Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology

Name: Tushar Panchal

En.No: 21162101014

Sub: MICROSEVICES

Branch: CBA

Batch:51

----------PRACTICAL 05----------

❖ Question (TASK) :

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/Tushar007079/MICROSERVICES_PRACTICALS/tree/1f5b899ddf838393d5e24bcf46c1c989fe1f044d/5

❖ **STEPS TO PERFORM THIS TASK :**

👉 Step 1 :

» Create Project Files.

» Create a new project directory and set up the following files inside it :

✓ **Index.html :-**

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Management</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <h1 class="animate-charcter">User Management</h1>

    <!-- Form to add a new user -->
    <h2>Add User</h2>
    <form id="addUserForm">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br>
        <label for="age">Age:</label>
        <input type="number" id="age" name="age" required><br>
        <label for="languages">Languages (comma-separated):</label>
        <input type="text" id="languages" name="languages" required><br>
        <button type="submit">Add User</button>
    </form>

    <!-- Add a button to trigger the "List all users" function -->
    <button id="listUsersButton">List All Users</button>

    <!-- Table to display users -->
    <h2>List of Users</h2>
    <div class="tbl-header">
        <table cellpadding="0" cellspacing="0" id="userTable" border="1"
style="display: none;">
            <thead>
```

```
                <tr>
                    <th>Name</th>
                    <th>Age</th>
                    <th>Languages</th>
                    <th>Actions</th>
                </tr>
            </thead>
    </div>
    <table id="userTable" border="1" style="display: none;">
        <tr>
            <th>Name</th>
            <th>Age</th>
            <th>Languages</th>
            <th>Actions</th>
        </tr>
    </table>

    <!-- Form to update a user -->
    <h2>Update User</h2>
    <form id="updateUserForm">
        <label for="updateName">Username to Update:</label>
        <input type="text" id="updateName" name="updateName" required><br>
        <label for="updateAge">New Age:</label>
        <input type="number" id="updateAge" name="updateAge" required><br>
        <button type="submit">Update User</button>
    </form>

    <!-- Form to delete a user -->
    <h2>Delete User</h2>
    <form id="deleteUserForm">
        <label for="deleteName">Username to Delete:</label>
        <input type="text" id="deleteName" name="deleteName" required><br>
        <button type="submit">Delete User</button>
    </form>

    <!-- Latest update information -->
    <div id="updateInfo"></div>

    <script>
        // Function to fetch and display the list of users
        function listUsers() {
            fetch('/user/list')
                .then(response => response.json())
                .then(data => {
                    const userTable = document.getElementById('userTable');
                    userTable.innerHTML = `
          <tr>
            <th>Name</th>
            <th>Age</th>
            <th>Languages</th>
            <th>Actions</th>
          </tr>
        `;
                    data.users.forEach(user => {
                        const row = document.createElement('tr');
                        // Check if user.language is defined before joining
                        const languages = user.language ? user.language.join(', ')
: '';
```

```
                        row.innerHTML = `
            <td>${user.name}</td>
            <td>${user.age}</td>
            <td>${languages}</td>
            <td>
              <button onclick="updateUser('${user.name}', ${user.age})"
class=>Update</button>
              <button onclick="deleteUser('${user.name}')">Delete</button>
            </td>
          `;
                        userTable.appendChild(row);
                    });
                })
                .catch(error => {
                    console.error('Error fetching user list:', error);
                });
        }

        // Function to show/hide the user table
        function toggleUserTable() {
            const userTable = document.getElementById('userTable');
            const isVisible = userTable.style.display !== 'none';
            userTable.style.display = isVisible ? 'none' : 'block';
        }

        // Add event listener to the "List All Users" button
        document.getElementById('listUsersButton').addEventListener('click',
function () {
            listUsers();
            toggleUserTable();
        });

        // Add event listener to the "List All Users" button
        document.getElementById('listUsersButton').addEventListener('click',
listUsers);

        // Function to add a new user
        function addUser(event) {
            event.preventDefault();
            const form = event.target;
            const formData = new FormData(form);
            const userData = {
                name: formData.get('name'),
                age: Number(formData.get('age')),
                language: formData.get('languages').split(',').map(lang =>
lang.trim()),
            };
            fetch('/user/add', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify(userData),
            })
                .then(response => {
                    if (response.status === 201) {
                        form.reset();
                        listUsers();
```

```
                } else {
                    return response.json();
                }
            })
            .then(data => {
                if (data.error) {
                    alert(data.error);
                }
            });
    }

    // Function to update a user
    function updateUser(username, age) {
        const updateNameInput = document.getElementById('updateName');
        const updateAgeInput = document.getElementById('updateAge');
        updateNameInput.value = username;
        updateAgeInput.value = age;
    }

    // Function to handle user update form submission
    function handleUpdate(event) {
        event.preventDefault();
        const form = event.target;
        const formData = new FormData(form);
        const usernameToUpdate = formData.get('updateName');
        const newAge = Number(formData.get('updateAge'));
        const updatedUser = { age: newAge };
        fetch(`/user/update/${usernameToUpdate}`, {
            method: 'PATCH',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(updatedUser),
        })
            .then(response => {
                if (response.status === 200) {
                    form.reset();
                    listUsers();
                } else {
                    return response.json();
                }
            })
            .then(data => {
                if (data.error) {
                    alert(data.error);
                }
            });
    }

    // Function to delete a user
    function deleteUser(username) {
        if (confirm(`Are you sure you want to delete the user ${username}?`)) {
            fetch(`/user/delete/${username}`, {
                method: 'DELETE',
            })
                .then(response => {
                    if (response.status === 204) {
                        listUsers();
```

```
                        } else {
                            return response.json();
                        }
                    })
                    .then(data => {
                        if (data.error) {
                            alert(data.error);
                        }
                    });
            }
        }

        // Fetch the initial list of users
        listUsers();

        // Add event listeners
        document.getElementById('addUserForm').addEventListener('submit', addUser);
        document.getElementById('updateUserForm').addEventListener('submit',
handleUpdate);
        document.getElementById('deleteUserForm').addEventListener('submit',
function (event) {
            event.preventDefault();
            const form = event.target;
            const formData = new FormData(form);
            const usernameToDelete = formData.get('deleteName');
            deleteUser(usernameToDelete);
            form.reset();
        });
        // Function to display the latest update information
        function displayUpdateInfo(updateType, username) {
            const updateInfoDiv = document.getElementById('updateInfo');
            updateInfoDiv.textContent = `Latest ${updateType}: User "${username}"`;
        }
    </script>
</body>

</html>
```

✓ app.js :-

```
const express = require('express');
const bodyParser = require('body-parser');
const fs = require('fs');
const path = require('path');
const http = require('http');

const app = express();
const PORT = 3000;
const cors = require('cors');
app.use(cors());

app.use(bodyParser.json());
app.use(express.static('public'));

// File to store user data
const usersFile = './users.json';

// Read existing user data from JSON file
```

```
const getUsers = () => {
  try {
    const data = fs.readFileSync(usersFile);
    return JSON.parse(data);
  } catch (error) {
    return [];
  }
};

// Write user data to JSON file
const saveUsers = (users) => {
  fs.writeFileSync(usersFile, JSON.stringify(users, null, 2));
};

// Add a variable to track the latest update
let latestUpdate = {
  type: '',
  username: '',
};

// Endpoint to add a new user
app.post('/user/add', (req, res) => {
  const users = getUsers();
  const newUser = req.body;
  const existingUser = users.find((user) => user.name === newUser.name);

  if (existingUser) {
    return res.status(400).json({ error: 'User already exists' });
  }

  users.push(newUser);
  saveUsers(users);

  latestUpdate = {
    type: 'add',
    username: newUser.name,
  };

  res.status(201).json(newUser);
});

// Endpoint to list all users
app.get('/user/list', (req, res) => {
  const users = getUsers();
  res.status(200).json({ users, latestUpdate });
});

// Endpoint to update a user by username
app.patch('/user/update/:username', (req, res) => {
  const username = req.params.username;
  const users = getUsers();
  const updatedUser = req.body;
  const index = users.findIndex((user) => user.name === username);

  if (index === -1) {
    return res.status(404).json({ error: 'User not found' });
  }
```

```
  users[index] = { ...users[index], ...updatedUser };
  saveUsers(users);

  latestUpdate = {
    type: 'update',
    username: updatedUser.name,
  };

  res.status(200).json(users[index]);
});

// Endpoint to delete a user by username
app.delete('/user/delete/:username', (req, res) => {
  const username = req.params.username;
  const users = getUsers();
  const index = users.findIndex((user) => user.name === username);

  if (index === -1) {
    return res.status(404).json({ error: 'User not found' });
  }

  users.splice(index, 1);
  saveUsers(users);

  latestUpdate = {
    type: 'delete',
    username,
  };

  res.status(204).send();
});


// Endpoint to list all users
app.get('/user/list', (req, res) => {
    console.log('Received a request to /user/list');
    const users = getUsers();
    res.status(200).json({ users });
});


// Create an HTTP server
const server = http.createServer(app);

// Serve the HTML file
app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, 'index.html'));
});

server.listen(PORT, () => {
  console.log(`Server is running on port http://localhost:${PORT}`);
});
```

➡ Step 2 :

» Install Dependencies.

- Open a terminal or command prompt, navigate to the project directory, and install the required dependencies. In this case, we need **'express','body-parser','http','fs(file-system)','path','cors'** to run the server.
- Run this following command to initialize the package.json file : `npm init -y`
- Run this following command to install the EXPRESS Module : `npm install express body-parser fs path http cors`
- Open a terminal or command prompt, navigate to the project directory, and install the required dependencies. In this case, we need **'express','body-parser','http','fs(file-system)','path','cors'** to run the server.

1. Express :

- Express is a popular and minimalistic web application framework for Node.js.

- It simplifies the process of building web applications by providing a set of tools and utilities for routing, middleware, handling requests and responses, etc.

- In your code, you're using Express to create routes and handle HTTP requests and responses.

2. Body-Parser :

- Body-parser is a middleware module for Express that simplifies handling of HTTP request bodies.

- It allows you to parse data from incoming requests, especially data that's sent through forms or in the request body (e.g., JSON data).

- In your code, you're using body-parser to parse JSON data from request bodies.

3. fs (File System) :

- The `fs` module is a core module in Node.js that provides functionality for interacting with the file system on your computer.

- You're using it in your code to read and write user data to a JSON file.

4. path :

- The `path` module is another core module in Node.js that provides utilities for working with file and directory paths.

- You're using it in your code to construct file paths for serving static files and sending responses.

5. http :

- The `http` module is a core module in Node.js that provides the basic functionality to create an HTTP server and make HTTP requests.

- In your code, you're using it to create an HTTP server to serve your Express application.

6. cors :

- CORS stands for Cross-Origin Resource Sharing.

- It's a security feature implemented by web browsers to control requests made across different origins (domains).

- The `cors` package provides Express middleware that enables you to handle CORS-related issues when making requests from different origins.

Step 3 :

- Run the server.
- In the Terminal/CMD , run this following command to start the NodeJS Server:
  `node app.js`
- The server will running on port http://localhost:3000.

Step 4 :

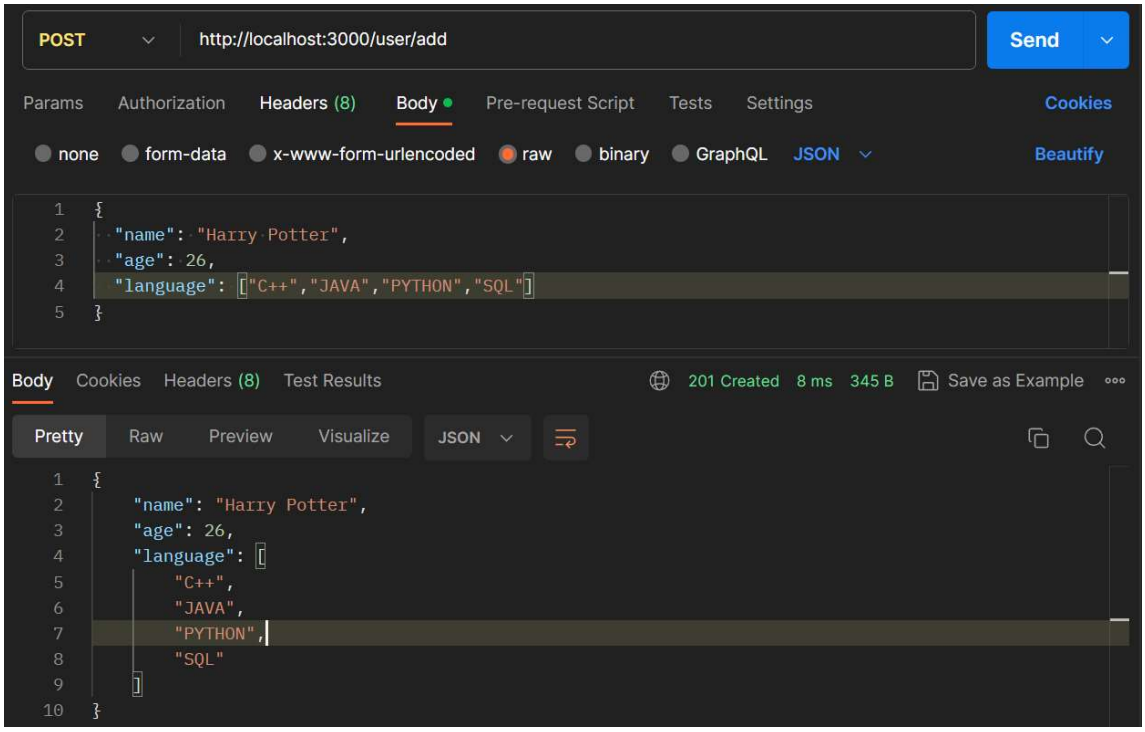
- Access the application
- Open web browser and enter the following address:

http://localhost:3000

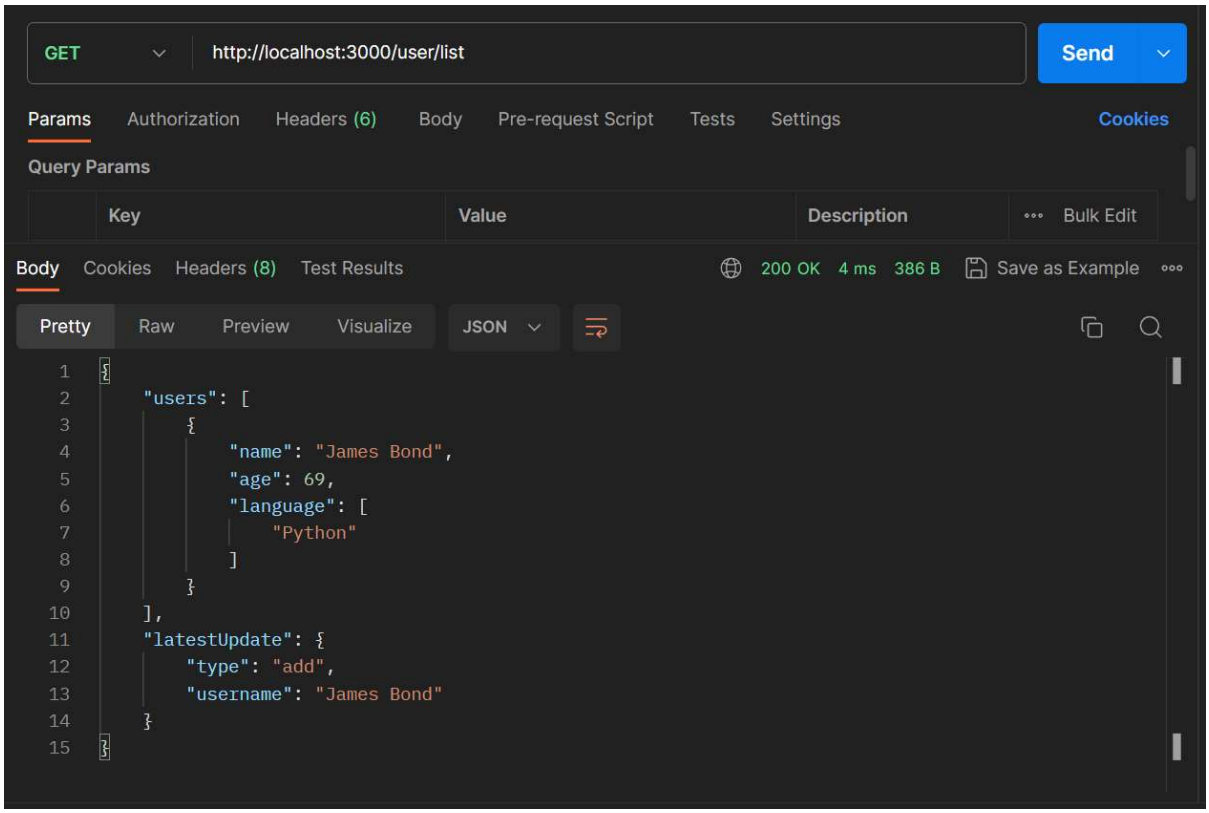
- That's it! You've successfully created and run the User Management application. You can now interact with the application through the web interface and test the functionalities to add, update, and delete users .

Interacting with API :

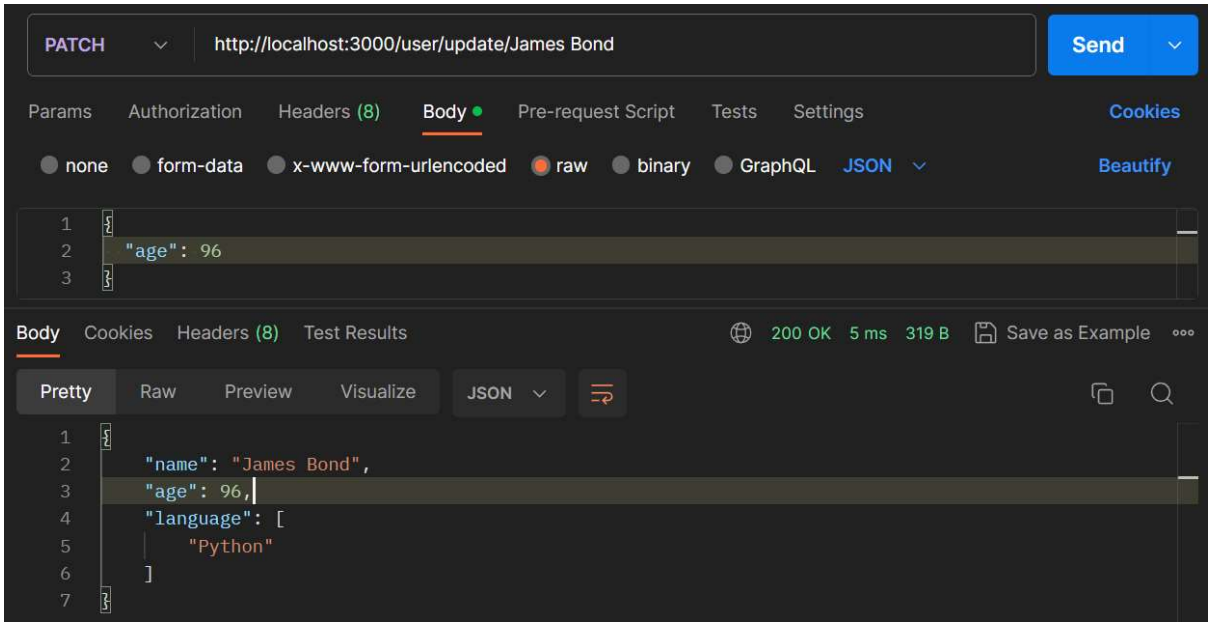
Add User: Send a POST request to http://localhost:3000/user/add with a JSON payload containing the user details.



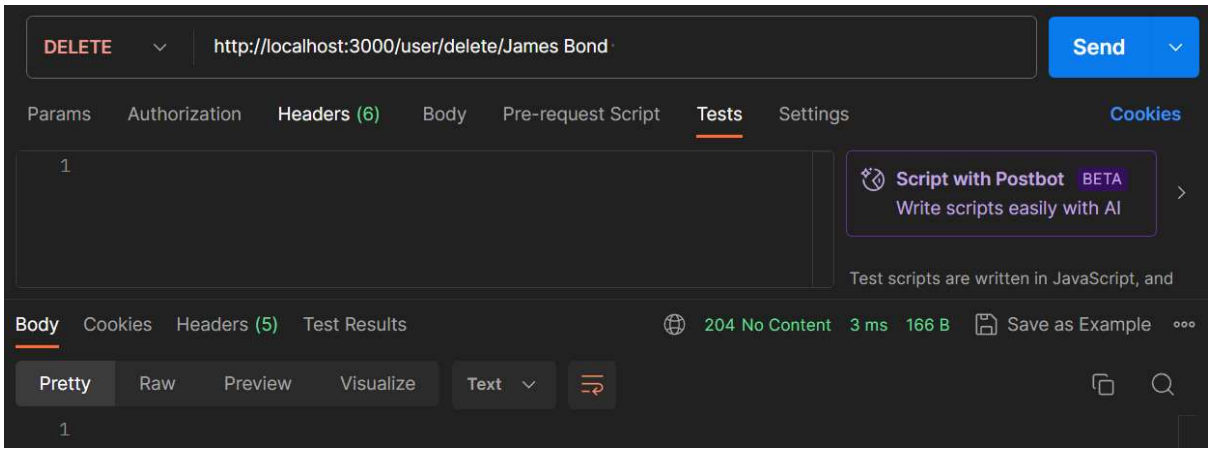
List All Users: Send a GET request to http://localhost:3000/user/list to get a list of all users.



Update User: Send a PATCH request to http://localhost:3000/user/update/:username (replace :username with the actual username) with a JSON payload containing the updated user details.

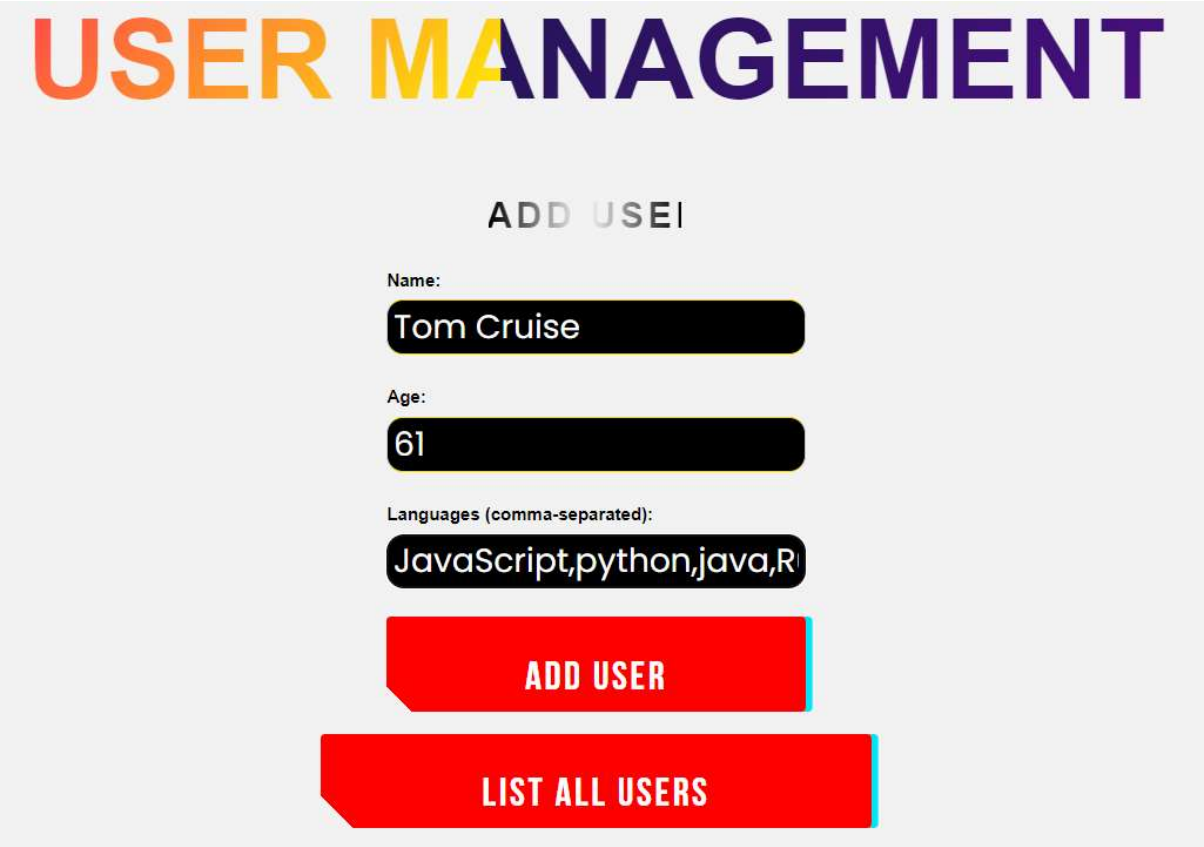


Delete User: Send a DELETE request to http://localhost:3000/user/delete/:username (replace :username with the actual username) to delete a user.

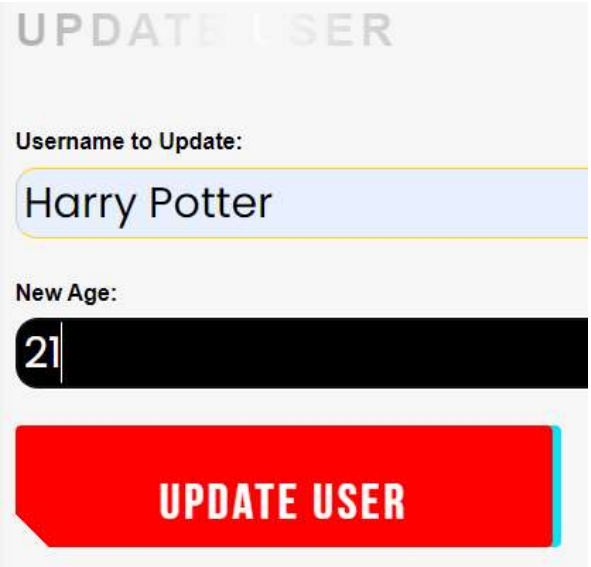


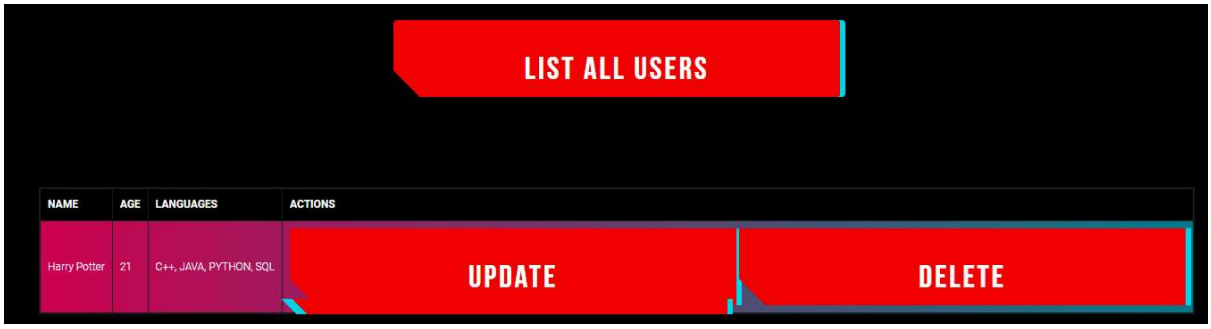
✓ Output :-

➪ Add User :

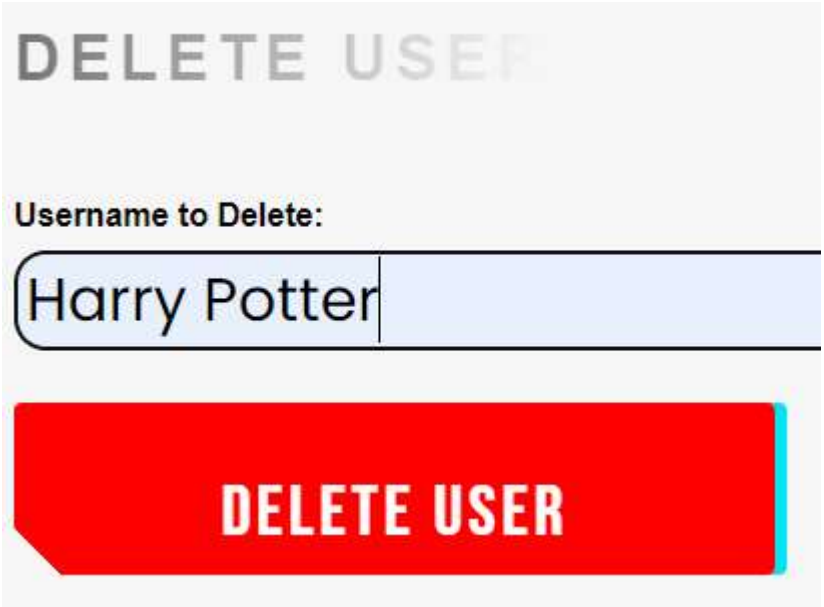


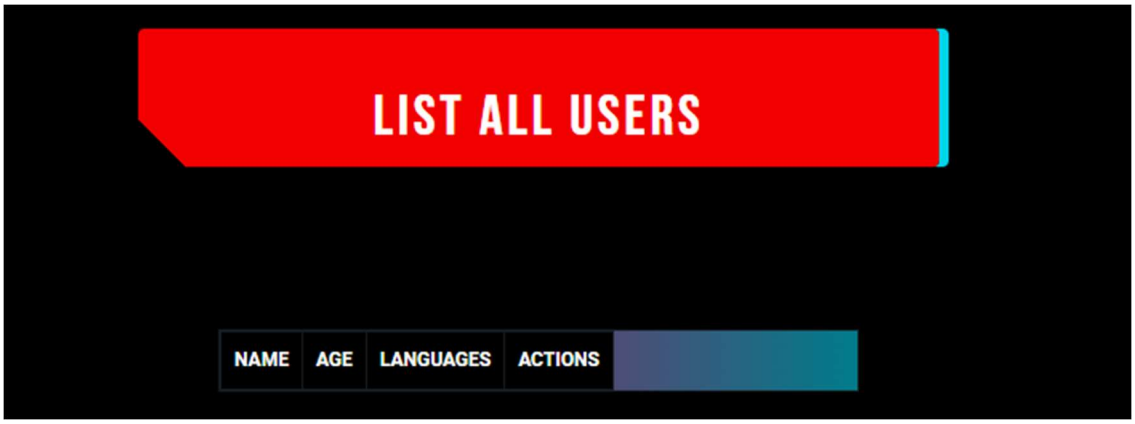
➪ Update User :





➥ Delete Employee by Name :





Get List of All Users Details :