

**Name: Patel Shivam S.**  
**Enroll no: 21162101019**  
**Sem: 5    Batch: 51    Branch: CBA**

**Sub: Microservices**

## **Practical 4**

**AIM:** Create REST API using NodeJS & Apply REST method to perform CRUD operations on resources at server regarding universities or Industries scenario.

1. Create a Rest API for universities or industry using NodeJS on any IDE
2. Perform CRUD operation on server using Postman.
3. Integrate it with HTML form, where it provides the option to POST the data of new employee, Get information of any employee based on ID, Get information of all employees, Update information of any employee based on their id and delete employee record based on their id

**GITHUB LINK:**

[https://github.com/Shivam3783/microservice\\_practicals/tree/main/prac4](https://github.com/Shivam3783/microservice_practicals/tree/main/prac4)

## CODE: 1)

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows a project structure with folders `OPEN EDITORS`, `PRAC4`, and files `Welcome`, `index.js`, `index.html`, `public`, `index.html`, `styles.css`, and `index.js`.
- EDITOR**: The `index.js` file is open, displaying the following code:

```
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const cors = require('cors');
4  const app = express();
5  app.use(cors());
6  app.use(bodyParser.urlencoded({ extended: false }));
7  app.use(bodyParser.json());
8  var port = 8080;
9
10 app.use(express.static(__dirname + '/public'));
11
12 var employees = [
13   { id: 1, name: 'abc', email: 'xyz@gmail.com', phone: 12345567890, age: 18 },
14   { id: 2, name: 'xya', email: 'abc@gmail.com', phone: 12345567890, age: 20 },
15   { id: 3, name: 'bnn', email: 'uyt@gmail.com', phone: 12345567890, age: 21 },
16   { id: 4, name: 'ytyt', email: 'pop@gmail.com', phone: 12345567890, age: 19 },
17 ];
18
19 var lastEmployeeId = employees.reduce((maxId, employee) => {
20   return Math.max(maxId, employee.id);
21 }, 0);
22
23 // let maxId = 0;
24 // for (let employee of employees) {
25 //   if (employee.id > maxId) {
26 //     maxId = employee.id;
27 //   }
28 }
29
30 app.get('/', (req, res) => {
31   res.send(`Hello ${req.query.name}!`);
32 });
33
34 app.listen(port, () => {
35   console.log(`Server is running on port ${port}`);
36 });
37
38 module.exports = app;
```

- TERMINAL**: Shows the following output from the Node.js application:

```
at done (/Users/shivampatel/Downloads/SEM 5/Micro Services/prac4/node_modules/raw-body/index.js:227:7)
at IncomingMessage.onEnd (/Users/shivampatel/Downloads/SEM 5/Micro Services/prac4/node_modules/raw-body/index.js:287:7)
at IncomingMessage.emit (node:events:513:28)
at endReadableNT (node:internal/streams/readable:1359:12)
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server is running on port http://localhost:8080
```

The screenshot shows the VS Code interface with the title bar "prac4". The Explorer sidebar on the left lists files: "OPEN EDITORS" (Welcome, index.js, index.html), "PRAC4" (node\_modules, public, index.html, styles.css), and "JS index.js" (selected). The main editor area displays the following code:

```
index.js > ...
83 app.get('/api/employe', (req, res) => {
84   res.send(employees);
85 })
86 app.get('/api/employe/:id', (req, res) => {
87   const employe = employees.find(({ id }) => id === parseInt(req.params.id));
88
89   if (!employe)
90     res.status(404).send(`<h2 style="font-family: Malgun Gothic; color: darkred;">Oops... Cant find what you're looking for!`);
91   else
92     res.send(employe);
93 }
94
95 app.get('/api/employe/:name', (req, res) => {
96   const employe = employees.find(({ name }) => name === req.params.name);
97
98   if (!employe)
99     res.status(404).send(`<h2 style="font-family: Malgun Gothic; color: darkred;">Oops... Cant find what you're looking for!`);
100  else
101    res.send(employe);
102
103 app.post('/api/employe', (req, res) => {
104   lastEmployeeId++;
105   var employe = {
106     id: lastEmployeeId,
107     name: req.body.name,
108     email: req.body.email,
109     phone: req.body.phone,
110     age: req.body.age,
111   };
112   employees.push(employe);
113   res.send(employe);
114 });
115
116 app.put('/api/employe/:id', (req, res) => {
117   var employe = employees.find(({ id }) => id === parseInt(req.params.id));
118   if (!employe) res.status(404).send(`<h2 style="font-family: Malgun Gothic; color: darkred;">Not Found!!`);
119   else employe.name = req.body.name;

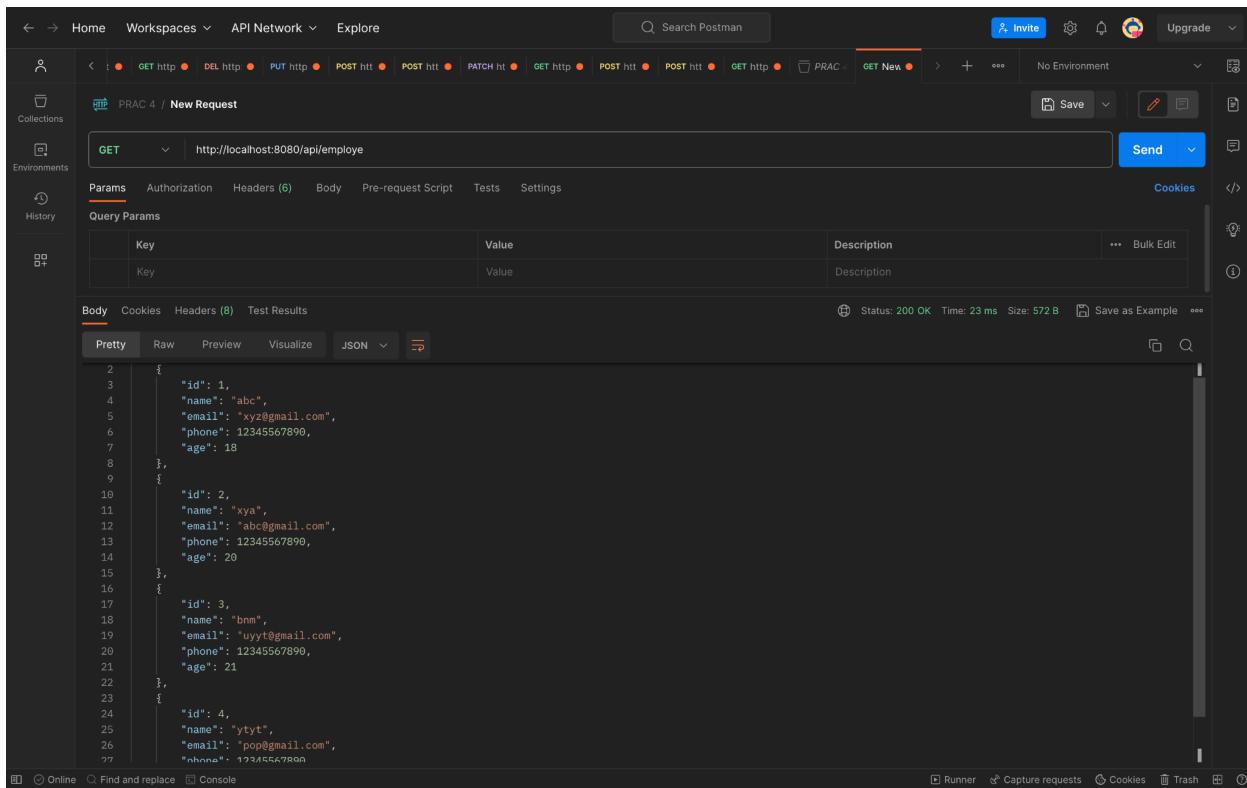
```

The screenshot shows the VS Code interface with the title bar "prac4". The Explorer sidebar on the left lists files: "OPEN EDITORS" (Welcome, index.js, index.html), "PRAC4" (node\_modules, public, index.html, styles.css), and "JS index.js" (selected). The main editor area displays the following code:

```
index.js > ...
138 app.delete('/api/employe/:id', (req, res) => {
139
140   const employe = employees.find(({ id }) => id === parseInt(req.params.id));
141   if (!employe) res.status(404).send(`<h2 style="font-family: Malgun Gothic; color: darkred;"> Not Found!!`);
142
143   const index = employees.indexOf(employe);
144   employees.splice(index, 1);
145
146   res.send(employe);
147 });
148
149 app.delete('/api/employe/:name', (req, res) => {
150
151   const employe = employees.find(({ name }) => name === req.params.name);
152   if (!employe) res.status(404).send(`<h2 style="font-family: Malgun Gothic; color: darkred;"> Not Found!!`);
153
154   const index = employees.indexOf(employe);
155   employees.splice(index, 1);
156
157   res.send(employe);
158 });
159
160
161 app.patch('/api/employe/patch/:id', (req, res) => {
162   var employe = employees.find(({ id }) => id === parseInt(req.params.id));
163   if (!employe) res.status(404).send(`<h2 style="font-family: Malgun Gothic; color: darkred;">Not Found!!`);
164
165   if (req.body.name) employe.name = req.body.name;
166   if (req.body.email) employe.email = req.body.email;
167   if (req.body.phone) employe.phone = req.body.phone;
168   if (req.body.age) employe.age = req.body.age;
169   if (req.body.sub) employe.sub = req.body.sub;
170
171   res.send(employe);
172 });
173
```

## 2) OUTPUT in POSTMAN:

## → Get All data (GET REQUEST)

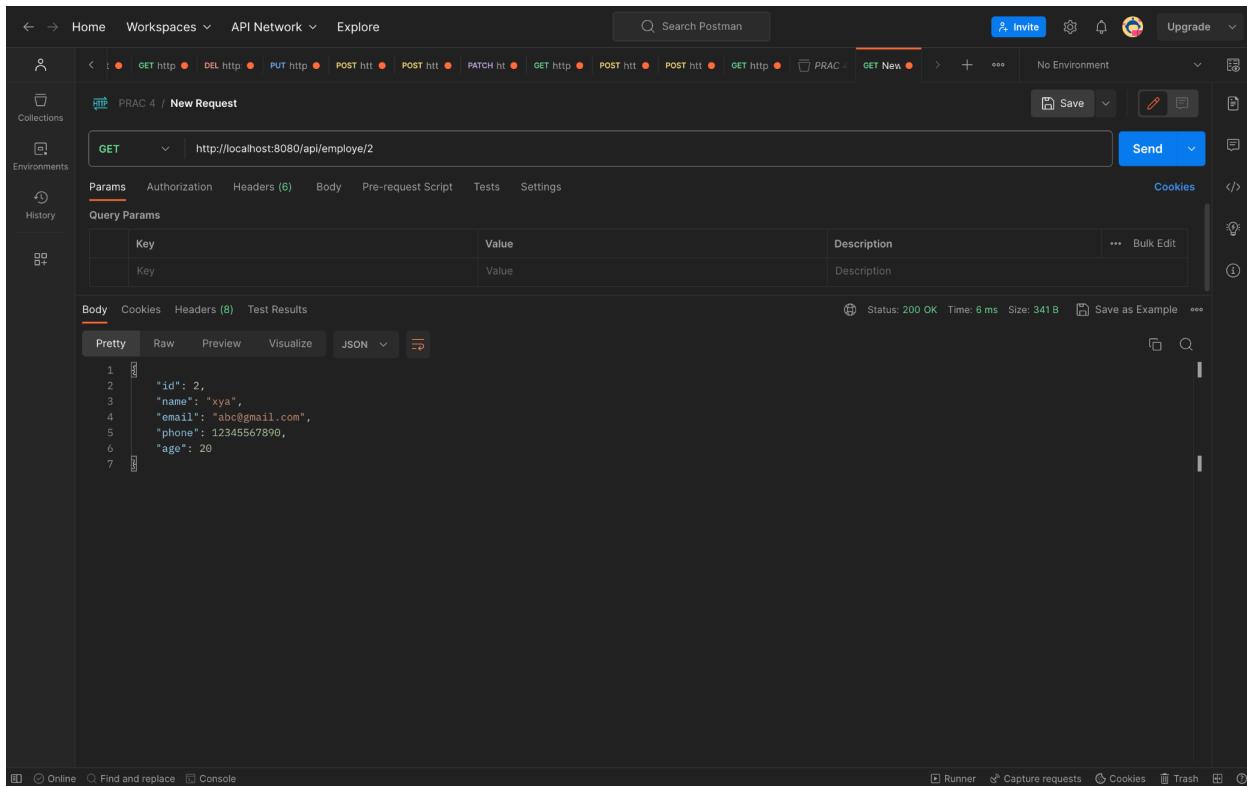


The screenshot shows the Postman interface with a collection named "PRAC 4". A new request is being prepared for the URL `http://localhost:8080/api/employee`. The "Body" tab is selected, displaying the following JSON response:

```
2 |   [
3 |     {
4 |       "id": 1,
5 |       "name": "abc",
6 |       "email": "xyz@gmail.com",
7 |       "phone": 12345567890,
8 |       "age": 18
9 |     },
10 |    {
11 |      "id": 2,
12 |      "name": "xya",
13 |      "email": "abc@gmail.com",
14 |      "phone": 12345567890,
15 |      "age": 20
16 |    },
17 |    {
18 |      "id": 3,
19 |      "name": "bnm",
20 |      "email": "uyyt@gmail.com",
21 |      "phone": 12345567890,
22 |      "age": 21
23 |    },
24 |    {
25 |      "id": 4,
26 |      "name": "tyt",
27 |      "email": "pop@mail.com",
28 |      "phone": 12345567890
29 |    }
30 |  ]
```

The status bar at the bottom indicates: Status: 200 OK Time: 23 ms Size: 572 B.

## → Get data by ID (GET REQUEST)



The screenshot shows the Postman interface with a collection named "PRAC 4". A new request is being prepared for the URL `http://localhost:8080/api/employee/2`. The "Body" tab is selected, displaying the following JSON response:

```
1 |   [
2 |     {
3 |       "id": 2,
4 |       "name": "xya",
5 |       "email": "abc@gmail.com",
6 |       "phone": 12345567890,
7 |       "age": 20
8 |     }
9 |   ]
```

The status bar at the bottom indicates: Status: 200 OK Time: 6 ms Size: 341 B.

## → Add data (POST REQUEST)

The screenshot shows the Postman interface with a 'New Request' tab titled 'PRAC 4'. The request method is set to 'POST' and the URL is 'http://localhost:8080/api/employee/'. The 'Body' tab is selected, showing JSON input:

```
1 {
2   "id": 5,
3   "name": "shivam_5",
4   "email": "5555@gmail.com",
5   "phone": 12345567890,
6   "age": 23
```

The response status is 200 OK with a time of 20 ms and a size of 347 B.

The screenshot shows the Postman interface with a 'New Request' tab titled 'PRAC 4'. The request method is set to 'GET' and the URL is 'http://localhost:8080/api/employee'. The 'Body' tab is selected, showing JSON output:

```
1 [
2   {
3     "id": 5,
4     "name": "shivam_5",
5     "email": "5555@gmail.com",
6     "phone": 12345567890,
7     "age": 23
8   }
9 ]
```

The response status is 200 OK with a time of 5 ms and a size of 653 B.

## → Update data (PUT REQUEST)

The screenshot shows the Postman interface with a 'PUT' request to `http://localhost:8080/api/employee/5`. The request body is a JSON object:

```
1 {
2   "name": "shivam_5_updated",
3   "email": "update@gmail.com",
4   "phone": 14345567890,
5   "age": 88
6 }
```

The response status is 200 OK with a time of 6 ms and a size of 357 B. The response body is identical to the request body.

The screenshot shows the Postman interface with a 'GET' request to `http://localhost:8080/api/employee`. The response status is 200 OK with a time of 5 ms and a size of 744 B. The response body is a JSON array of five employee objects:

```
1 [
2   {
3     "id": 1,
4     "name": "abc@gmail.com",
5     "email": "12345567890",
6     "phone": 12345567890,
7     "age": 20
8   },
9   {
10     "id": 2,
11     "name": "bnm",
12     "email": "uyt@gmail.com",
13     "phone": 12345567890,
14     "age": 21
15   },
16   {
17     "id": 3,
18     "name": "tyt",
19     "email": "pop@gmail.com",
20     "phone": 12345567890,
21     "age": 19
22   },
23   {
24     "id": 4,
25     "name": "tyt",
26     "email": "pop@gmail.com",
27     "phone": 12345567890,
28     "age": 19
29   },
30   {
31     "id": 5,
32     "name": "shivam_5_updated",
33     "email": "update@gmail.com",
34     "phone": 14345567890,
35     "age": 88
36   }
]
```

## → Delete data (DELETE REQUEST)

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8080/api/employee/5`. The response status is 200 OK with a size of 357 B.

```
1   "id": 5,
2   "name": "shivam_5_updated",
3   "email": "update@gmail.com",
4   "phone": 12345567890,
5   "age": 88
```

Body tab content:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/employee`. The response status is 200 OK with a size of 572 B.

```
1
2
3
4
5
6   "id": 1,
7   "name": "xyz",
8   "email": "abc@gmail.com",
9   "phone": 12345567890,
10  "age": 18
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

Body tab content:

### 3) CODE: HTML Part

The screenshot shows the VS Code interface with the file `index.js` open in the editor. The code implements a REST API for managing employees using Express.js. It includes routes for getting all employees, getting an employee by ID, creating a new employee, updating an employee, and deleting an employee. The code uses a local database of employees and handles requests from the client.

```
JS index.js # styles.css index.html
JS index.js
# styles.css public
index.html public
PRAC4 node_modules
public
index.html
# styles.css
JS index.js
{} package-lock.json
{} package.json
B
...
OUTLINE
TIMELINE
MYSQL
prac4
```

```
JS index.js > ...
33 app.get('/', (req, res) => {
34   res.send('Welcome to Ganpat REST API!');
35 });
36
37 app.get('/api/all', (req, res) => {
38   res.send(employees);
39 })
40 app.post('/api/get', (req, res) => {
41   const employe = employees.find({ id }) => id == req.body.id;
42
43   if (!employe)
44     res.status(404).send('<h2 style="font-family: Malgun Gothic; color: darkred;">Oops... Cant find what you are looking for</h2>');
45   else
46     res.send(employe);
47 }
48 app.post('/api/employe', (req, res) => {
49   lastEmployeeId++;
50   var employe = {
51     id: lastEmployeeId,
52     name: req.body.name,
53     email: req.body.email,
54     phone: req.body.phone,
55     age: req.body.age
56   };
57   employees.push(employe);
58   res.send(employe);
59 });
60
61 app.post('/api/put', (req, res) => {
62   var employe = employees.find({ id }) => id == req.body.id;
63   if (!employe) res.status(404).send('<h2 style="font-family: Malgun Gothic; color: darkred;">Not Found!! </h2>');
64   employe.name= req.body.name;
65   employe.email= req.body.email;
66   employe.phone= req.body.phone;
67   employe.age= req.body.age;
68
69   res.send(employe);
70 }
```

The screenshot shows the VS Code interface with the file `index.html` open in the editor. The code defines an HTML page for managing employees. It features a title, a header, and two forms: one for getting employee data by ID and another for adding new employee data. The page uses CSS for styling and links to the `styles.css` file.

```
EXPLORER ... prac4
OPEN EDITORS Welcome JS index.js
index.html public
PRAC4 node_modules
public
index.html
# styles.css
JS index.js
{} package-lock.json
{} package.json
B
...
OUTLINE
TIMELINE
MYSQL
prac4
```

```
EXPLORER ... prac4
OPEN EDITORS Welcome index.html
index.html public
PRAC4 node_modules
public
index.html
# styles.css
JS index.js
{} package-lock.json
{} package.json
B
...
OUTLINE
TIMELINE
MYSQL
prac4
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Employee Management</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Employee Management</h1>
    <div class="forms">
      <h2>Get Employee Data by ID</h2>
      <form id="getForm" action="/api/get" method="POST">
        <label for="vhn">Enter Employee ID:</label>
        <input type="number" id="getId" name="id" required>
        <button type="submit">Get Employee Data</button>
      </form>

      <h2>Add New Employee</h2>
      <form id="addForm" action="/api/employe" method="POST">
        <label for="Name">Name:</label>
        <input type="text" id="Name" name="name">
        <label for="Email">Email:</label>
        <input type="email" id="Email" name="email">
        <label for="Phone">Phone:</label>
        <input type="tel" id="Phone" name="phone">
      </form>
    </div>
  </div>
</body>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

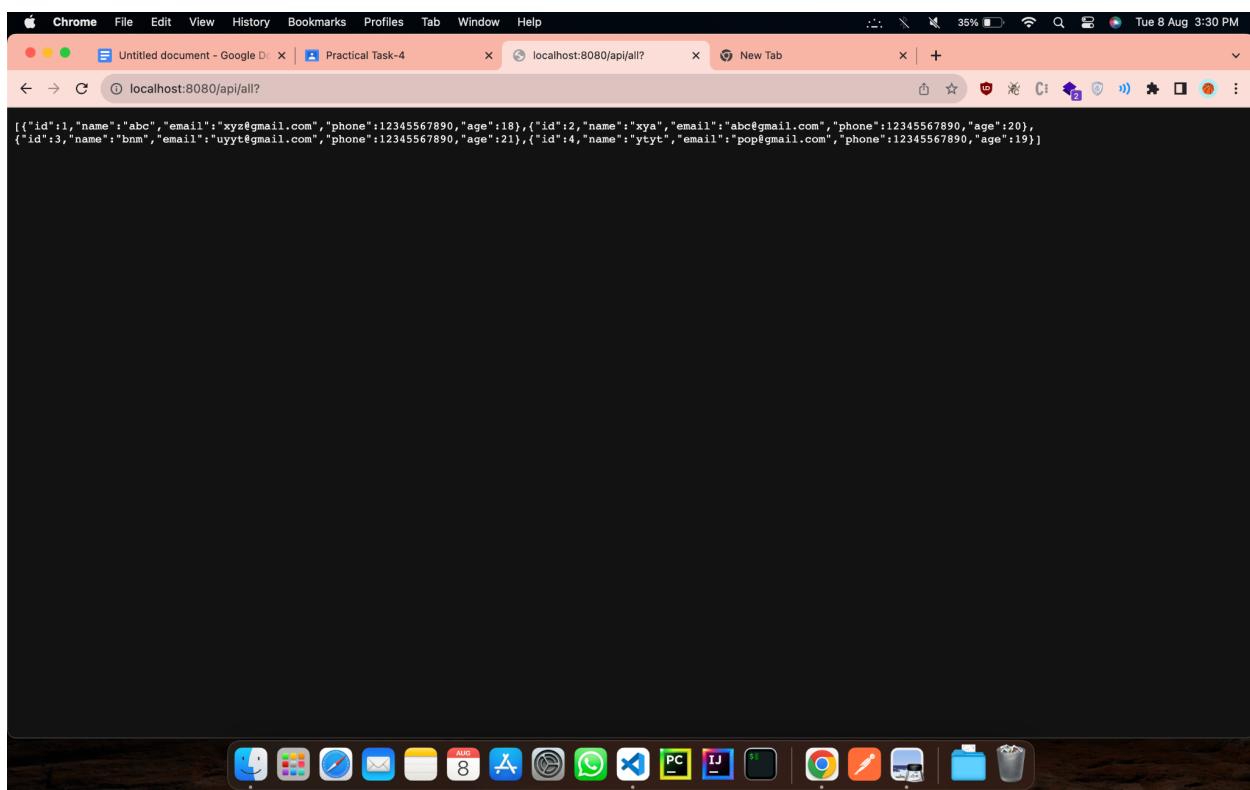
```
at done (/Users/shivampatel/Downloads/SEM 5/Micro Services/prac4/node_modules/raw-body/index.js:227:7)
at IncomingMessage.onEnd (/Users/shivampatel/Downloads/SEM 5/Micro Services/prac4/node_modules/raw-body/index.js:287:7)
at IncomingMessage.emit (node:events:513:28)
at endReadableNT (node:internal/streams/readable:1359:12)
[nodemon] restarting due to changes...
[nodemon] starting 'node index.js'
Server is running on port http://localhost:8080
```

OUTPUT:

## → Get all data

The screenshot shows a web browser window with the following details:

- Address Bar:** localhost:8080
- Content Area:**
  - Name:** Input field
  - Email:** Input field
  - Phone:** Input field
  - Age:** Input field
  - Buttons:** "Update Employee Data" (blue)
  - Section:** Delete Employee Data
  - Text:** Enter Employee ID: Input field
  - Buttons:** "Delete Employee Data" (blue)
  - Section:** Get All Employee Data
  - Buttons:** "Get All Data" (blue)



## → Get data by ID

The screenshot shows a Chrome browser window with the URL `localhost:8080`. The page title is "Employee Management". Below it, a section titled "Get Employee Data by ID" contains a form with a text input field labeled "Enter Employee ID:" and a blue button labeled "Get Employee Data".

The screenshot shows a Chrome browser window with the URL `localhost:8080/api/get`. The page displays the JSON response: 

```
{"id":2,"name":"xya","email":"abc@gmail.com","phone":12345567890,"age":20}
```

## → Add data

Add New Employee

Name: shivam\_5

Email: shivammp019@gmail.com

Phone: 5555555555

Age: 22

Add Employee

## Update Employee Data

Enter Employee ID:

Name:

```
[{"id":1,"name":"abc","email":"xyz@gmail.com","phone":12345567890,"age":18}, {"id":2,"name":"xya","email":"abc@gmail.com","phone":12345567890,"age":20}, {"id":3,"name":"bnn","email":"uyyt@gmail.com","phone":12345567890,"age":21}, {"id":4,"name":"tyt","email":"pop@gmail.com","phone":12345567890,"age":19}, {"id":5,"name":"shivam_5","email":"shivammp019@gmail.com","phone":5555555555,"age":22}]
```

## → Update data

The screenshot shows a web browser window titled "Employee Management" on the tab bar. The URL in the address bar is "localhost:8080". The main content area displays a form titled "Update Employee Data". The form fields are as follows:

- Enter Employee ID:
- Name:
- Email:
- Phone:
- Age:

Below the form is a blue button labeled "Update Employee Data".

The screenshot shows a web browser window titled "Employee Management" on the tab bar. The URL in the address bar is "localhost:8080/api/all?". The main content area displays a form titled "Delete Employee Data". The form field is:

Enter Employee ID:

Below the form is a blue button labeled "Delete Employee Data".

At the bottom of the browser window, the status bar shows the date and time as "Tue 8 Aug 12:15 PM".

On the right side of the browser window, there is a JSON response displayed in a dark panel:

```
[{"id":1,"name":"abc","email":"xyz@gmail.com","phone":12345567890,"age":18}, {"id":2,"name":"xya","email":"abc@gmail.com","phone":12345567890,"age":20}, {"id":3,"name":"bnm","email":"uyyt@gmail.com","phone":12345567890,"age":21}, {"id":4,"name":"shivam_4","email":"shivammp019@gmail.com","phone":8511555669,"age":12}, {"id":5,"name":"shivam_5","email":"shivammp019@gmail.com","phone":5555555555,"age":22}]
```

## → Delete data

The screenshot shows a web application interface with three main sections:

- Update Employee Data:** Contains fields for Name, Email, Phone, and Age, followed by a blue "Update Employee Data" button.
- Delete Employee Data:** Contains a dropdown menu for Enter Employee ID (set to 4) and a blue "Delete Employee Data" button.
- Get All Employee Data:** Contains a blue "Get All Data" button.

The screenshot shows the results of a GET request to `localhost:8080/api/all?`. The response is a JSON array containing four employee records:

```
[{"id":1,"name":"abc","email":"xyz@gmail.com","phone":12345567890,"age":18}, {"id":2,"name":"xya","email":"abc@gmail.com","phone":12345567890,"age":20}, {"id":3,"name":"bnm","email":"uyyt@gmail.com","phone":12345567890,"age":21}, {"id":5,"name":"shivam_5","email":"shivammp019@gmail.com","phone":5555555555,"age":22}]
```