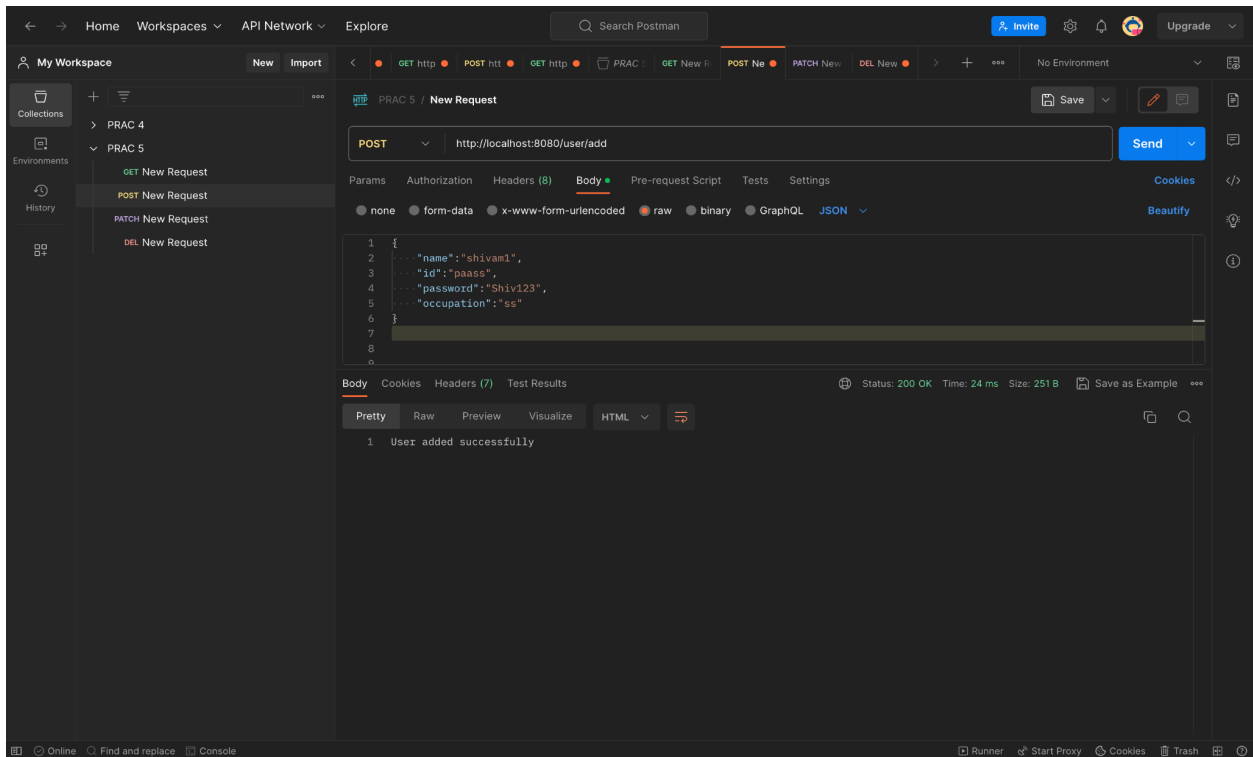
The screenshot shows the VS Code editor with the 'index.js' file open. The code implements a POST endpoint for adding a user. The left sidebar shows the Explorer view with the project structure, including 'demo\_test', 'prac1', 'prac2', 'prac3', 'prac4', 'prac5', 'node\_modules', 'package-lock.json', 'package.json', 'users.json', and '21162101019\_S...'. The main editor area shows the following code:

```
pracs > JS index.js > app.post('/user/add') callback
1  const express = require('express');
2  const fs = require('fs');
3
4  const app = express();
5  app.use(express.json());
6  var port = 8080;
7
8  let userData = require('./users.json');
9
10 app.post('/user/add', (req, res) => {
11   const { name, password, id, occupation } = req.body;
12
13   if(!name || !password || !id || !occupation) {
14     return res.status(400).send('Missing required fields');
15   }
16
17   if(userData.find(u => u.name === name)) {
18     return res.status(409).send('User already exists');
19   }
20
21   userData.push({
22     name,
23     password,
24     id,
25     occupation
26   });
27
28   fs.writeFileSync('./users.json', JSON.stringify(userData));
29
30   res.send('User added successfully');
31 });
32
33 app.get('/user/list', (req, res) => {
34   res.send(userData);
35 });
36
37 app.patch('/user/update/:name', (req, res) => {
```

The screenshot shows the VS Code editor with the 'index.js' file open, displaying the implementation of the PATCH and DELETE endpoints. The left sidebar shows the Explorer view with the project structure, including 'demo\_test', 'prac1', 'prac2', 'prac3', 'prac4', 'prac5', 'node\_modules', 'package-lock.json', 'package.json', 'users.json', and '21162101019\_S...'. The main editor area shows the following code:

```
pracs > JS index.js > app.post('/user/add') callback
36
37 app.patch('/user/update/:name', (req, res) => {
38
39   const name = req.params.name;
40   const update = req.body;
41
42   const user = userData.find(u => u.name === name);
43
44   if (!user) {
45     return res.status(404).send('User not found');
46   }
47
48   Object.assign(user, update);
49
50   fs.writeFileSync('./users.json', JSON.stringify(userData));
51
52   res.send('User updated successfully');
53 });
54
55 app.delete('/user/delete/:name', (req, res) => {
56   const name = req.params.name;
57
58   const index = userData.findIndex(u => u.name === name);
59
60   if(index === -1) {
61     return res.status(404).send('User not found');
62   }
63
64   userData.splice(index, 1);
65
66   fs.writeFileSync('./users.json', JSON.stringify(userData));
67
68   res.send('User deleted successfully');
69 });
70
71 app.listen(port, () => {
```

## OUTPUT in POSTMAN: → Add data (POST REQUEST)



## → GET data (GET REQUEST)

The screenshot displays the Postman application interface. On the left sidebar, the 'My Workspace' section is active, showing a collection named 'PRAC 5' with a 'New Request' button. The main panel shows a 'New Request' tab for a GET request to the URL 'http://localhost:8080/user/list'. The 'Send' button is visible. Below the URL bar, the 'Params' tab is selected, showing a table for 'Query Params' with columns 'Key', 'Value', and 'Description'. The 'Body' tab is also visible, showing a JSON response in 'Pretty' format. The status bar at the bottom indicates a successful response with status '200 OK', time '18 ms', and size '307 B'.

**GET Request Details:**

- Method: GET
- URL: http://localhost:8080/user/list
- Status: 200 OK
- Time: 18 ms
- Size: 307 B

**Response Body (JSON):**

```
{
  "name": "shivam1",
  "password": "Shiv123",
  "id": "paass",
  "occupation": "ss"
}
```

## → Update data (PATCH REQUEST)

The screenshot displays the Postman interface with a PATCH request configured and executed. The request is sent to the URL `http://localhost:8080/user/update/shivam1`. The body of the request is a JSON object: `{ "name": "Shivam_UPDATE", "id": "hello_NEW" }`. The response status is `200 OK` with a time of `31 ms` and a size of `253 B`. The response body, shown in the 'Pretty' view, is `1 User updated successfully`.

**Request Details:**

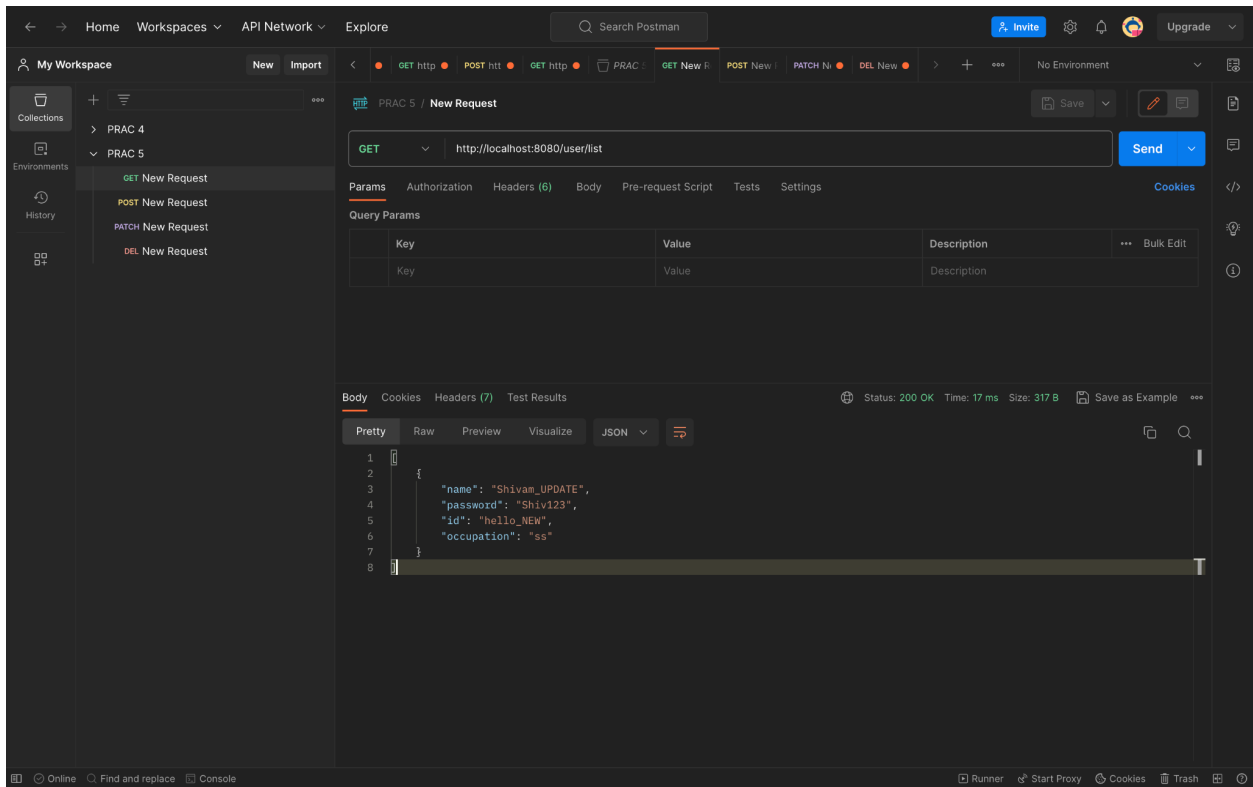
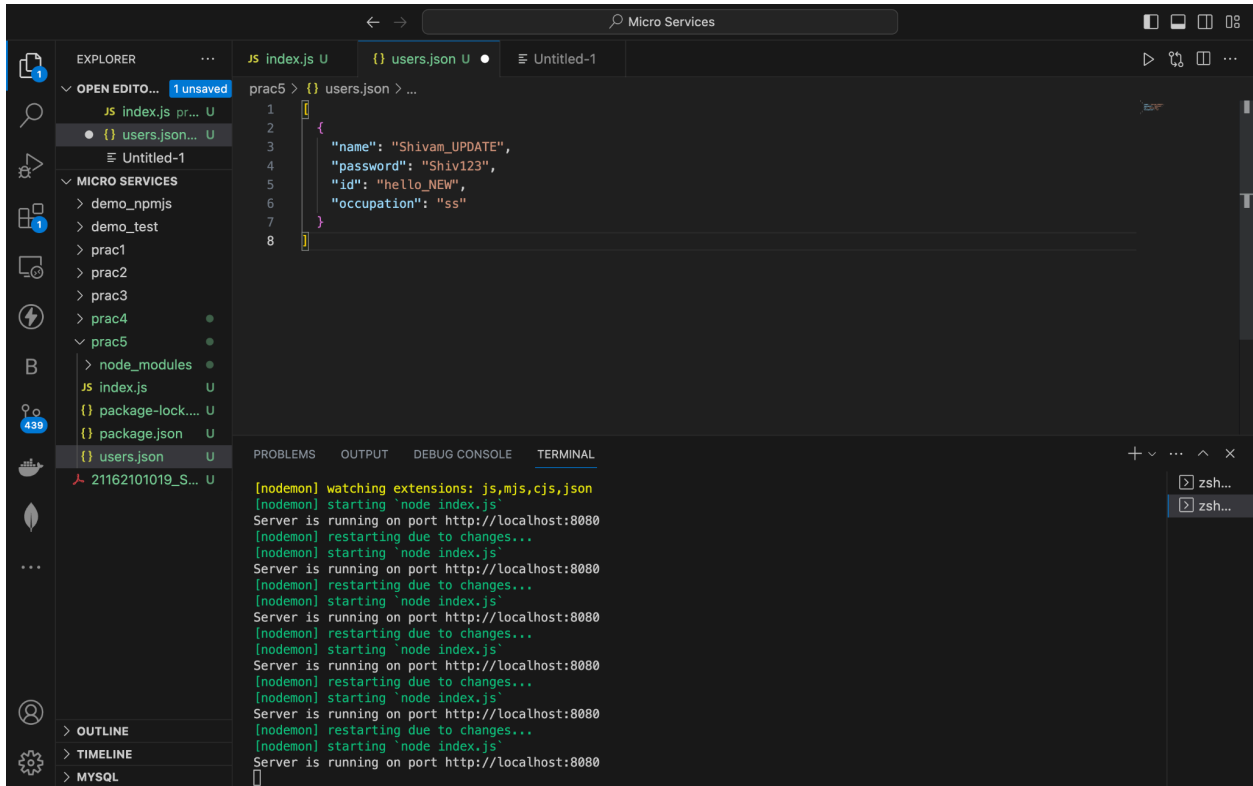
- Method: PATCH
- URL: `http://localhost:8080/user/update/shivam1`
- Body Type: raw (JSON)
- Body Content:

```
1 {
2   "name": "Shivam_UPDATE",
3   "id": "hello_NEW"
4 }
5
```

**Response Details:**

- Status: 200 OK
- Time: 31 ms
- Size: 253 B
- Body (Pretty):

```
1 User updated successfully
```



## → Delete data (DELETE REQUEST)

The screenshot displays the Postman interface with a new DELETE request configured. The request is set to the URL `http://localhost:8080/user/delete/Shivan_UPDATE`. The response status is `200 OK` with a message `User deleted successfully`.

**Request Details:**

- Method: DELETE
- URL: `http://localhost:8080/user/delete/Shivan_UPDATE`
- Params: Empty table

Key	Value	Description
Key	Value	Description

**Response Details:**

- Status: 200 OK
- Time: 6 ms
- Size: 253 B
- Body: `1 User deleted successfully`

