

**Name - Yash Lakhtariya**  
**Enrollment number - 21162101012**  
**Branch - CBA      Batch - 51**  
**Microservices Practical 5**

**Scenario**: 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).

**Code**:

```
const express = require("express");
const app = express();
const fs = require("fs").promises;
const bodyParser = require("body-parser");
app.use(bodyParser.json());

const jsonfile = "./user.json";
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 51  
Microservices Practical 5

```
async function read() {  
  try {  
    const data = await fs.readFile(jsonfile, "utf8");  
    return JSON.parse(data);  
  } catch (error) {  
    throw error;  
  }  
}  
  
async function write(users) {  
  try {  
    await fs.writeFile(jsonfile, JSON.stringify(users, null, 2));  
  } catch (error) {  
    throw error;  
  }  
}  
  
app.get("/user", async (req, res) => {  
  res.send("Welcome to YSL User Database!");  
});  
  
app.get("/user/list", async (req, res) => {  
  try {  
    const users = await read();
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 51  
Microservices Practical 5

```
res.json(users);
} catch (error) {
res.status(500).json({ error: "Internal server error" });
}
});

app.get("/user/:username", async (req, res) => {
const username = req.params.username;
try {
const users = await read();
const user = users.find((user) => user.username === username);
if (user) {
res.json(user);
} else {
res.status(404).json({ error: "User not found" });
}
} catch (error) {
res.status(500).json({ error: "Internal server error" });
}
});

app.post("/user/add", async (req, res) => {
const newUser = req.body;
if (
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 51  
Microservices Practical 5

```
!newUser.username ||
!newUser.id ||
!newUser.password ||
!newUser.occupation
) {
return res.status(400).json({ error: "Missing required fields" });
}
try {
const users = await read();
if (users.some((user) => user.username === newUser.username)) {
return res.status(409).json({ error: "Username already exists" });
}
users.push(newUser);
await write(users);
res.json({ message: "User added successfully" });
} catch (error) {
res.status(500).json({ error: "Internal server error" });
}
});

app.put("/user/update/:username", async (req, res) => {
const username = req.params.username;
const updatedFields = req.body;
try {
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 51  
Microservices Practical 5

```
const users = await read();

const index = users.findIndex((user) => user.username === username);

if (index !== -1) {
  users[index] = { ...users[index], ...updatedFields };
  await write(users);
  res.json({ message: "User updated successfully" });
} else {
  res.status(404).json({ error: "User not found" });
}
} catch (error) {
  res.status(500).json({ error: "Internal server error" });
}
});

app.delete("/user/delete/:username", async (req, res) => {
  const username = req.params.username;

  try {
    const users = await read();

    const index = users.findIndex((user) => user.username === username);

    if (index !== -1) {
      users.splice(index, 1);
      await write(users);
      res.json({ message: "User deleted successfully" });
    } else {
```

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA      Batch - 51

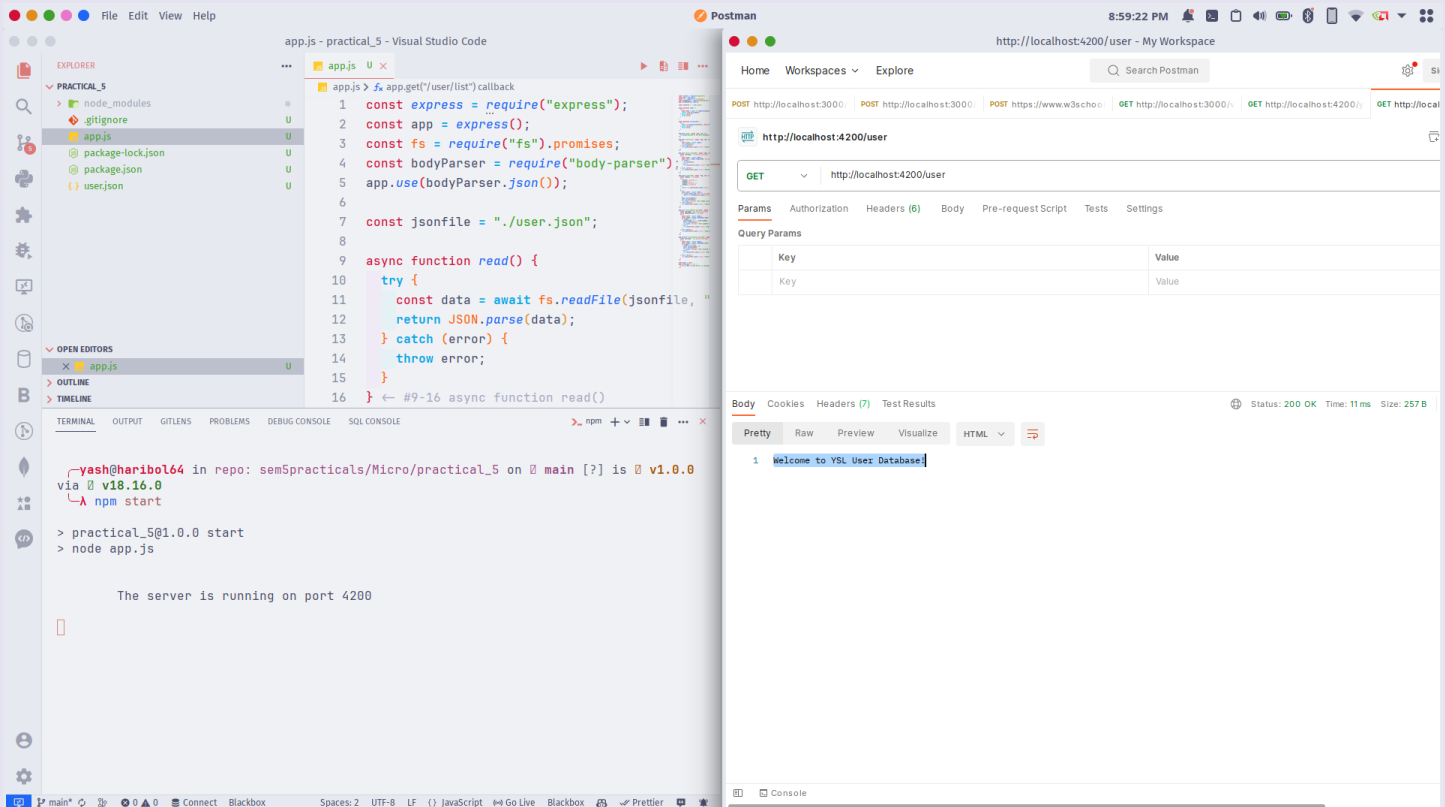
Microservices Practical 5

```
res.status(404).json({ error: "User not found" });  
  
}  
} catch (error) {  
res.status(500).json({ error: "Internal server error" });  
}  
});  
  
const port = 4200;  
app.listen(port, () => {  
console.log(`\n\tThe server is running on port ${port}\n`);  
});
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 51  
Microservices Practical 5

## Screenshots :

- Homepage of local server



Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 51  
Microservices Practical 5

- List users

The screenshot displays the Postman application interface. The top bar shows the time as 8:59:46 PM and includes standard system icons. The main interface is divided into three sections: History, Request Editor, and Response Viewer.

**History:** A list of recent requests is shown on the left. The most recent request is a GET request to `http://localhost:4200/user/list`, which is highlighted. Other requests include POST requests to `http://localhost:3000/login` and `https://www.w3schools.com/xml/temperconvert.a`.

**Request Editor:** The selected request is a GET request to `http://localhost:4200/user/list`. The interface shows tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Body tab is active, showing the request body in JSON format.

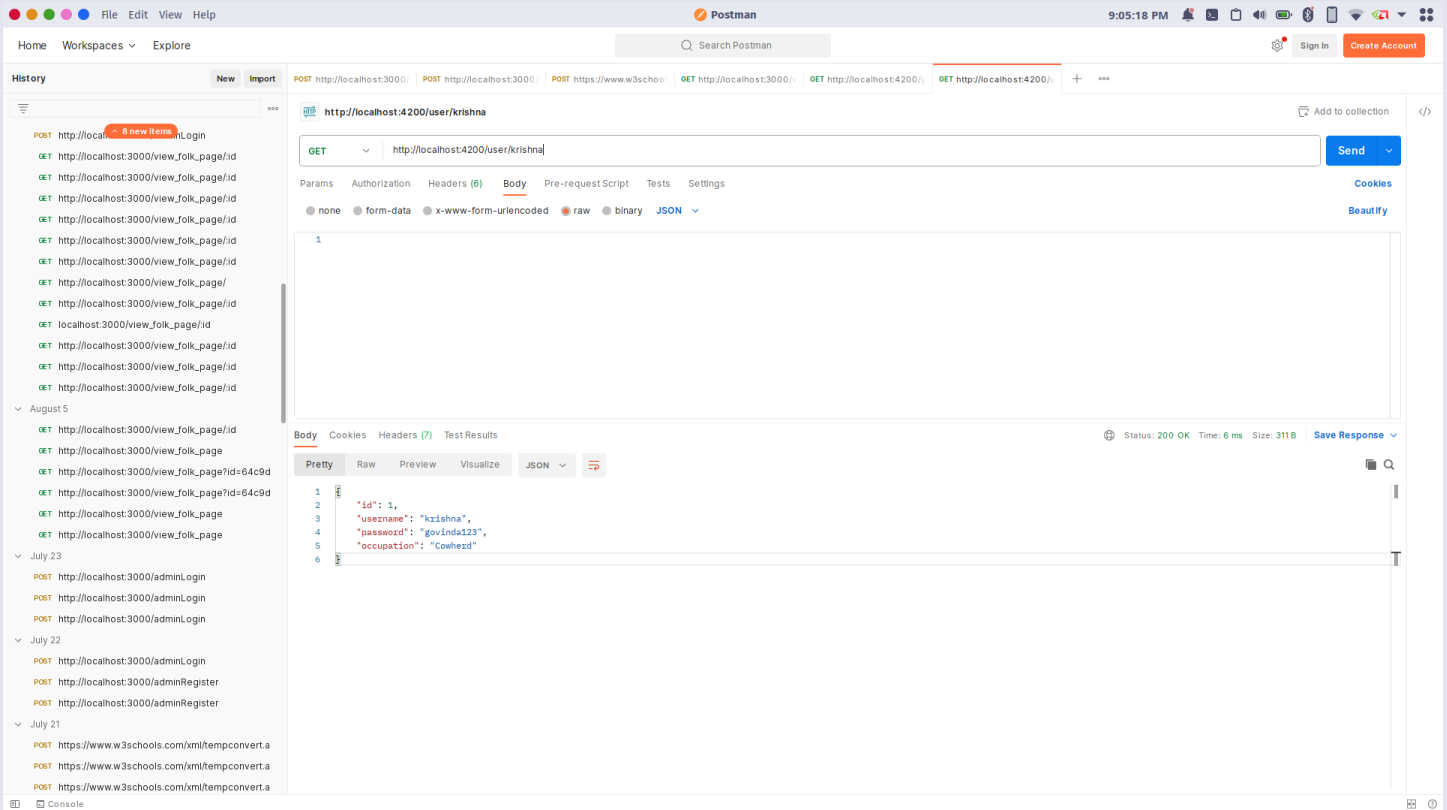
**Response Viewer:** The response is displayed in the right pane. It shows a status of 200 OK, a time of 4 ms, and a size of 1.08 KB. The response body is a JSON array of user objects, including details like id, username, password, and occupation.

```
1 {
2   {
3     "id": 1,
4     "username": "krishna",
5     "password": "govinda123",
6     "occupation": "Cowherd"
7   },
8   {
9     "id": 2,
10    "username": "madhava",
11    "password": "radhamadhav",
12    "occupation": "Flute Player"
13  },
14  {
15    "id": 3,
16    "username": "parthasarathi",
17    "password": "kuntiputra",
18    "occupation": "Charioteer of Arjuna"
19  },
20  {
21    "id": 4,
22    "username": "govinda",
23    "password": "hazibol",
24    "occupation": "Pleasure Giver"
25  },
26  {
27    "id": 5,
28    "username": "muralidhara",
29    "password": "vrajaasa",
30    "occupation": "Bearer of the Flute"
31  },
32  {
33    "id": 6,
34    "username": "Narsinha",
35    "password": "naxahzil",
36    "occupation": "Supreme Protector"
37  },
38  {
39    "id": 7,
```



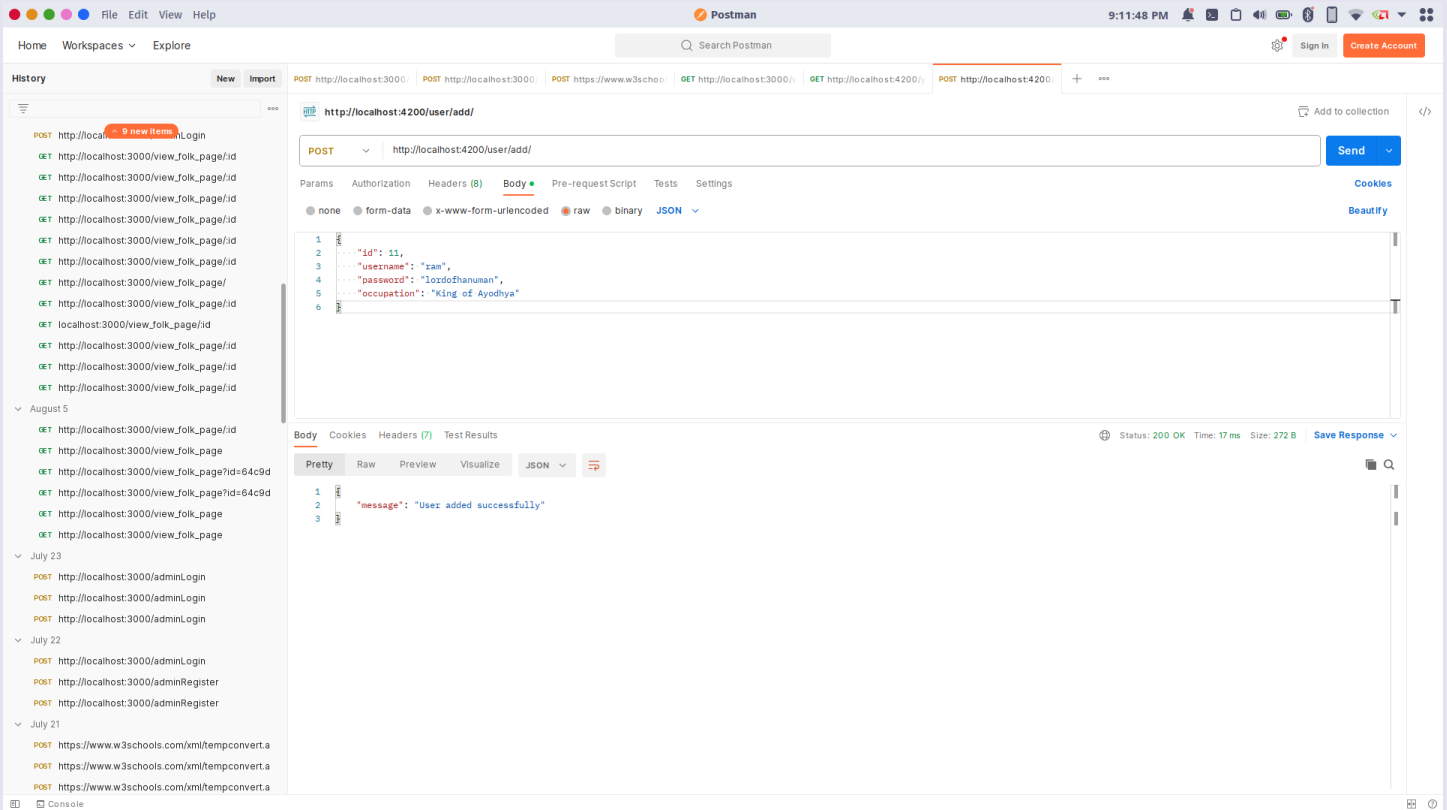
Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA Batch - 51  
Microservices Practical 5

- GET users by its username



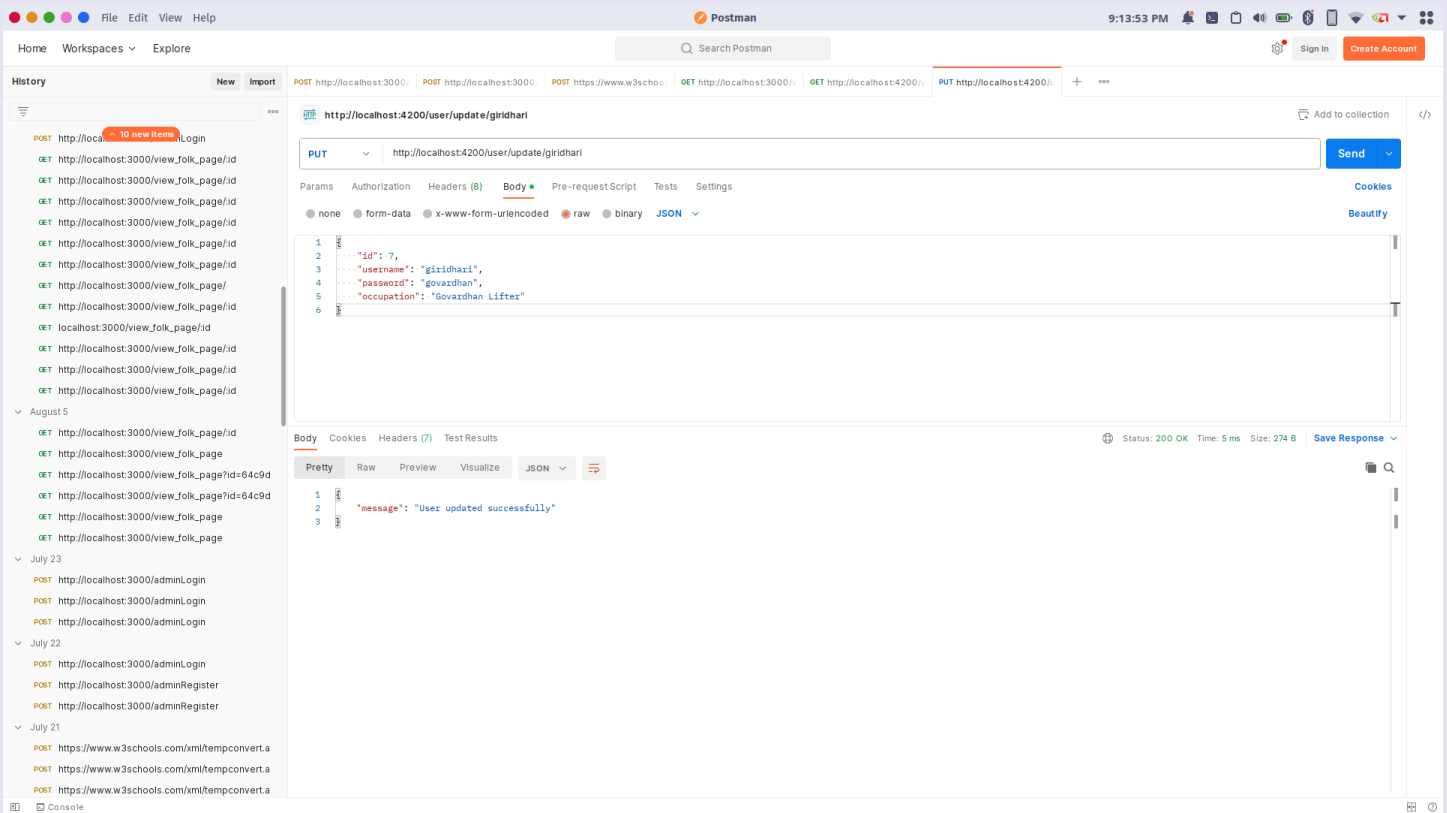
Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 51  
Microservices Practical 5

- Add user by POST



Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA Batch - 51  
Microservices Practical 5

- Update user by username



Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA Batch - 51  
Microservices Practical 5

- Delete user by username

