**Guvi mini project – 1**

( Back end Documentation )

This document outlines the back-end API provided by the EmployeeController class for managing employee data in an employee system application.

**Technology Stack:**

* Spring Framework
* MongoDB (based on ObjectId usage)
* **Employee Entity:**
* The Employee entity (not shown here) represents an employee record with its corresponding attributes.
* **Employee Service:**
* The EmployeeService class (assumed to exist) handles the business logic for employee data operations. This documentation focuses on the controller exposing these functionalities through endpoints.
* **API Endpoints:**

1. **Get All Employees:**

 **HTTP Method:** GET

 **URL:** / (root path)

 **Return:**

* A list of Employee objects representing all employees in the system.

 **Example Response:**

JSON

[

{

"id": "ObjectId(...)",

"name": "John Doe",

"email": "john.doe@example.com",

"department": "Engineering"

},

// ... other employee objects

]

1. **Add a New Employee:**

 **HTTP Method:** POST

 **URL:** /register

 **Request Body:**

* An Employee object containing the new employee's details.

 **Return:**

* Redirects to the home page (/) upon successful registration.

**(Improvement)** Stores a success message in a session attribute (successMsg) for UI feedback (implement as needed).

 **Example Request Body:**

JSON

{

"name": "Jane Smith",

"email": "jane.smith@example.com",

"department": "Sales"

}

1. **Get Employee Edit Form:**

 **HTTP Method:** GET

 **URL:** /edit/{id}

 **Path Variable:**

* {id}: The ObjectId of the employee to edit.

 **Return:**

* Renders the "edit.html" template with the employee details pre-filled for editing.

 **Error Handling:**

* If the employee with the given ID is not found, handle the error appropriately (e.g., redirect to an error page).

1. **Update Employee Data:**

 **HTTP Method:** POST

 **URL:** /update

 **Request Body:**

* An Employee object containing the updated employee details.

 **Return:**

* Redirects to the home page (/) upon successful update.
* **(Improvement)** Stores a success message in a session attribute (successMsg) for UI feedback (implement as needed).

 **Error Handling:**

* Handle potential errors during update (e.g., data validation issues).

**5. Delete an Employee:**

* **HTTP Method:** GET
* **URL:** /delete/{id}
* **Path Variable:**
  + {id}: The ObjectId of the employee to delete.
* **Return:**
  + Redirects to the home page (/) upon successful deletion.
  + **(Improvement)** Stores a success message in a session attribute (successMsg) for UI feedback (implement as needed).
* **Error Handling:**
  + If the employee with the given ID is not found, handle the error appropriately (e.g., redirect to an error page).

**Additional Notes:**

* This documentation assumes the existence of corresponding HTML templates (index.html, add\_employee.html, and edit.html) for rendering views.
* Error handling details are not explicitly shown in the code snippets but should be implemented for a robust API.
* Security aspects like user authentication and authorization are not covered here but are crucial for real-world applications.
* The original code in the prompt had a logical error in the updateEmployee method where service.addEmployee was used instead of service.updateEmployee. This documentation assumes the correct usage of service.updateEmployee for updating employee data.

This documentation provides a clear overview of the employee management functionalities offered by the EmployeeController through its API endpoints. It also incorporates improvements for session-based success messages.