**EMPLOYEE MANAGEMENT APPLICATION**

**Introduction**

The Employee Management System is a web-based application that enables seamless employee data management within an organization. The system allows administrators to perform CRUD operations on employee records, manage user accounts, assign roles, and handle user registration securely. Built using Spring Boot with Thymeleaf, it follows an MVC design pattern and integrates features like pagination, sorting, and user authentication.

**Technology Stack**

* Backend: Spring Boot (Java)
* Frontend: Thymeleaf (HTML, CSS, Bootstrap)
* Database: H2 Database / MySQL
* Security: Spring Security
* API Documentation: SpringDoc OpenAPI / Swagger

**Core Functionalities**

**1. Employee Management**

* Create, update, delete, and view employee records.
* Search employees with pagination and sorting features.
* View employee details on a user-friendly dashboard**.**

**2. User Registration and Authentication**

* New users can register their accounts via a registration page.
* Secure password storage using BCrypt hashing.
* Role-based access management (e.g., admin or user).

**3. Role Management**

* Users are assigned specific roles (ROLE\_USER, ROLE\_ADMIN).
* Roles determine the accessibility and visibility of system features.

**4. Pagination and Sorting**

* The application supports pagination for large data sets.
* Data can be sorted dynamically by various fields (e.g., first name, last name).

**Usage Instructions**

**1. Setup**

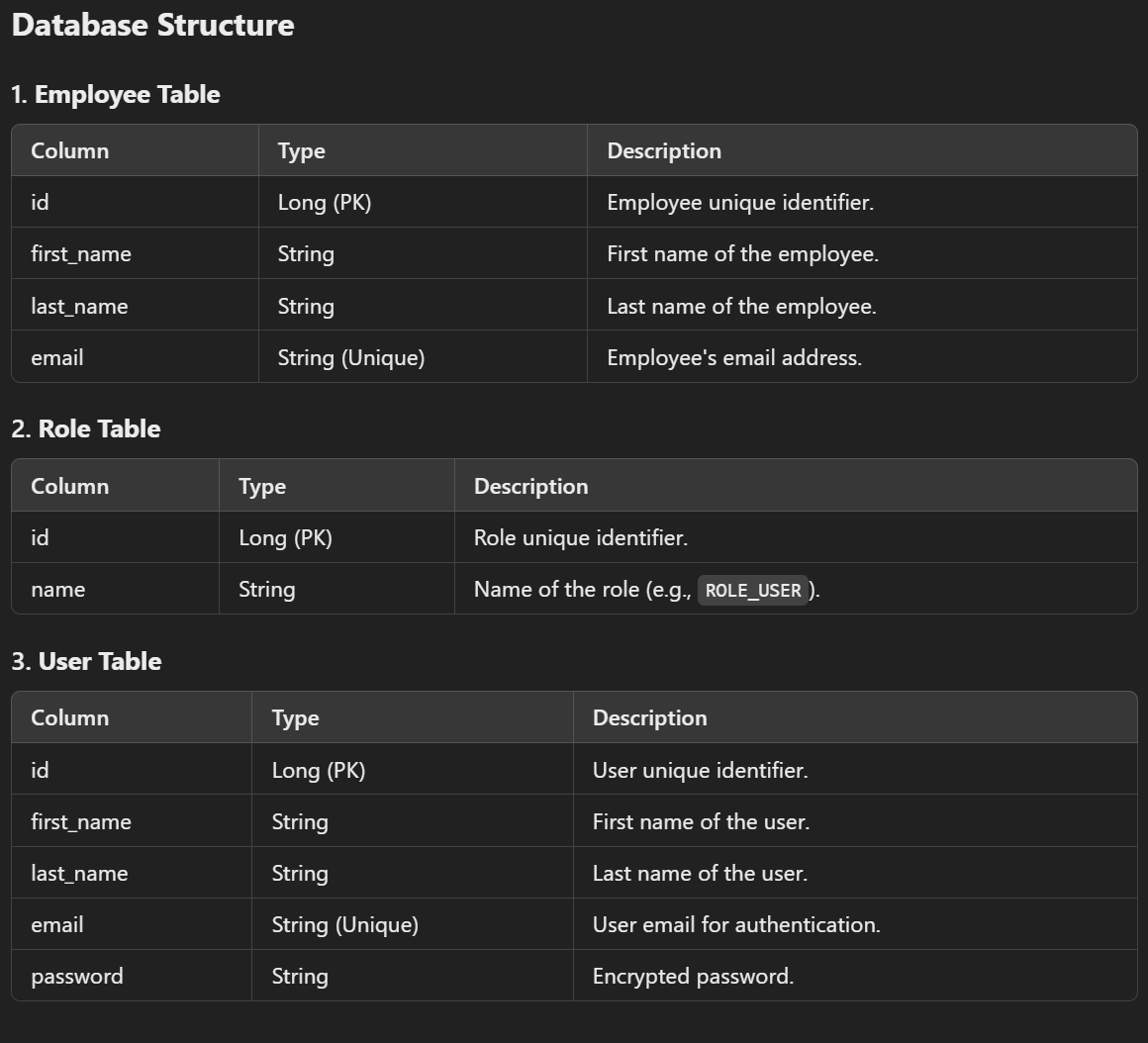
* Clone the project repository.
* Configure the database in the application.properties file.
* Run the application using an IDE or mvn spring-boot:run.

**2. Access the Application**

* Open the browser and navigate to http://localhost:8080.
* Use the registration page to create a new user or login with an existing account.

**3. Managing Employees**

* Add, update, or delete employee records from the dashboard.
* View paginated and sorted employee data.

****

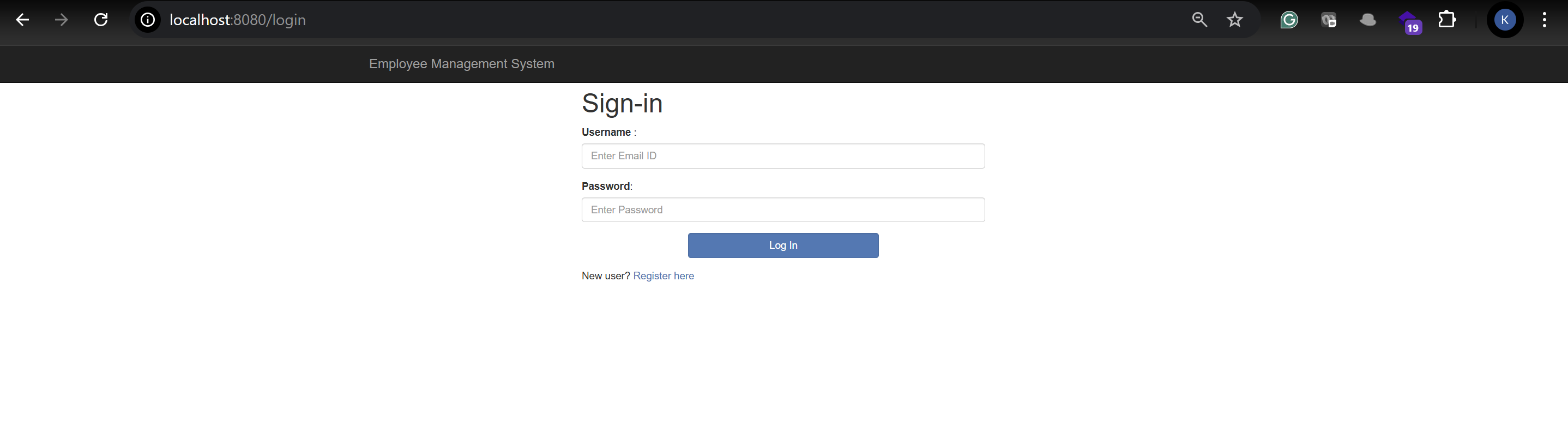
**Controllers**

**EmployeeController**

* Endpoints:
  + GET / - Displays the list of employees with pagination and sorting.
  + GET /showNewEmployeeForm - Renders the form to add a new employee.
  + POST /saveEmployee - Saves a new employee.
  + GET /showFormForUpdate/{id} - Displays a pre-filled form to update an employee.
  + GET /deleteEmployee/{id} - Deletes an employee by ID.
  + GET /page/{pageNo} - Fetches paginated data with sorting parameters.
* Thymeleaf Views:
  + index.html: Displays a list of employees.
  + new\_employee.html: Form to add a new employee.
  + update\_employee.html: Form to update existing employee details.

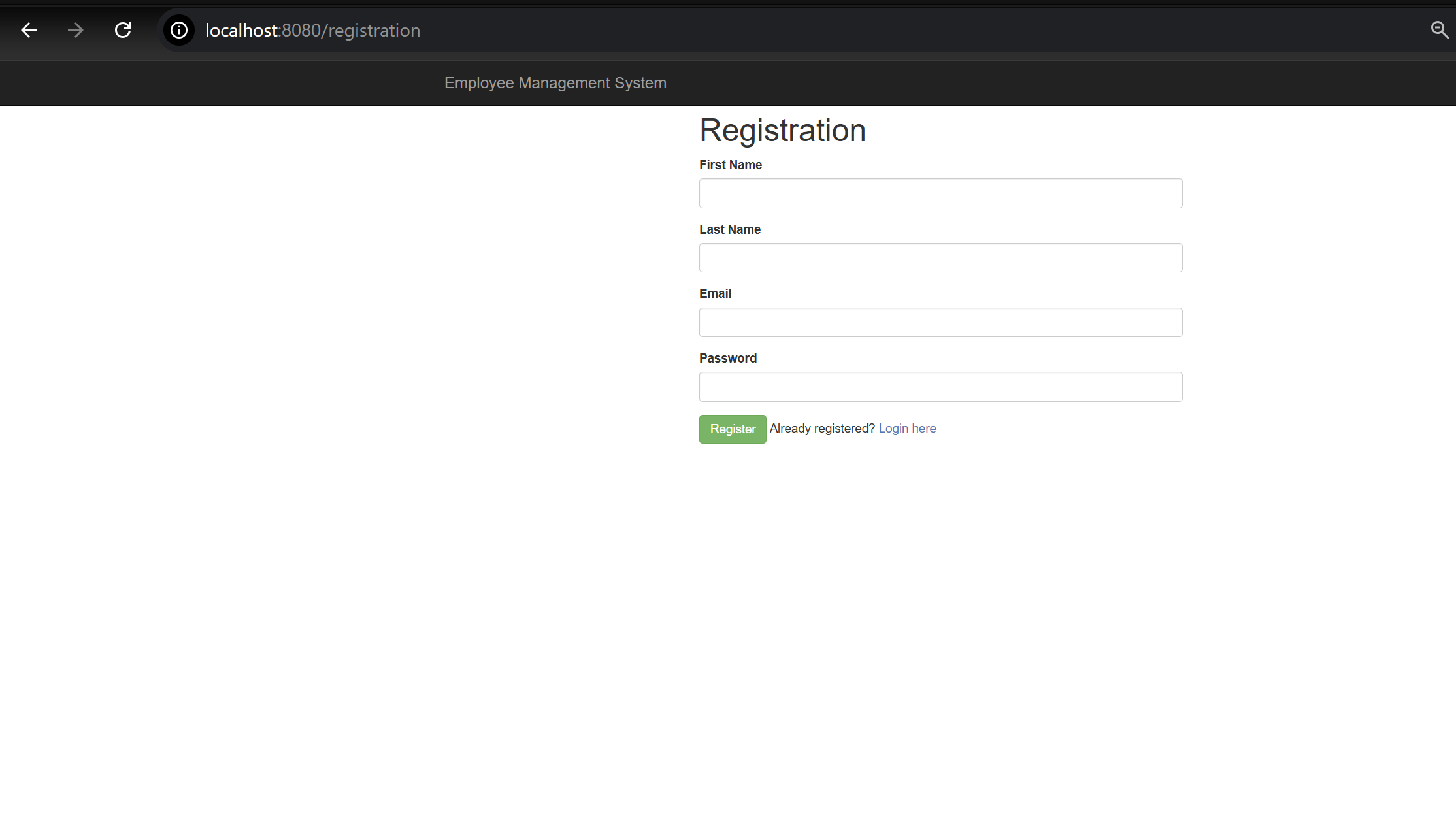
UserRegistrationController

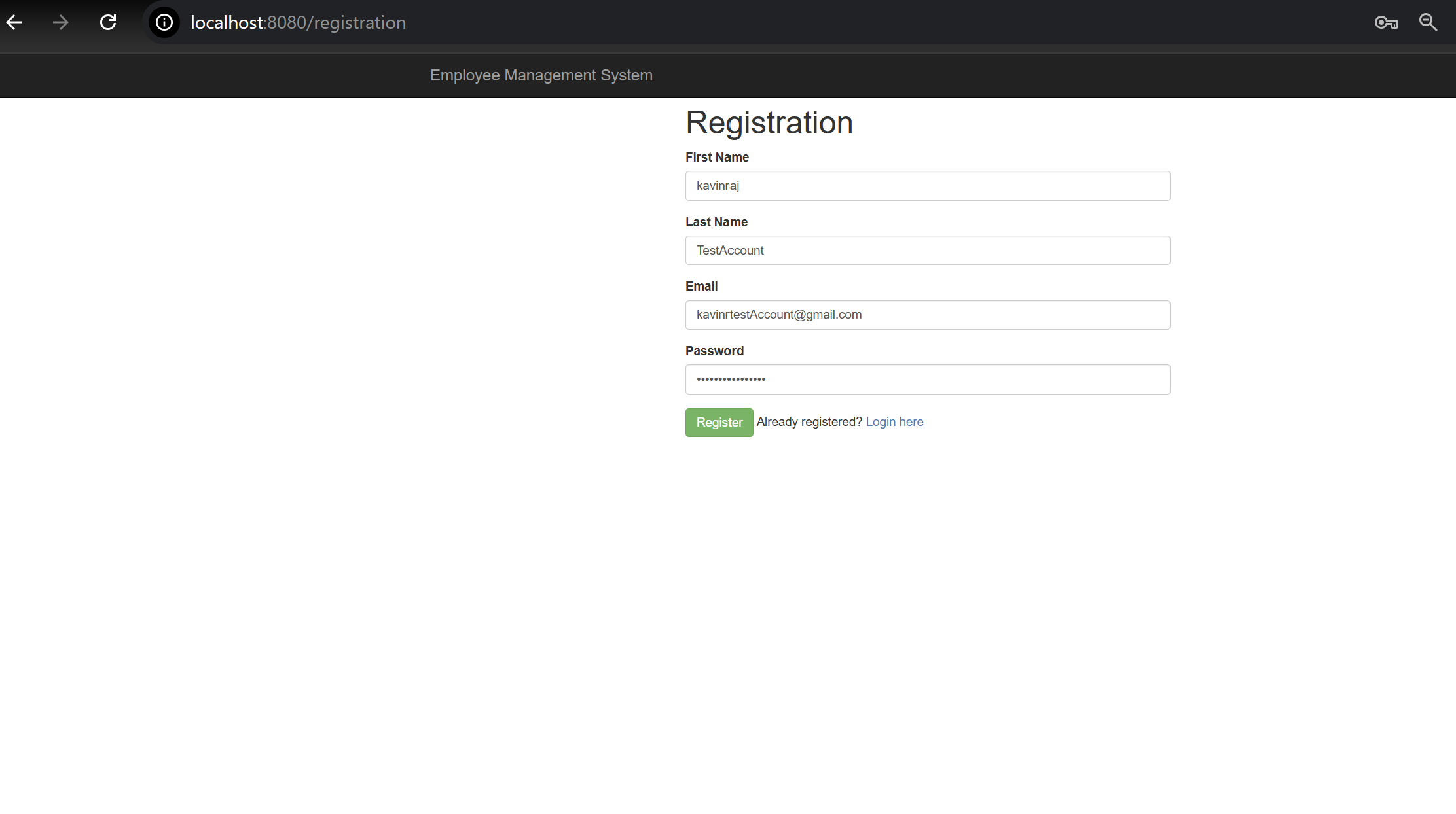
* Endpoints:
  + GET /registration - Renders the user registration form.
  + POST /registration - Handles new user registration.
* Thymeleaf Views:
  + registration.html: Form for new user registration.

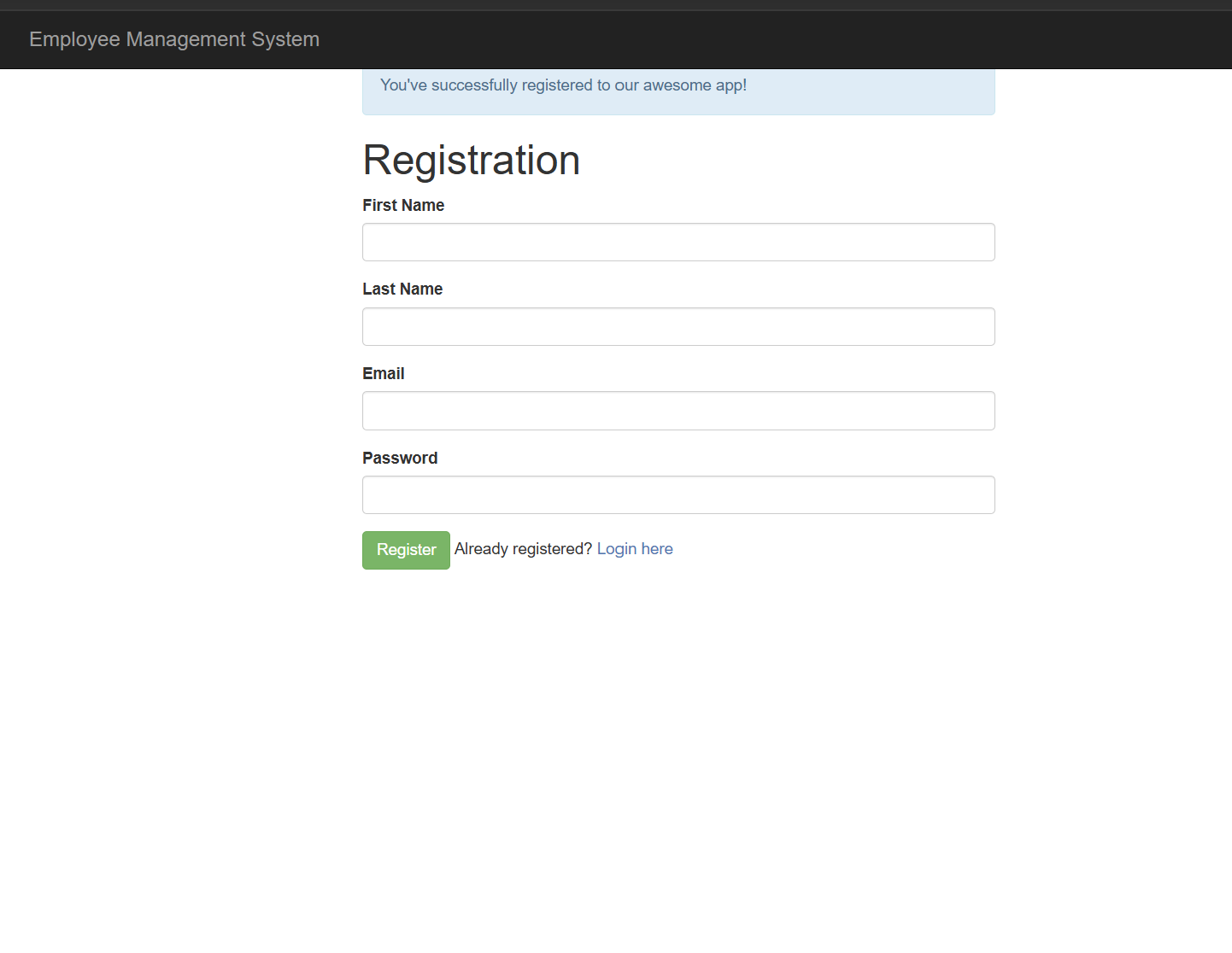
**Landing page  
  
**

**If you already have an account, you can directly provide your credentials and click 'Sign In' to log in.**

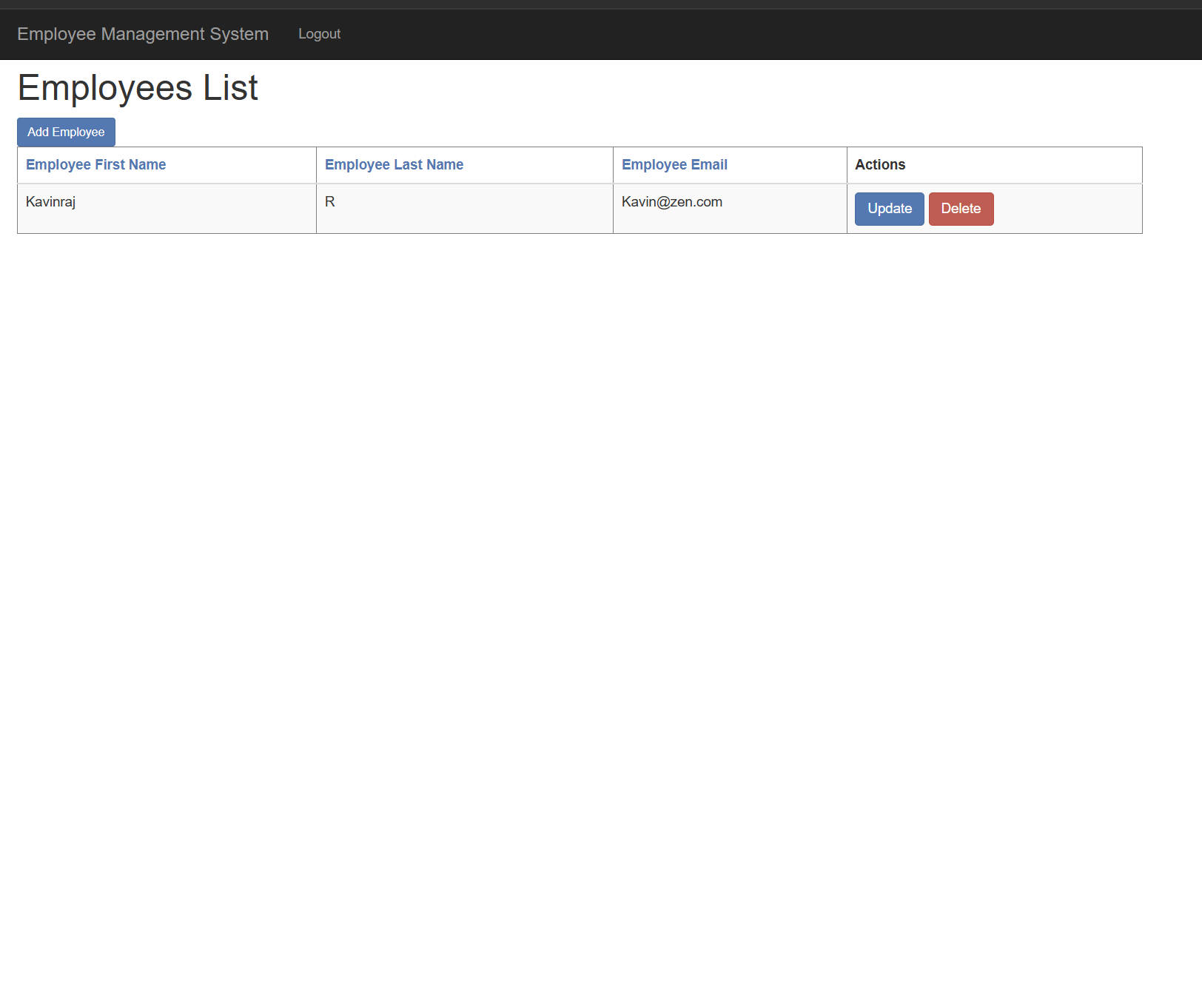
**Else By clicking register here button you can create an account**

****

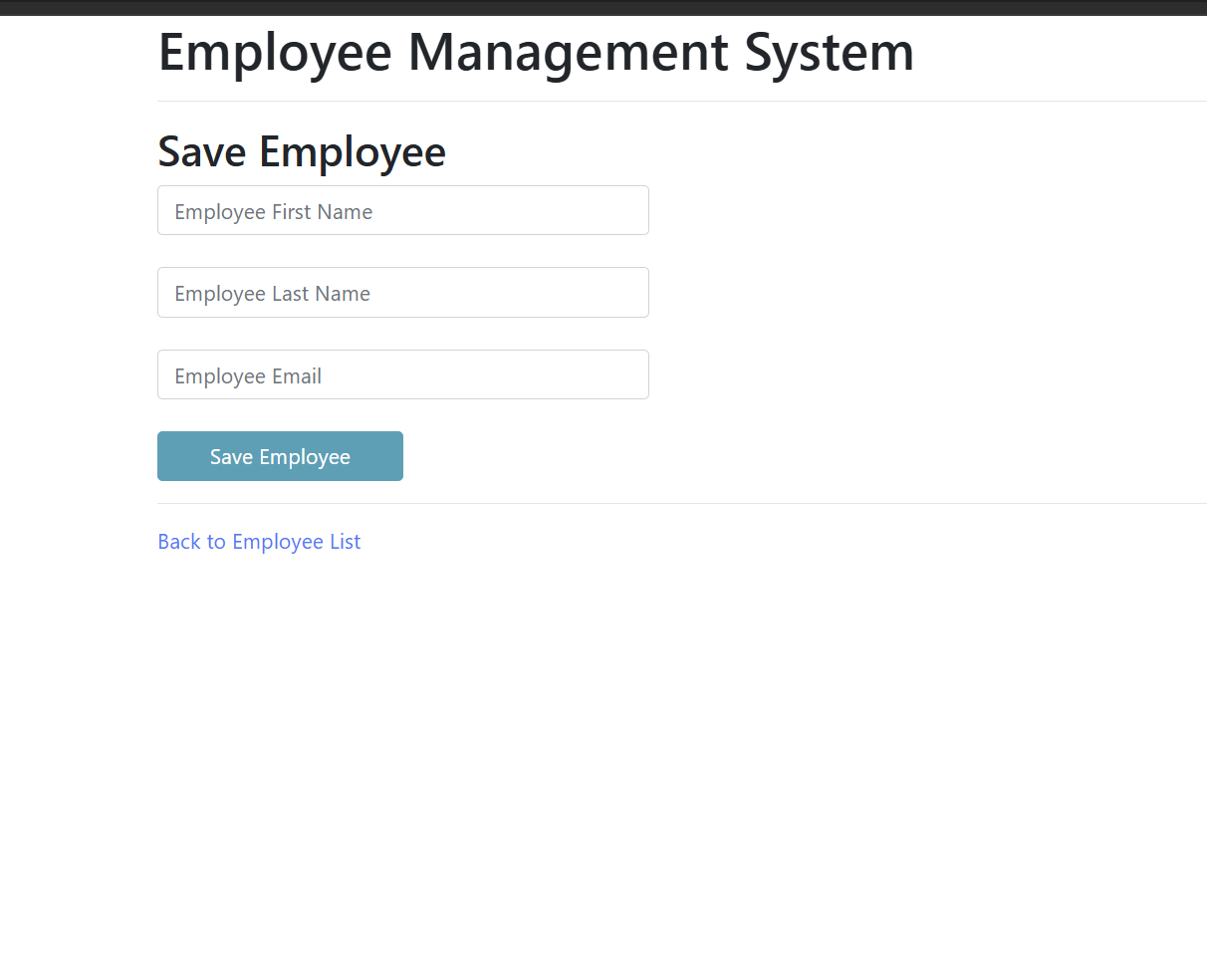
**By providing all the information you can create an account   
  
**

**Once you click the Register button, you will get a popup like the one below then click on Login Here button to log in  
  
**

**Home Page After Login**

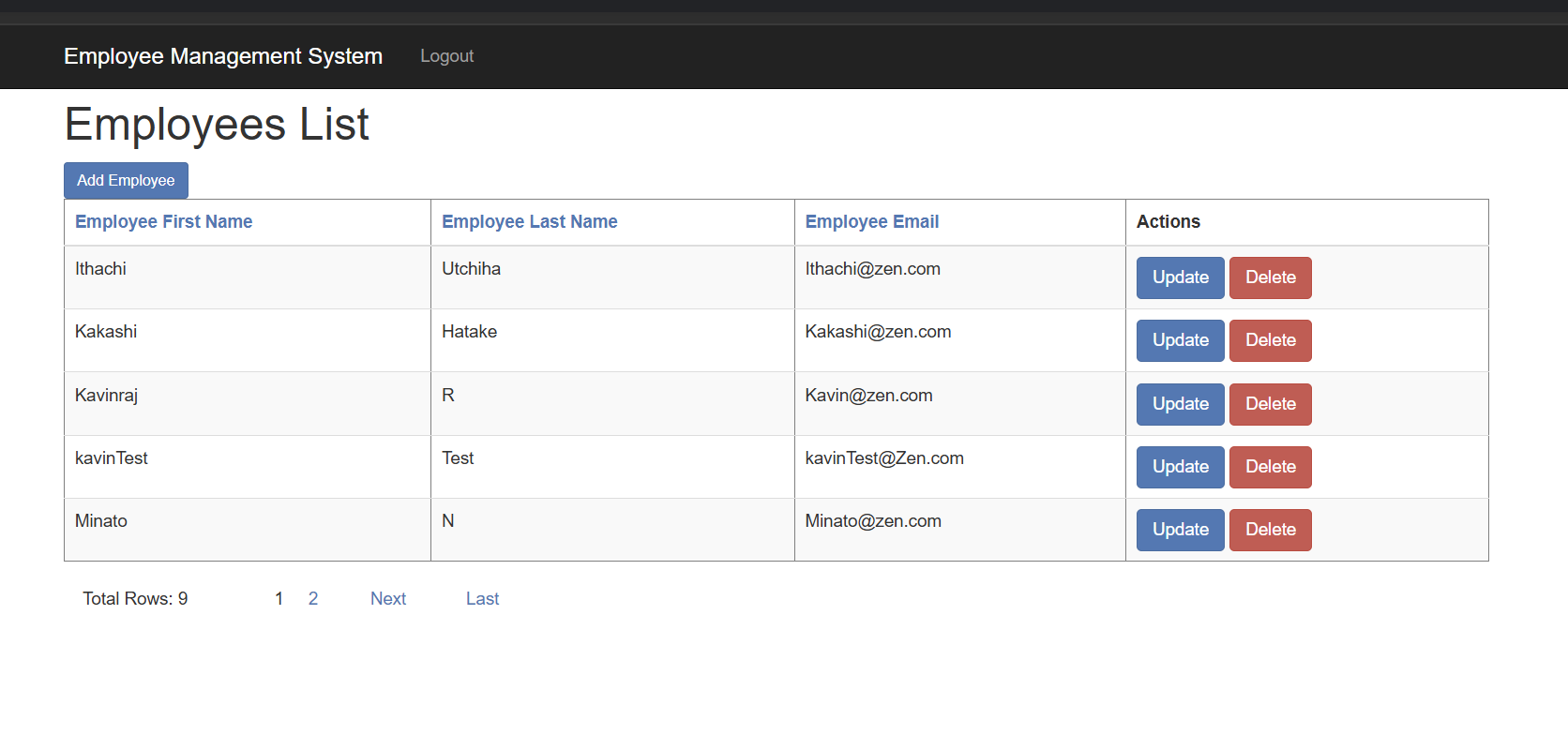
****

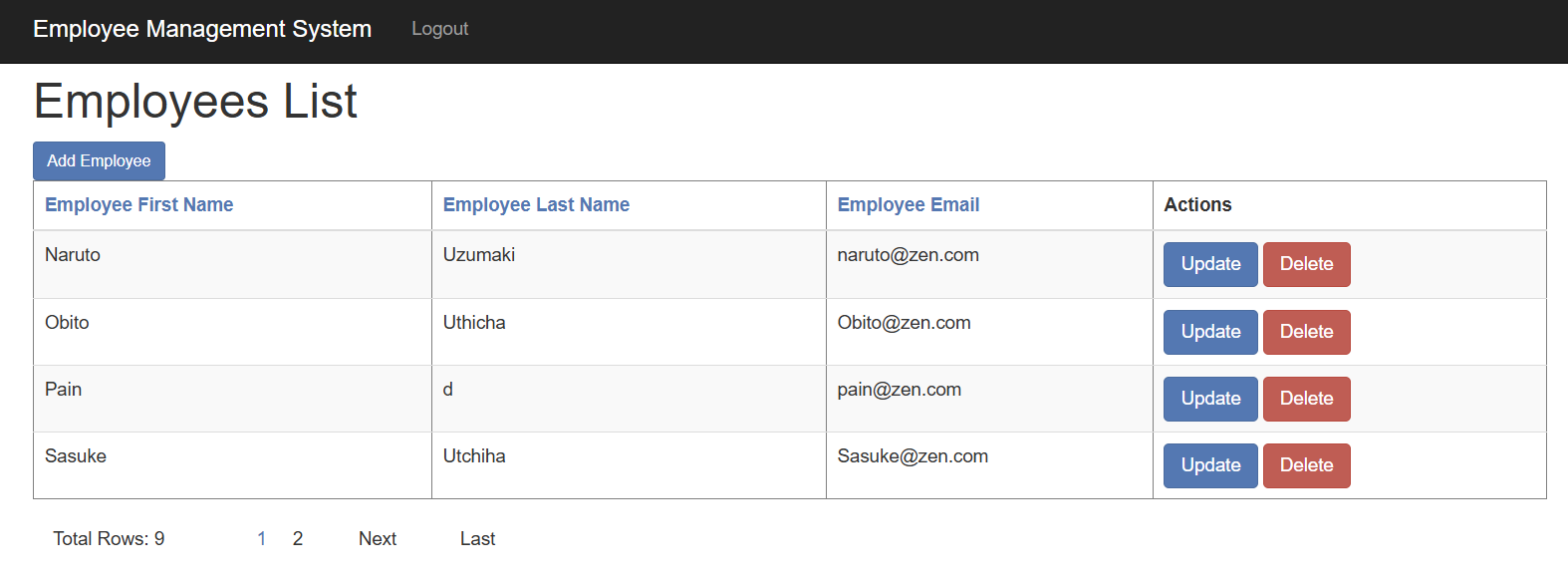
**By clicking add employee button we can add an employee**

****

**Fill in the employee details and click on the save employee button to the same employee details in DB**

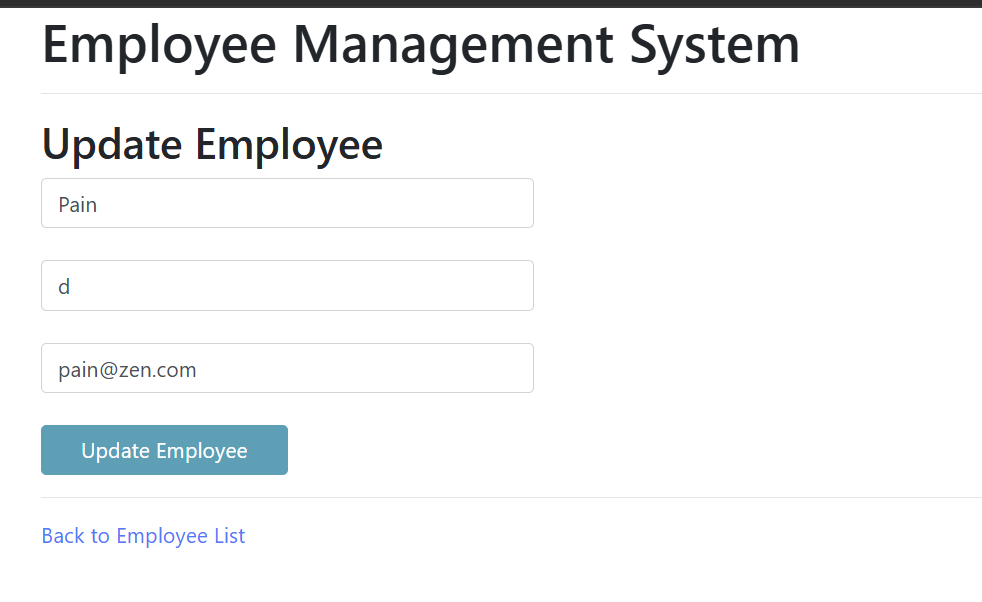
**I have introduced the pagination so that we do not need to scroll infinite times, it’s already sorted in ascending order and based on name we can click on the page number or next button.**

****

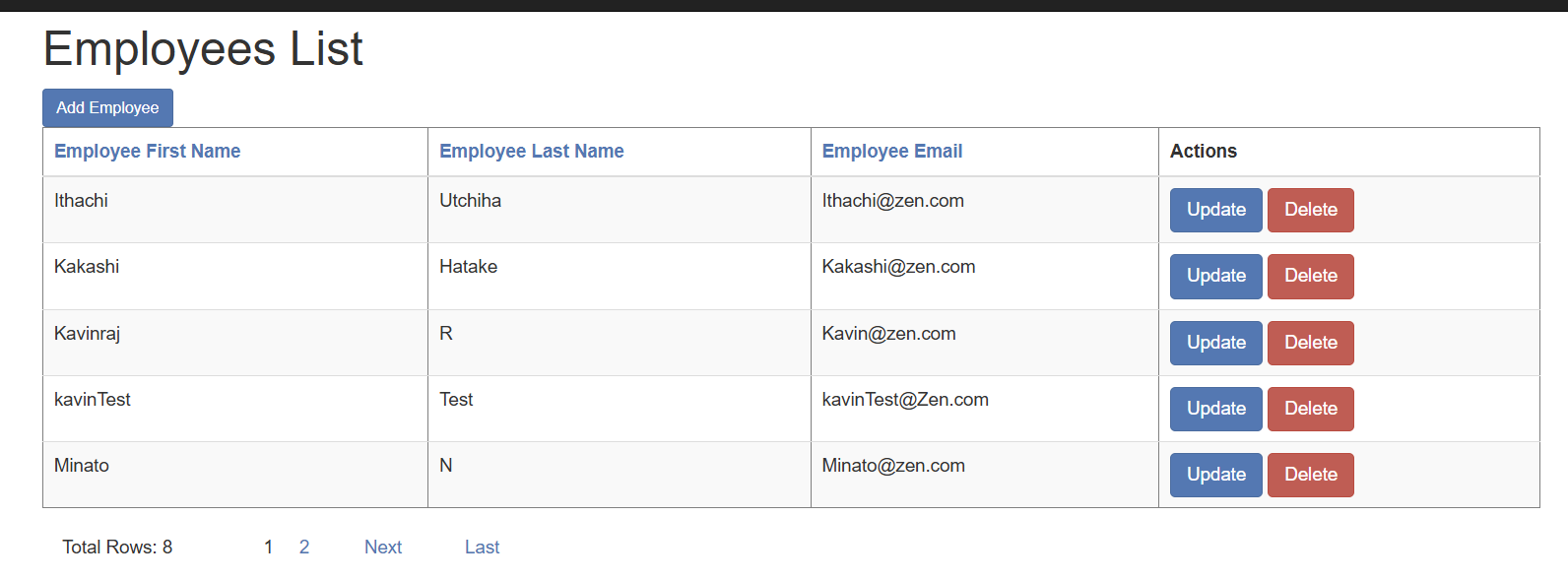
****

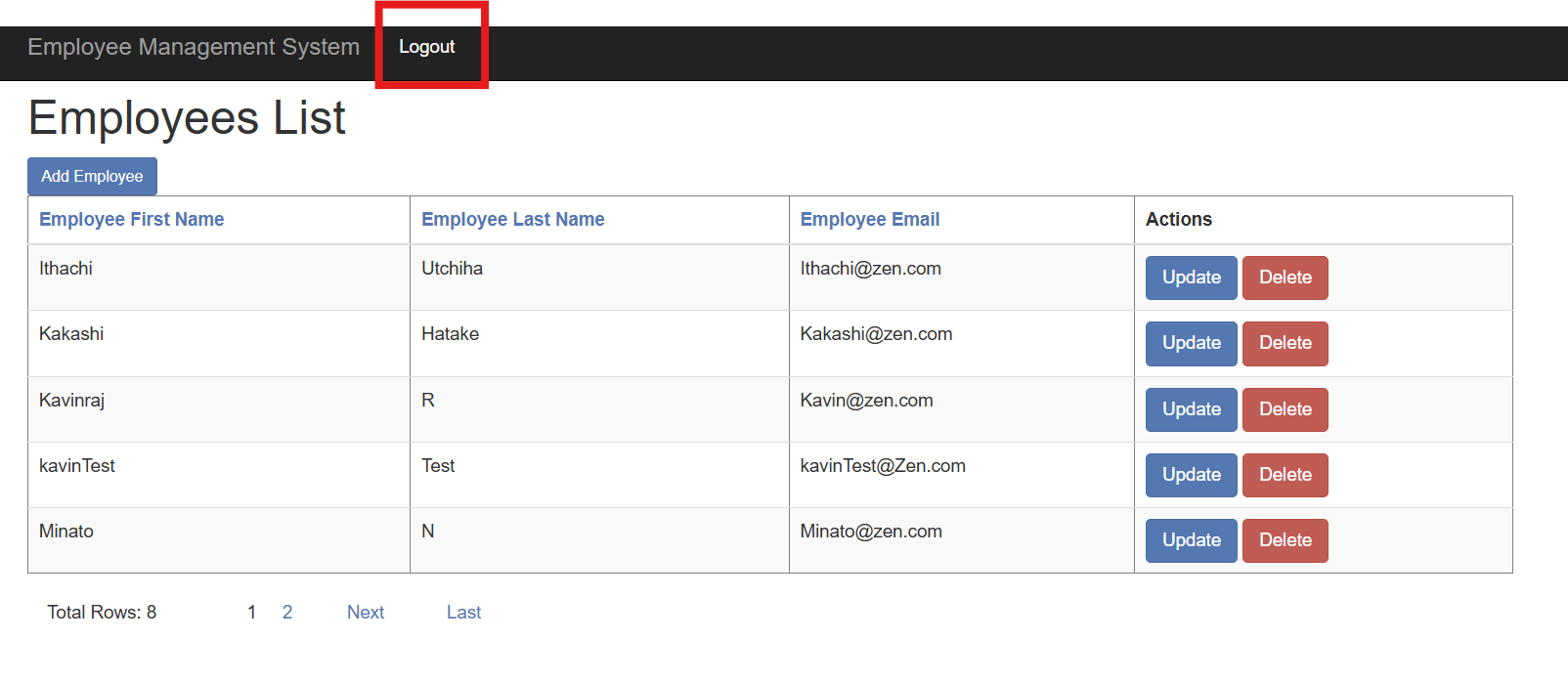
**Update:**

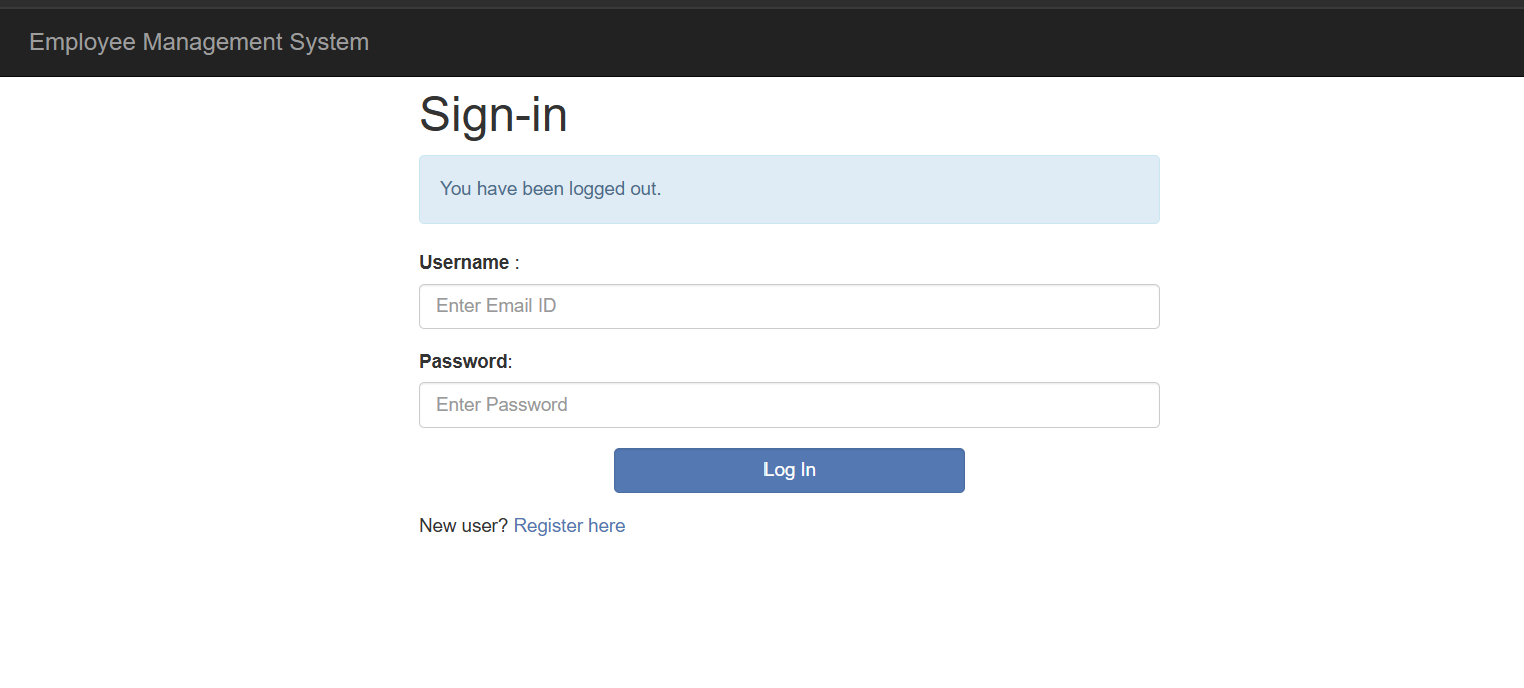
**We can update/ edit the employee details by clicking the update button.**

****

**Delete:  
  
we can delete the employee by clicking the delete button**

**After deleting count became 8 from 9  
  
**

**Logout:  
  
**

****

**Swagger Api details**

**GET / - Display the homepage with a paginated list of employees.**

**GET /showNewEmployeeForm - Display a form to add a new employee.**

**POST /saveEmployee - Save the new employee data.**

**GET /showFormForUpdate/{id} - Display a form for updating an employee's details.**

**GET /deleteEmployee/{id} - Delete an employee by their ID.**

**GET /page/{pageNo} - Fetch paginated employee data with sort functionality.**

**GET /registration: Shows the registration form.**

**POST /registration: Handles the user registration by accepting user data and creating a new account. On successful registration, it redirects to the registration page with a success message (?success).**

**1. Employee Management**

**1.1 View All Employees**

**GET /  
Description: Fetches a paginated and sorted list of employees.**

* **Parameters:**
  + **pageNo (Query, Integer): The page number. Default: 1.**
  + **sortField (Query, String): Field to sort by. Default: firstName.**
  + **sortDir (Query, String): Sort direction (asc or desc). Default: asc.**
* **Response:**
  + **200 OK: List of employees with pagination details.**
  + **Example:**

**JSON**

**CopyEdit**

**{**

**"currentPage": 1,**

**"totalPages": 5,**

**"totalItems": 25,**

**"listEmployees": [**

**{**

**"id": 1,**

**"firstName": "John",**

**"lastName": "Doe",**

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

**},**

**...**

**]**

**}**

**1.2 Add a New Employee**

**GET /showNewEmployeeForm  
Description: Displays the form for adding a new employee.**

**POST /saveEmployee  
Description: Saves a new employee to the database.**

* **Request Body:**

**JSON**

**CopyEdit**

**{**

**"firstName": "John",**

**"lastName": "Doe",**

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

**}**

* **Response:**
  + **302 Redirect: Redirects to /.**

**1.3 Update Employee Details**

**GET /showFormForUpdate/{id}  
Description: Displays the form to update employee details.**

* **Path Parameter:**
  + **id (Long): ID of the employee to update.**
* **Response:**
  + **200 OK: Returns the employee details for the given ID.**

**1.4 Delete an Employee**

**GET /deleteEmployee/{id}  
Description: Deletes an employee by ID.**

* **Path Parameter:**
  + **id (Long): ID of the employee to delete.**
* **Response:**
  + **302 Redirect: Redirects to /.**

**2. User Registration**

**2.1 User Registration Form**

**GET /registration  
Description: Displays the user registration form.**

**2.2 Register a User**

**POST /registration  
Description: Registers a new user.**

* **Request Body:**

**json**

**CopyEdit**

**{**

**"firstName": "Jane",**

**"lastName": "Doe",**

**"email": "jane.doe@example.com",**

**"password": "securePassword123"**

**}**

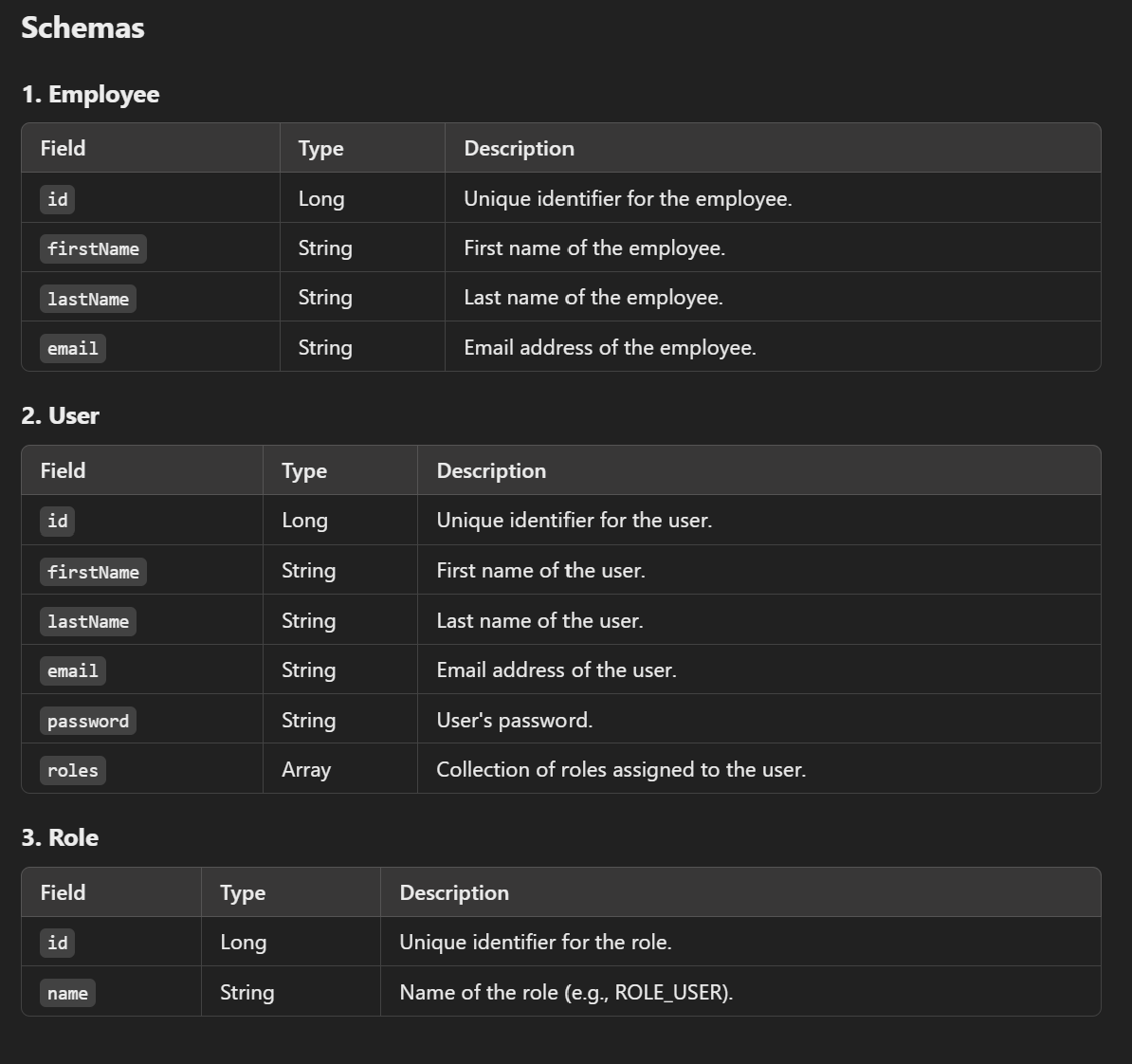
* **Response:**
  + **302 Redirect: Redirects to /registration?success.**

**3. Pagination**

**GET /page/{pageNo}  
Description: Fetches a paginated list of employees.**

* **Path Parameter:**
  + **pageNo (Integer): Page number to retrieve.**
* **Query Parameters:**
  + **sortField (String): Field to sort by. Default: firstName.**
  + **sortDir (String): Sort direction (asc or desc). Default: asc.**
* **Response:**
  + **200 OK: Returns a list of employees with pagination and sorting details.**

**Schemas**

****

**1. Employee**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| **id** | **Long** | **Unique identifier for the employee.** |
| **firstName** | **String** | **First name of the employee.** |
| **lastName** | **String** | **Last name of the employee.** |
| **email** | **String** | **Email address of the employee.** |

**2. User**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| **id** | **Long** | **Unique identifier for the user.** |
| **firstName** | **String** | **First name of the user.** |
| **lastName** | **String** | **Last name of the user.** |
| **email** | **String** | **Email address of the user.** |
| **password** | **String** | **User's password.** |
| **roles** | **Array** | **Collection of roles assigned to the user.** |

**3. Role**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| **id** | **Long** | **Unique identifier for the role.** |
| **name** | **String** | **Name of the role (e.g., ROLE\_USER).** |

**Error Responses**

* **404 Not Found: When an employee or user is not found.**
  + **Example:**

**JSON**

**CopyEdit**

**{**

**"timestamp": "2025-01-26T12:00:00Z",**

**"status": 404,**

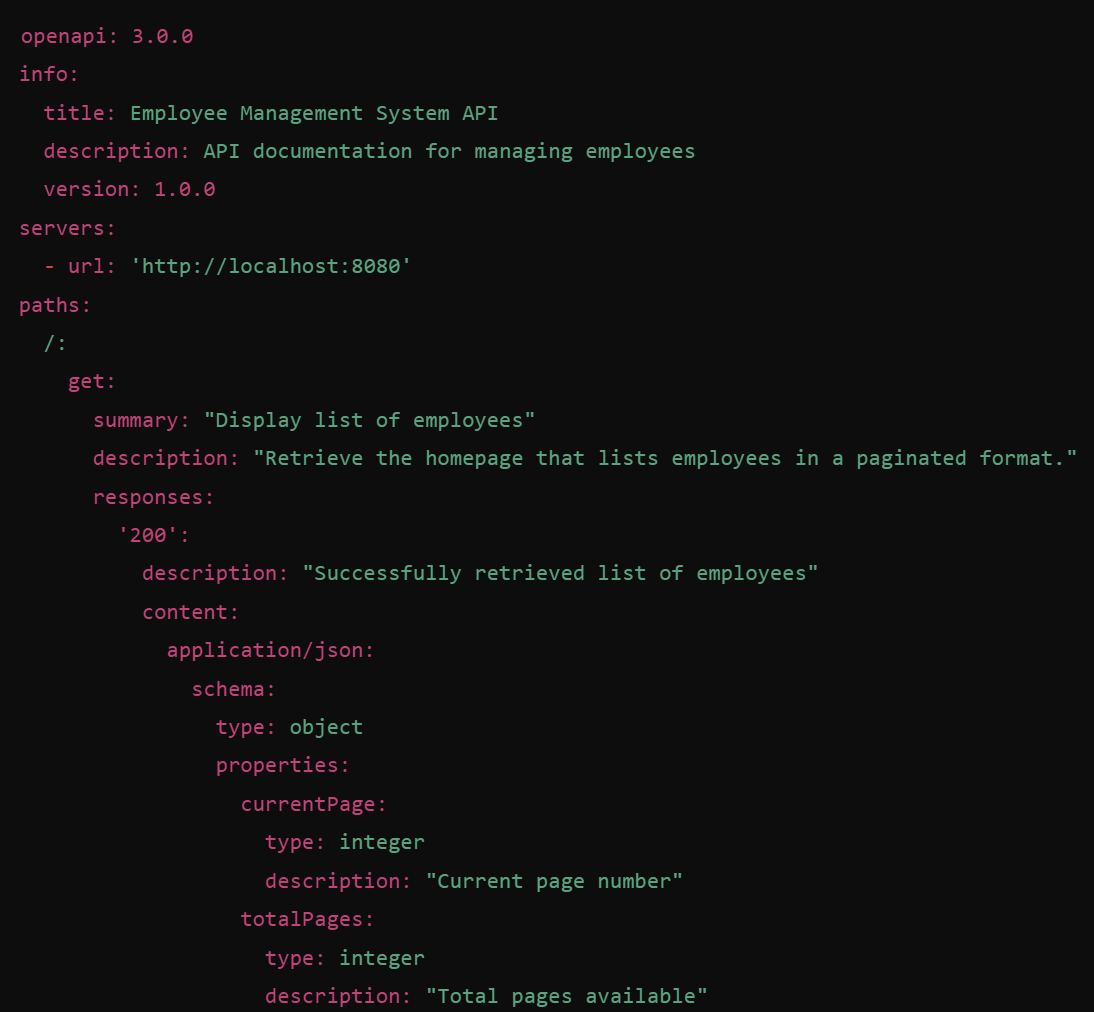
**"error": "Not Found",**

**"message": "Employee not found for id :: 1",**

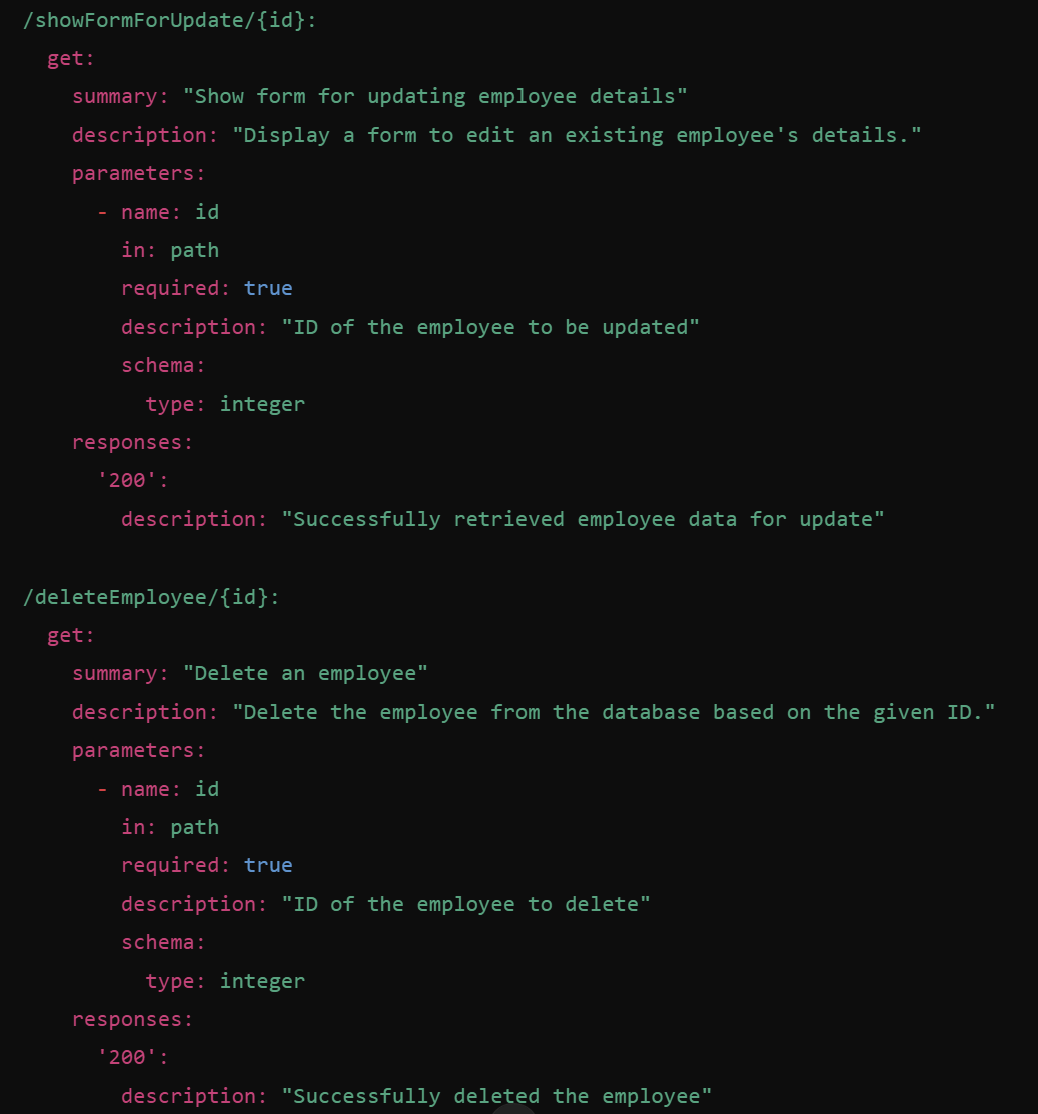
**"path": "/showFormForUpdate/1"**

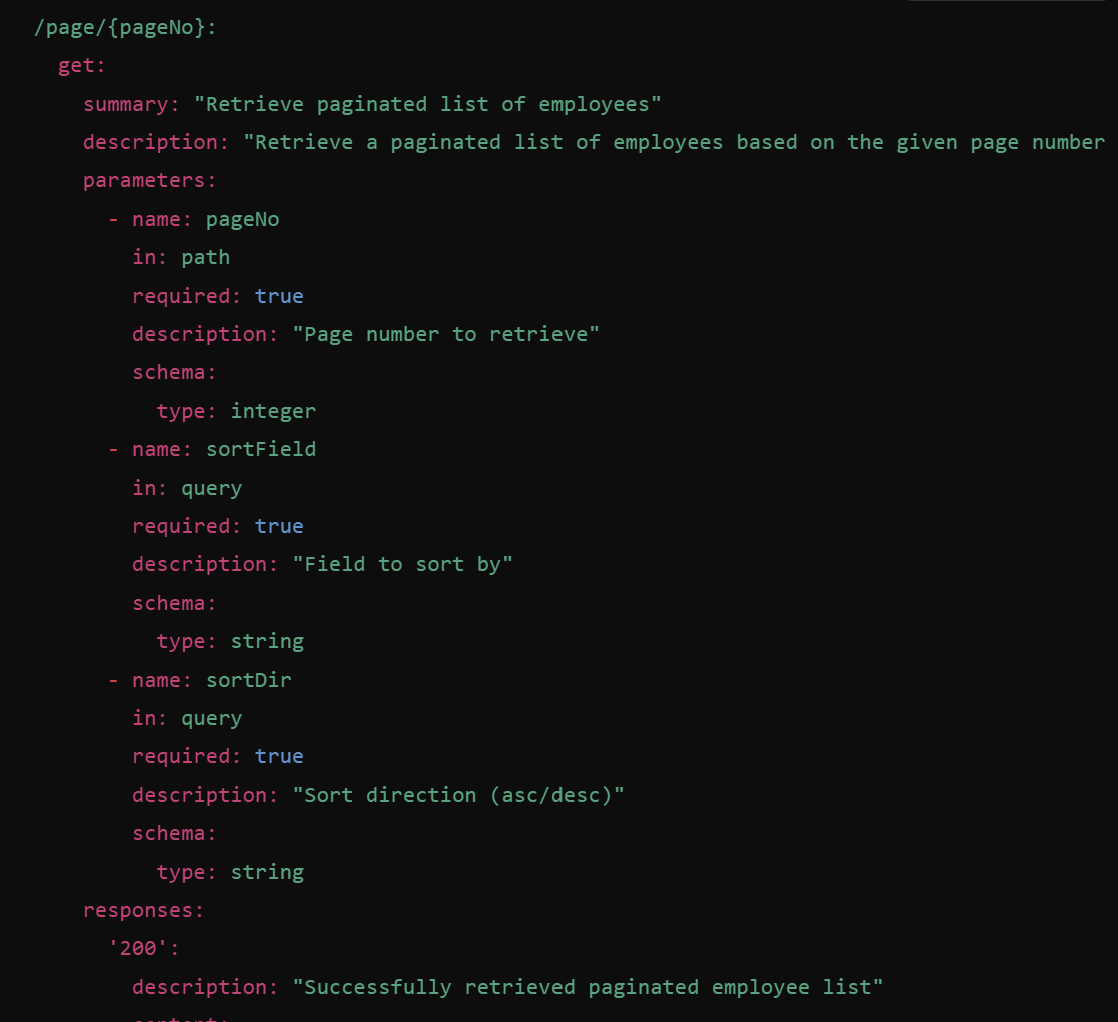
**}**

* **400 Bad Request: For invalid input data.**
* **500 Internal Server Error: For unexpected server issues.**

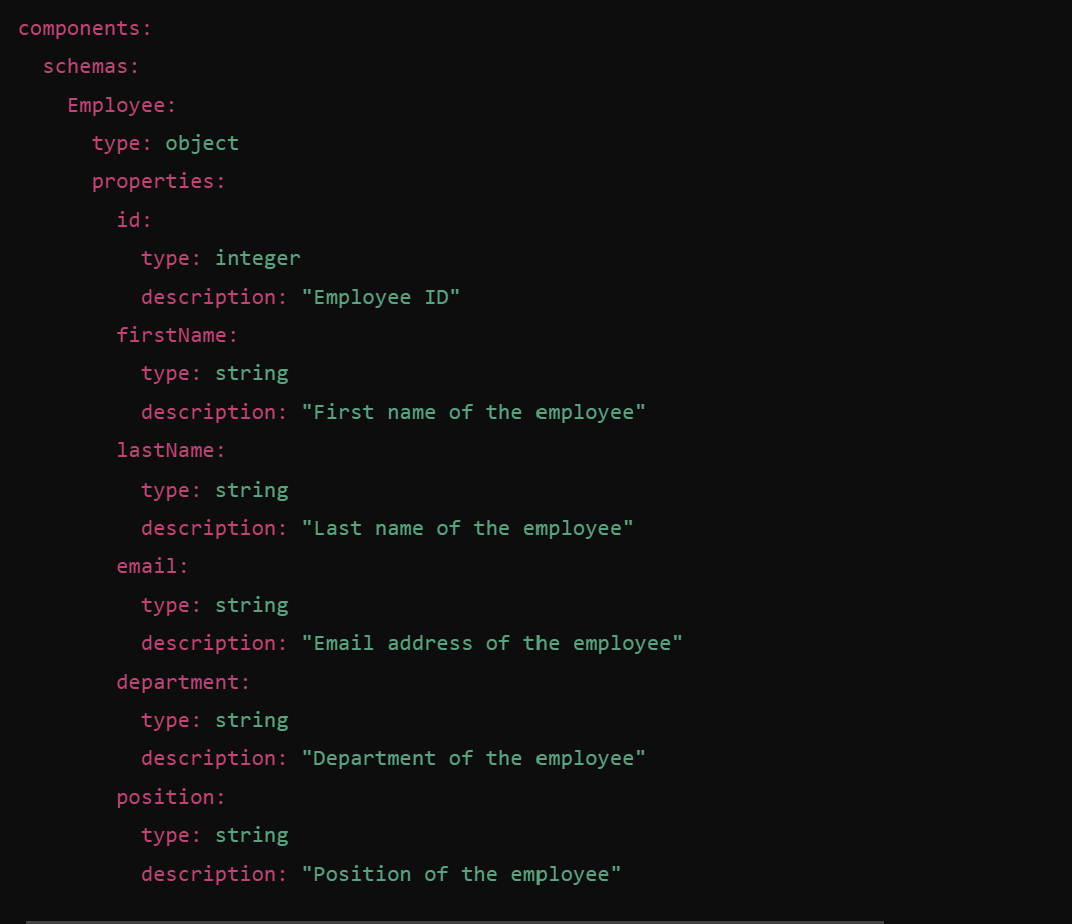
****

****

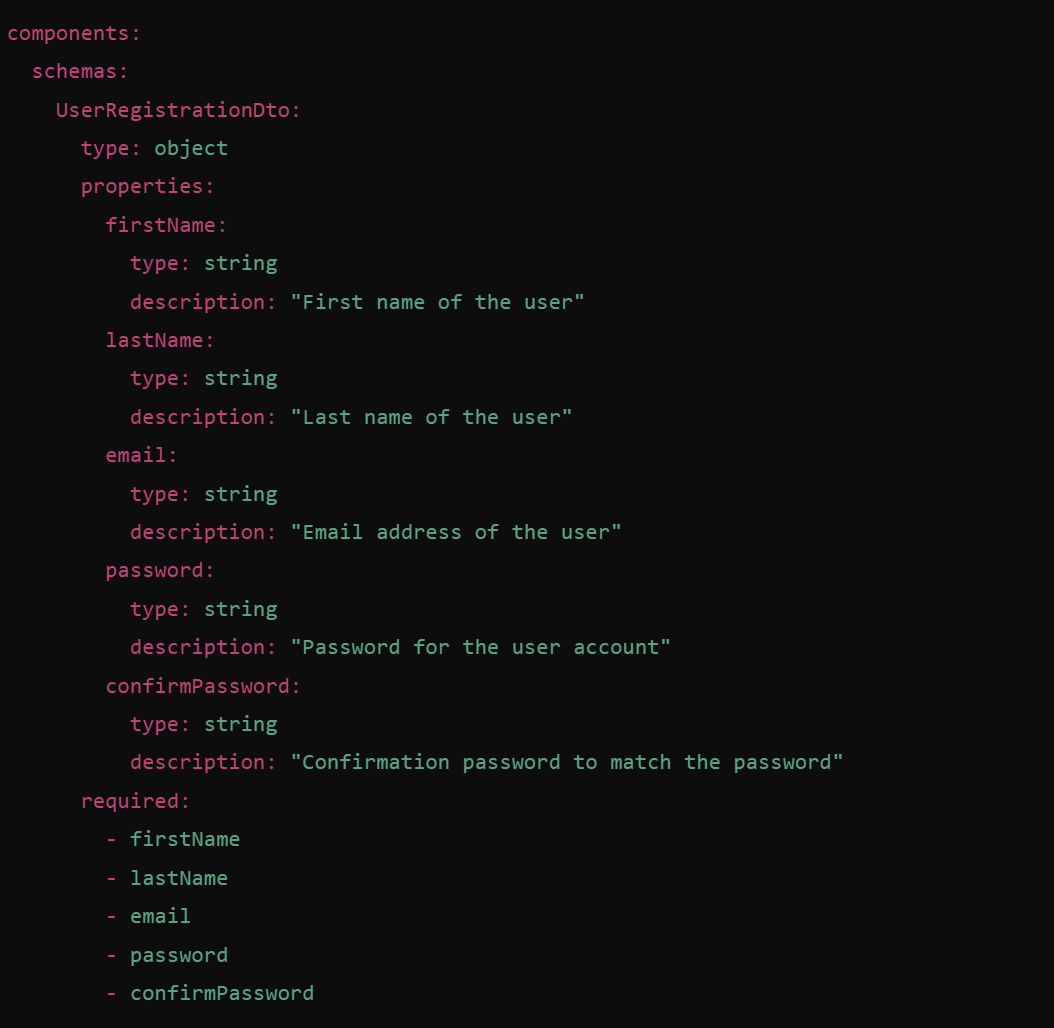
****

****

****

****

****

****

**openapi: 3.0.0**

**info:**

**title: Employee Management System API**

**description: API documentation for managing employees**

**version: 1.0.0**

**servers:**

**- url: 'http://localhost:8080'**

**paths:**

**/:**

**get:**

**summary: "Display list of employees"**

**description: "Retrieve the homepage that lists employees in a paginated format."**

**responses:**

**'200':**

**description: "Successfully retrieved list of employees"**

**content:**

**application/json:**

**schema:**

**type: object**

**properties:**

**currentPage:**

**type: integer**

**description: "Current page number"**

**totalPages:**

**type: integer**

**description: "Total pages available"**

**totalItems:**

**type: integer**

**description: "Total number of items"**

**listEmployees:**

**type: array**

**items:**

**$ref: '#/components/schemas/Employee'**

**/showNewEmployeeForm:**

**get:**

**summary: "Show form for adding a new employee"**

**description: "Displays the form to add a new employee."**

**responses:**

**'200':**

**description: "Successfully retrieved the employee form"**

**/saveEmployee:**

**post:**

**summary: "Save a new employee"**

**description: "Create a new employee and save it to the database."**

**requestBody:**

**required: true**

**content:**

**application/json:**

**schema:**

**$ref: '#/components/schemas/Employee'**

**responses:**

**'200':**

**description: "Successfully saved the employee"**

**/showFormForUpdate/{id}:**

**get:**

**summary: "Show form for updating employee details"**

**description: "Display a form to edit an existing employee's details."**

**parameters:**

**- name: id**

**in: path**

**required: true**

**description: "ID of the employee to be updated"**

**schema:**

**type: integer**

**responses:**

**'200':**

**description: "Successfully retrieved employee data for update"**

**/deleteEmployee/{id}:**

**get:**

**summary: "Delete an employee"**

**description: "Delete the employee from the database based on the given ID."**

**parameters:**

**- name: id**

**in: path**

**required: true**

**description: "ID of the employee to delete"**

**schema:**

**type: integer**

**responses:**

**'200':**

**description: "Successfully deleted the employee"**

**/page/{pageNo}:**

**get:**

**summary: "Retrieve paginated list of employees"**

**description: "Retrieve a paginated list of employees based on the given page number and sort parameters."**

**parameters:**

**- name: pageNo**

**in: path**

**required: true**

**description: "Page number to retrieve"**

**schema:**

**type: integer**

**- name: sortField**

**in: query**

**required: true**

**description: "Field to sort by"**

**schema:**

**type: string**

**- name: sortDir**

**in: query**

**required: true**

**description: "Sort direction (asc/desc)"**

**schema:**

**type: string**

**responses:**

**'200':**

**description: "Successfully retrieved paginated employee list"**

**content:**

**application/json:**

**schema:**

**type: object**

**properties:**

**currentPage:**

**type: integer**

**totalPages:**

**type: integer**

**totalItems:**

**type: integer**

**listEmployees:**

**type: array**

**items:**

**$ref: '#/components/schemas/Employee'**

**components:**

**schemas:**

**Employee:**

**type: object**

**properties:**

**id:**

**type: integer**

**description: "Employee ID"**

**firstName:**

**type: string**

**description: "First name of the employee"**

**lastName:**

**type: string**

**description: "Last name of the employee"**

**email:**

**type: string**

**description: "Email address of the employee"**

**department:**

**type: string**

**description: "Department of the employee"**

**position:**

**type: string**

**description: "Position of the employee"**

**openapi: 3.0.0**

**info:**

**title: User Registration API**

**description: API documentation for user registration and account creation.**

**version: 1.0.0**

**servers:**

**- url: 'http://localhost:8080'**

**paths:**

**/registration:**

**get:**

**summary: "Show registration form"**

**description: "Display the form to register a new user."**

**responses:**

**'200':**

**description: "Successfully retrieved the registration form"**

**post:**

**summary: "Register new user"**

**description: "Register a new user account using the provided registration data."**

**requestBody:**

**required: true**

**content:**

**application/json:**

**schema:**

**$ref: '#/components/schemas/UserRegistrationDto'**

**responses:**

**'200':**

**description: "User successfully registered"**

**'302':**

**description: "Redirected to the registration page with success message"**

**components:**

**schemas:**

**UserRegistrationDto:**

**type: object**

**properties:**

**firstName:**

**type: string**

**description: "First name of the user"**

**lastName:**

**type: string**

**description: "Last name of the user"**

**email:**

**type: string**

**description: "Email address of the user"**

**password:**

**type: string**

**description: "Password for the user account"**

**confirmPassword:**

**type: string**

**description: "Confirmation password to match the password"**

**required:**

**- firstName**

**- lastName**

**- email**

**- password**

**- confirmPassword**

**Schema Structure**

**Schema Name: Employee**

**Fields:**

* **id: Auto-generated unique identifier for the employee.**
* **firstName: First name of the employee (required).**
* **lastName: Last name of the employee (required).**
* **email: Email address of the employee (required, in email format).**

****

**components:**

**schemas:**

**Employee:**

**type: object**

**properties:**

**id:**

**type: integer**

**format: int64**

**description: Unique identifier for the employee.**

**firstName:**

**type: string**

**description: The first name of the employee.**

**lastName:**

**type: string**

**description: The last name of the employee.**

**email:**

**type: string**

**format: email**

**description: The email address of the employee.**

**required:**

**- firstName**

**- lastName**

**- email**

**example:**

**id: 1**

**firstName: John**

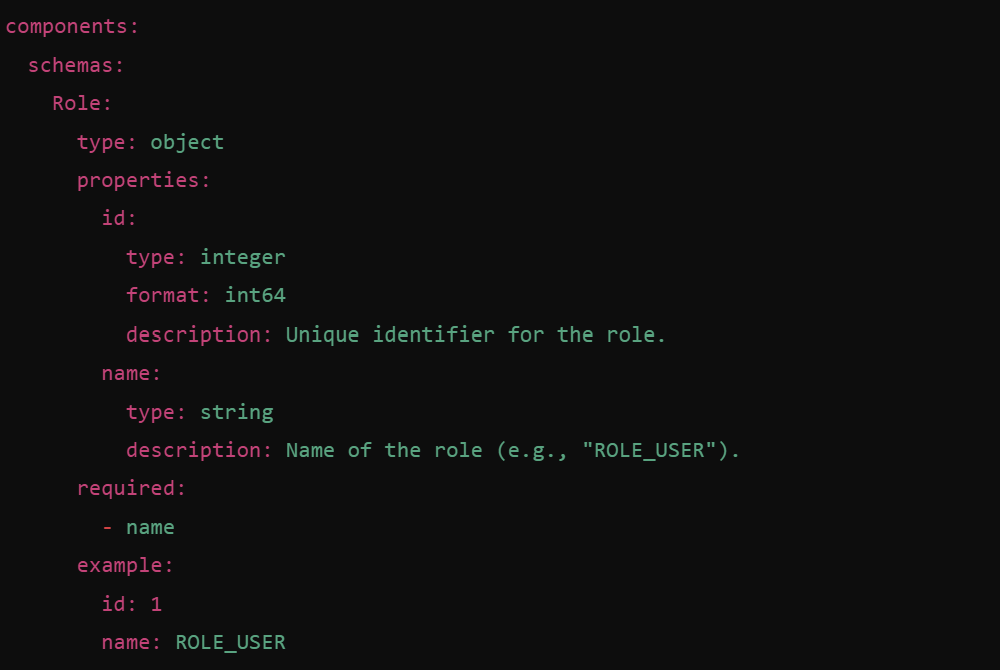
**lastName: Doe**

**email:** [**john.doe@example.com**](mailto:john.doe@example.com)

**Schema Name: Role**

**Fields:**

* **id: Auto-generated unique identifier for the role.**
* **name: Name of the role (e.g., "ROLE\_USER", "ROLE\_ADMIN") (required).**

****

**components:**

**schemas:**

**Role:**

**type: object**

**properties:**

**id:**

**type: integer**

**format: int64**

**description: Unique identifier for the role.**

**name:**

**type: string**

**description: Name of the role (e.g., "ROLE\_USER").**

**required:**

**- name**

**example:**

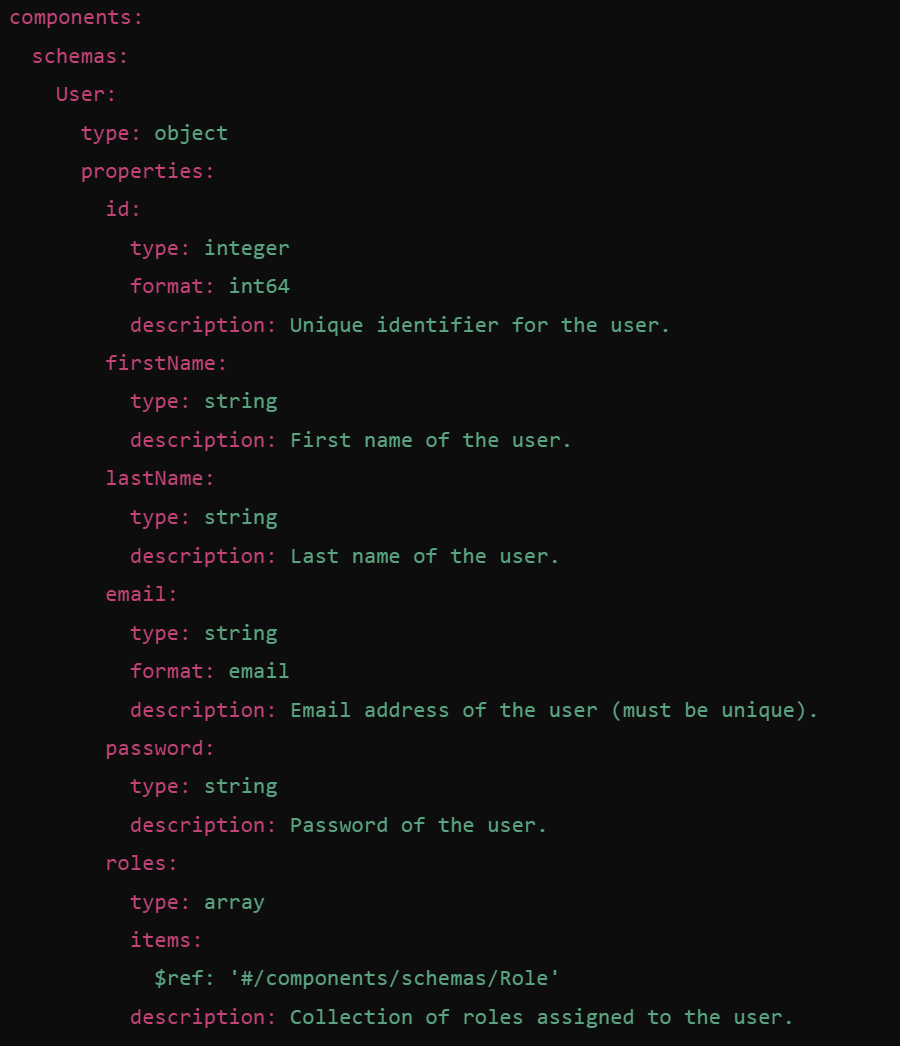
**id: 1**

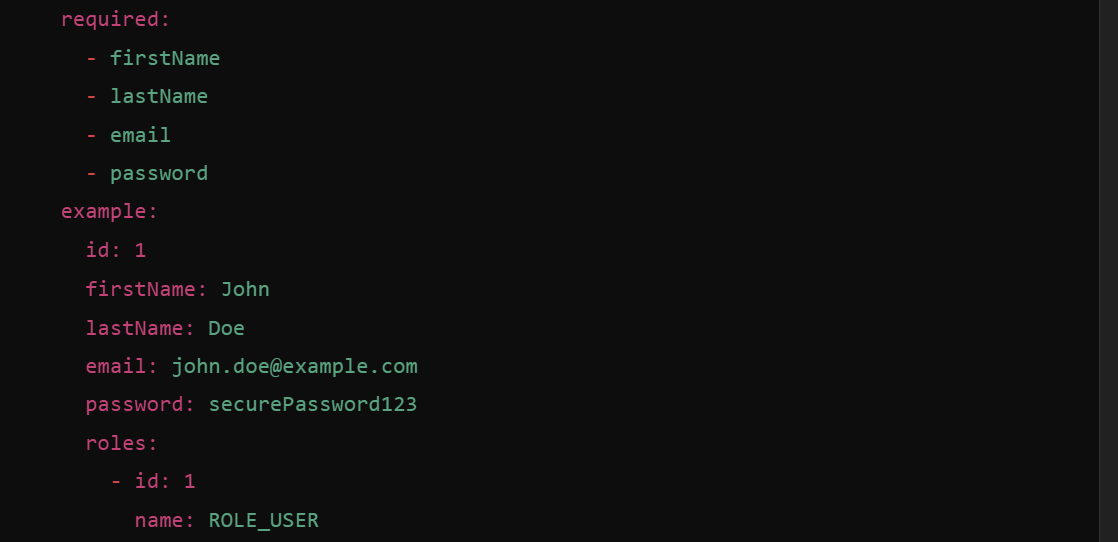
**name: ROLE\_USER**

**Schema Name: User**

**Fields:**

* **id: Auto-generated unique identifier for the user.**
* **firstName: First name of the user (required).**
* **lastName: Last name of the user (required).**
* **email: Email address (must be unique and required).**
* **password: Password for the user's account (required).**
* **roles: Array of roles assigned to the user, referencing the Role schema.**

****

****

**components:**

**schemas:**

**User:**

**type: object**

**properties:**

**id:**

**type: integer**

**format: int64**

**description: Unique identifier for the user.**

**firstName:**

**type: string**

**description: First name of the user.**

**lastName:**

**type: string**

**description: Last name of the user.**

**email:**

**type: string**

**format: email**

**description: Email address of the user (must be unique).**

**password:**

**type: string**

**description: Password of the user.**

**roles:**

**type: array**

**items:**

**$ref: '#/components/schemas/Role'**

**description: Collection of roles assigned to the user.**

**required:**

**- firstName**

**- lastName**

**- email**

**- password**

**example:**

**id: 1**

**firstName: John**

**lastName: Doe**

**email: john.doe@example.com**

**password: securePassword123**

**roles:**

**- id: 1**

**name: ROLE\_USER**

**Conclusion**

This Employee Management System serves as a robust foundation for managing employee data in an organization. Its modular design allows for easy scalability and integration with other services. The application leverages Spring Boot for seamless development, Spring Security for authentication, and Thymeleaf for dynamic HTML content rendering.