Ganpat University | Institute of Computer Technology

॥ विद्यया समाजोत्कर्षः ॥

Name: Tushar Panchal

En.No: 21162101014

Sub: MICROSEVICES

Branch: CBA

Batch:51

----------PRACTICAL 04----------

❖ Question (TASK) :

You are building a Node.js application for a scheduling platform. As part of the functionality, you need to display the current date and time whenever a user interacts with the platform.

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 usingPostman.

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

❖ 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 :-

```html
<!DOCTYPE html>
<html>

<head>
  <title>Employee Management</title>
  <link rel="stylesheet" href="/styles.css">
</head>

<body>
  <div class="container">
    <p class="animate-charcter">Employee Management</p>

    <div class="add-employee">
      <h2>Add Employee</h2>
      <form id="employeeForm">
        <label for="id">ID:</label>
        <input type="number" id="id" name="id" required><br>
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br>
        <label for="role">Role:</label>
        <input type="text" id="role" name="role" required><br>
        <label for="department">Department:</label>
        <input type="text" id="department" name="department" required><br>
        <button type="submit">Add Employee</button>
      </form>
    </div>

    <div class="get-employee">
      <h2>Get Employee Details by ID</h2>
      <form id="getEmployeeForm">
        <label for="employeeId">Employee ID:</label>
        <input type="number" id="employeeId" name="employeeId" required>
        <button type="submit">Get Employee Details</button>
      </form>
    </div>



    <div class="update-employee">
      <h2>Update Employee</h2>
      <form id="updateForm">
        <label for="updateId">Employee ID:</label>
        <input type="number" id="updateId" name="updateId" required><br>
        <label for="updateName">New Name:</label>
        <input type="text" id="updateName" name="updateName" required><br>
        <label for="updateRole">New Role:</label>
        <input type="text" id="updateRole" name="updateRole" required><br>
        <label for="updateDepartment">New Department:</label>
        <input type="text" id="updateDepartment" name="updateDepartment" required><br>
        <button type="submit">Update Employee</button>
      </form>
    </div>


    <div class="delete-employee">
      <h2> Delete Employee</h2>
      <form id="deleteForm">
        <label for="deleteName">Employee Name:</label>
        <input type="text" id="deleteName" name="deleteName" required>
```

```
        <button type="submit">Delete Employee</button>
      </form>
    </div>

    <div id="employeeInfo" class="btn-shine">

    </div>
  <button id="getAllEmployeesButton">Get All Employee Details</button>

  </div>

  <script src="/script.js"></script>
</body>

</html>
```

✓ Server.js :-

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path'); // Import the path module to work with file paths

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

// Sample data for universities or industries
let employees = [
  { id: 1, name: 'John Doe', role: 'Professor', department: 'Computer Science' },
  { id: 2, name: 'Jane Smith', role: 'Engineer', department: 'Mechanical Engineering' },
];

// Serve static files from the "public" directory
app.use(express.static(path.join(__dirname, 'public')));

// Route to get all employees
app.get('/employees', (req, res) => {
  res.json(employees);
});

// Route to get an employee by ID
app.get('/employees/:id', (req, res) => {
  const id = parseInt(req.params.id);
  const employee = employees.find((emp) => emp.id === id);
  if (employee) {
    res.json(employee);
  } else {
    res.status(404).json({ error: 'Employee not found' });
  }
});

// Route to add a new employee
app.post('/employees', (req, res) => {
  const { id, name, role, department } = req.body;
  if (!id || !name || !role || !department) {
    return res.status(400).json({ error: 'Please provide all required fields' });
  }

  const newEmployee = { id, name, role, department };
  employees.push(newEmployee);
  console.log("New employee added:", newEmployee);
```

```
  res.status(201).json(newEmployee);
});

// Route to update an existing employee by ID

// Route to update an existing employee by ID
app.put('/employees/:id', (req, res) => {
  const id = parseInt(req.params.id);
  const { name, role, department } = req.body;
  const employeeIndex = employees.findIndex((emp) => emp.id === id);

  if (employeeIndex !== -1) {
    // Employee with the provided ID exists, update the data
    employees[employeeIndex].name = name || employees[employeeIndex].name;
    employees[employeeIndex].role = role || employees[employeeIndex].role;
    employees[employeeIndex].department = department ||
employees[employeeIndex].department;
    console.log("Employee updated:", employees[employeeIndex]);
    res.json(employees[employeeIndex]);
  } else {
    // Employee with the provided ID doesn't exist, create a new employee
    const newEmployee = { id, name, role, department };
    employees.push(newEmployee);
    console.log("New employee added:", newEmployee);
    res.status(201).json(newEmployee);
  }
});

// Route to delete an employee by NAME
app.delete('/employees/:name', (req, res) => {
  const name = req.params.name;
  employees = employees.filter((emp) => emp.name !== name);
  res.json({ message: 'Employee deleted successfully' });
});

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

// Start the server
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

✓ Script.js :-

```
const form = document.getElementById('employeeForm');
const updateForm = document.getElementById('updateForm');
const deleteForm = document.getElementById('deleteForm');
const employeeInfo = document.getElementById('employeeInfo');

form.addEventListener('submit', (e) => {
  e.preventDefault();
  const id = form.elements.id.value;
  const name = form.elements.name.value;
  const role = form.elements.role.value;
  const department = form.elements.department.value;

  fetch('/employees', {
    method: 'POST',
```

```
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ id, name, role, department }),
  })
    .then((response) => response.json())
    .then((data) => {
      employeeInfo.innerHTML = `
        <p>Employee added successfully:</p>
        <p>ID: ${data.id}</p>
        <p>Name: ${data.name}</p>
        <p>Role: ${data.role}</p>
        <p>Department: ${data.department}</p>
      `;
      form.reset();
    })
    .catch((error) => {
      employeeInfo.innerHTML = `<p>Error: ${error.message}</p>`;
    });
});

updateForm.addEventListener('submit', (e) => {
  e.preventDefault();
  const updateId = updateForm.elements.updateId.value;
  const updateName = updateForm.elements.updateName.value;
  const updateRole = updateForm.elements.updateRole.value;
  const updateDepartment = updateForm.elements.updateDepartment.value;

  fetch(`/employees/${updateId}`, {
    method: 'PUT',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ id:updateId,name: updateName, role: updateRole, department:
updateDepartment }),
  })
    .then((response) => response.json())
    .then((data) => {
      employeeInfo.innerHTML = `
        <p>Employee updated successfully:</p>
        <p>ID: ${data.id}</p>
        <p>Name: ${data.name}</p>
        <p>Role: ${data.role}</p>
        <p>Department: ${data.department}</p>
      `;
      updateForm.reset();
    })
    .catch((error) => {
      employeeInfo.innerHTML = `<p>Error: ${error.message}</p>`;
    });
});

deleteForm.addEventListener('submit', (e) => {
  e.preventDefault();
  const deleteName = deleteForm.elements.deleteName.value;

  fetch(`/employees/${deleteName}`, {
    method: 'DELETE',
  })
    .then((response) => response.json())
    .then((data) => {
      employeeInfo.innerHTML = `<p>${data.message}</p>`;
```

```
      deleteForm.reset();
    })
    .catch((error) => {
      employeeInfo.innerHTML = `<p>Error: ${error.message}</p>`;
    });
});
```

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'** 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`

Step 3 :

- Set up the server.
- This server.js code sets up an Express server, serves static files from the "public" directory, and defines various routes to handle employee data.

Step 4 :

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

Step 5 :

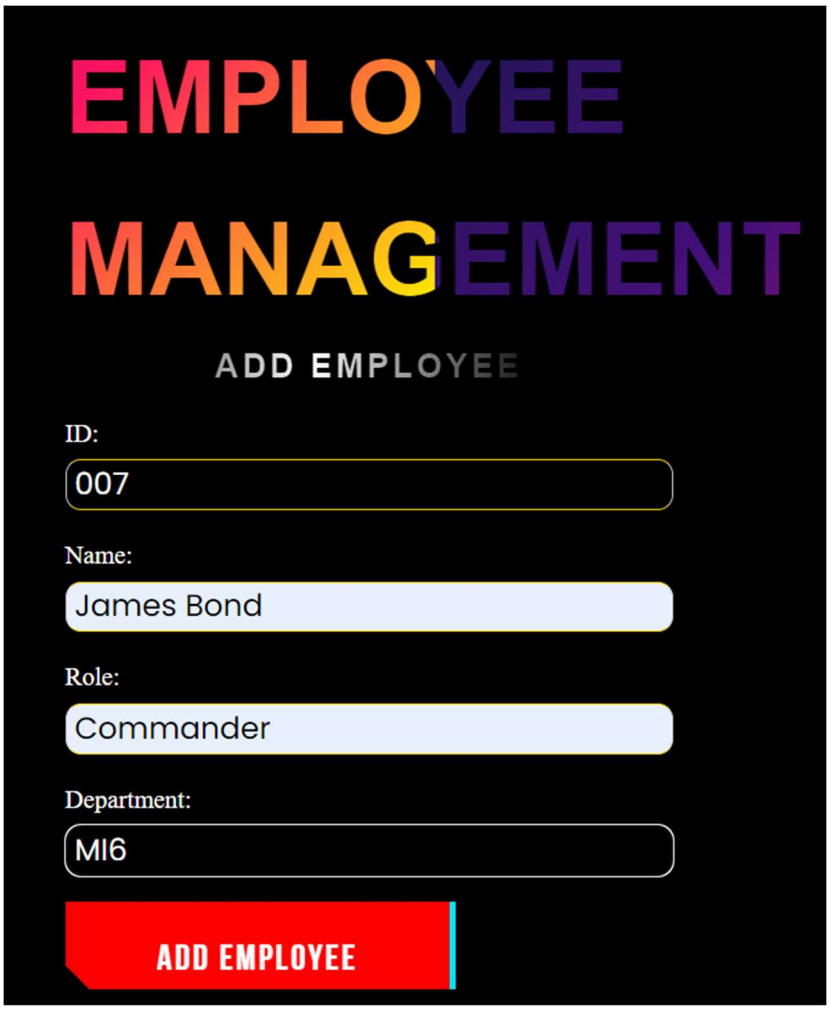
- Access the application
- Open web browser and enter the following address:

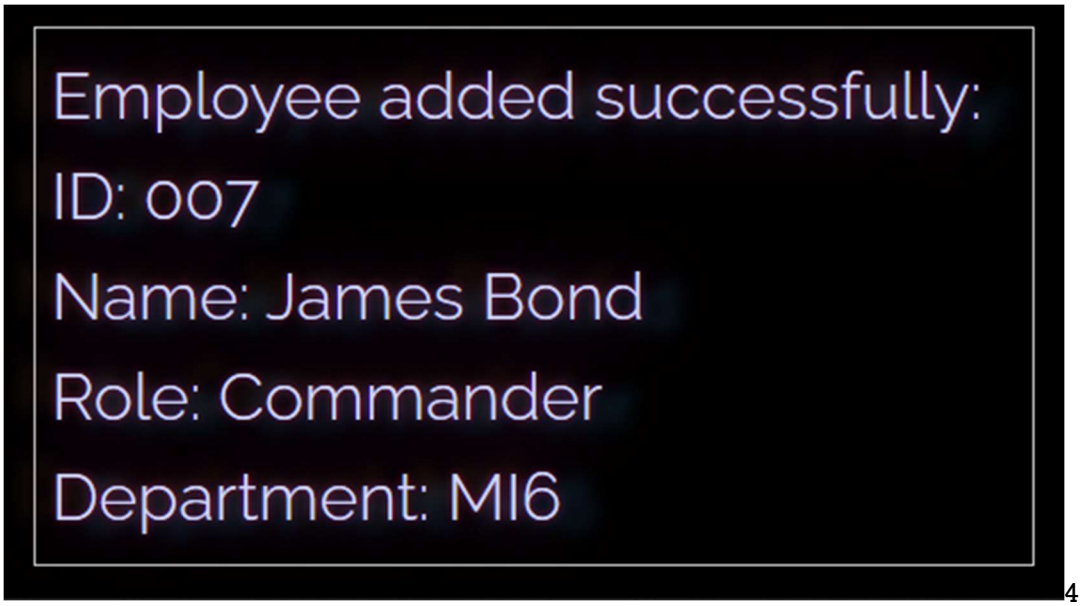
http://localhost:3000

- That's it! You've successfully created and run the Employee Management application. You can now interact with the application through the web interface and test the functionalities to add, update, and delete employees. The data is managed on the server using the defined routes in **'server.js'** .

✓ **Output :-**

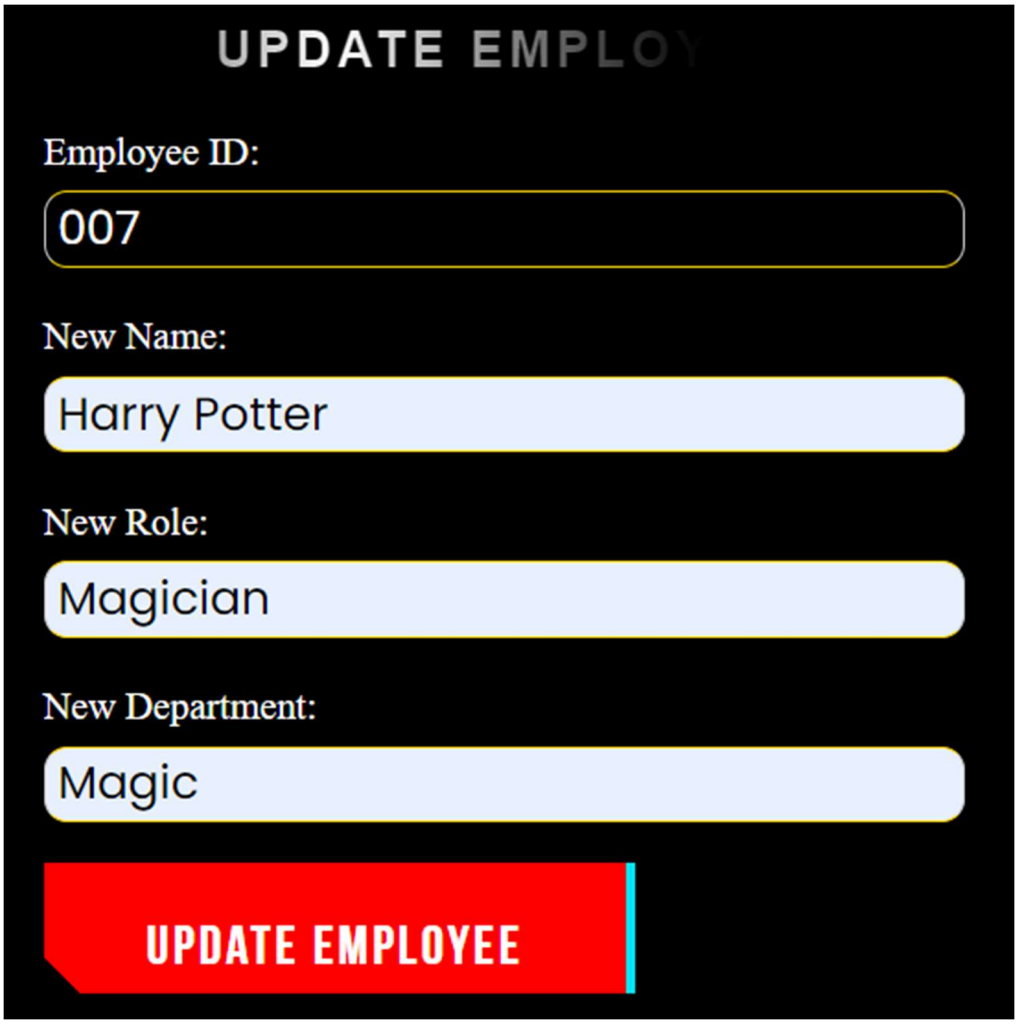
➡ **Add Employee :**

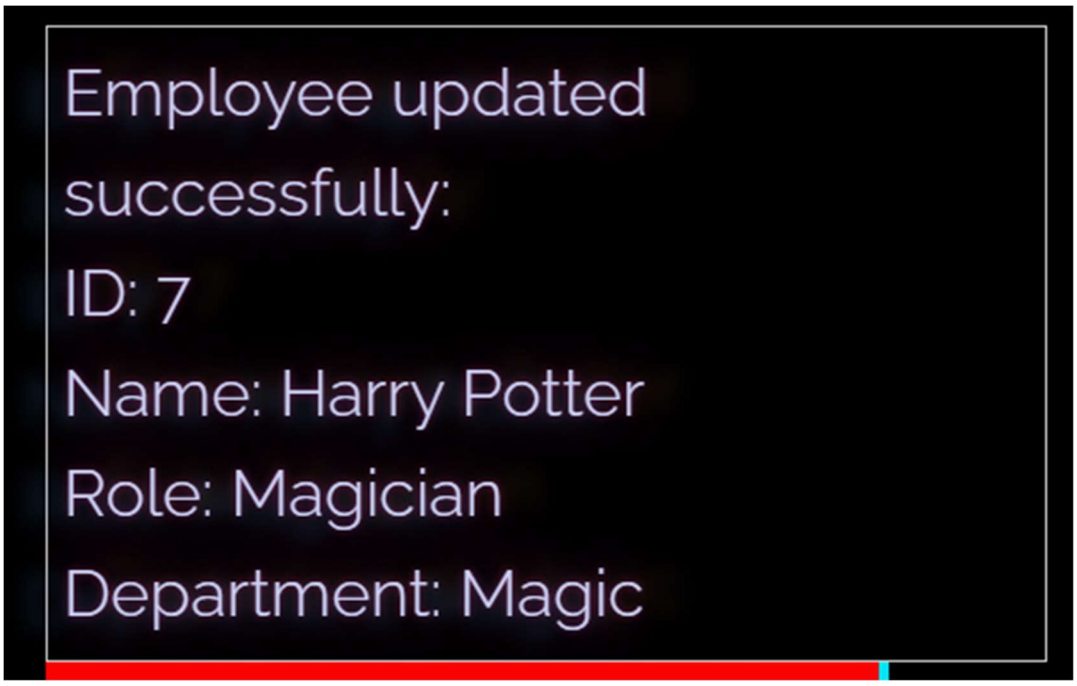




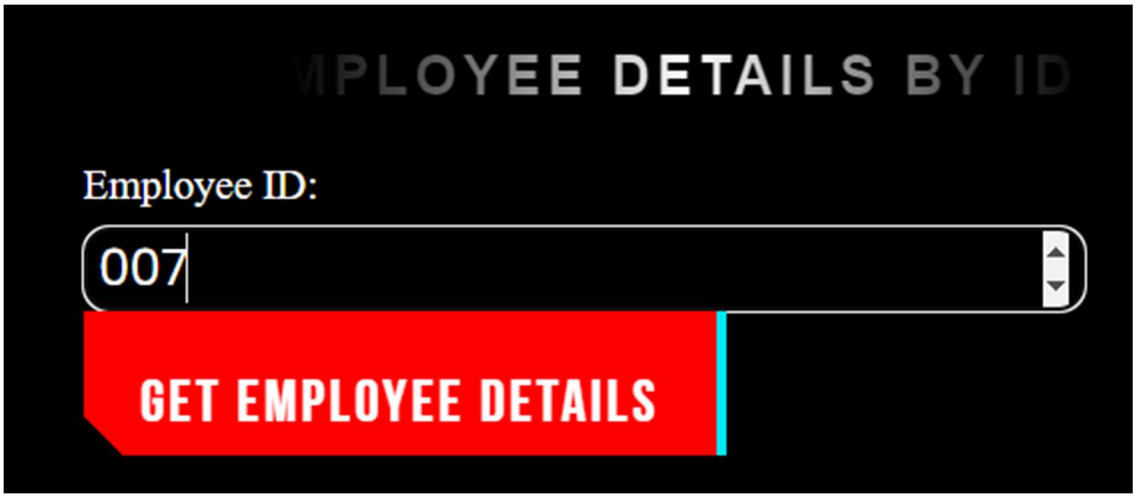
4

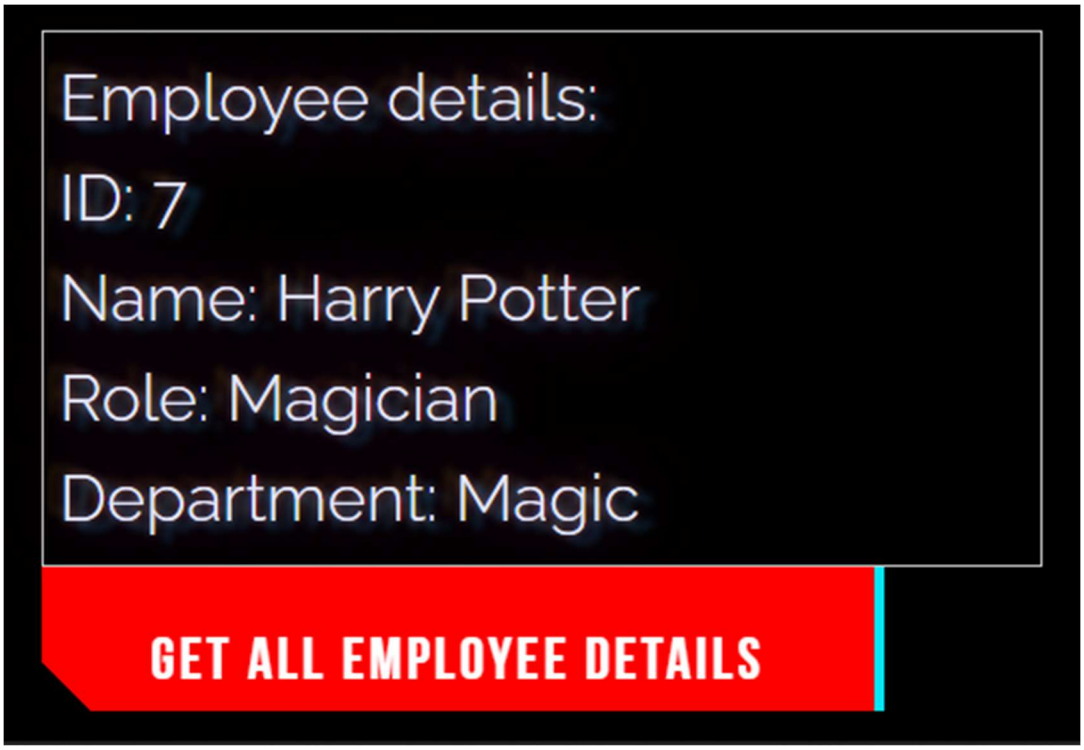
Update Employee :



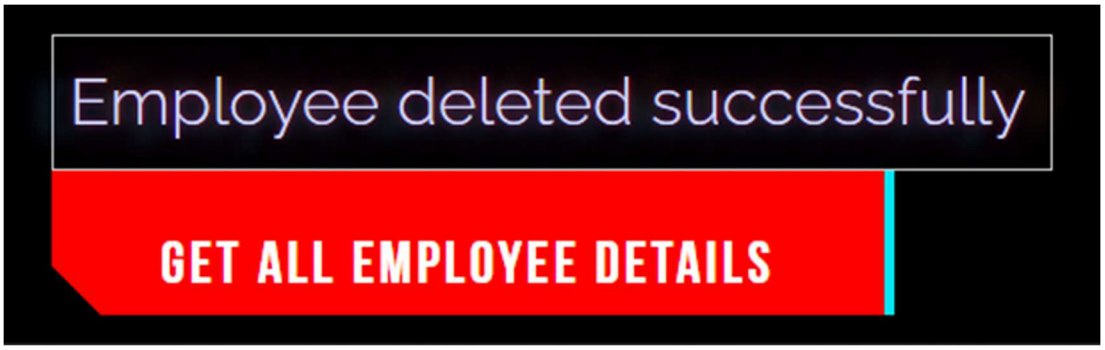


Get Employee by id :





Delete Employee by Name :



➡ Get All Employees Details :