

Employee Management System (EMS)

1. Project Setup

Backend (Spring Boot):

Requirements:

- - Java 17 or 21+
- - Maven
- - MySQL Server
- - Postman (for API testing)
- - IDE (like IntelliJ IDEA or Eclipse)

Steps:

1. Clone the Project or Download the Code.
2. Open in IDE (IntelliJ IDEA or Eclipse).
3. Setup application.properties file:

```
spring.datasource.url=jdbc:mysql://localhost:3306/ems
spring.datasource.username=root
spring.datasource.password=yasin1986
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
server.port=8080
```

4. Add Backend Dependencies in (pom.xml file):

Spring Web – For building REST APIs

Spring Data JPA – For database operations using JPA/Hibernate

Spring Boot Validation – For validating user input (e.g., @Email, @NotBlank)

MySQL Driver – For connecting to the MySQL database

Lombok – To reduce boilerplate code like getters/setters

Spring Boot DevTools – For hot-reloading during development

5. Install Maven dependencies:

mvn clean install

6. Run the Spring Boot application:

mvn spring-boot:run

Frontend (HTML, TailwindCSS, JavaScript):

1. Open frontend folder in VS Code.
2. Run Live Server Extension.
3. Frontend will run at <http://127.0.0.1:5500/>

2. ☐ API Endpoints

Method	Endpoint	Description
GET	/api/employee/	Home Page Message
POST	/api/employee/add	Add New Employee
GET	/api/employee/displayAll	Get All Employees
GET	/api/employee/display/{id}	Get Employee by ID
PUT	/api/employee/update/{id}	Update Employee by ID
DELETE	/api/employee/delete/{id}	Delete Employee by ID

3. API Request and Response Format

Example for POST /api/employee/add Request Body:

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "role": "Software Engineer",
  "salary": 75000
}
```

Example for successful response:

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com",
  "role": "Software Engineer",
  "salary": 75000
}
```

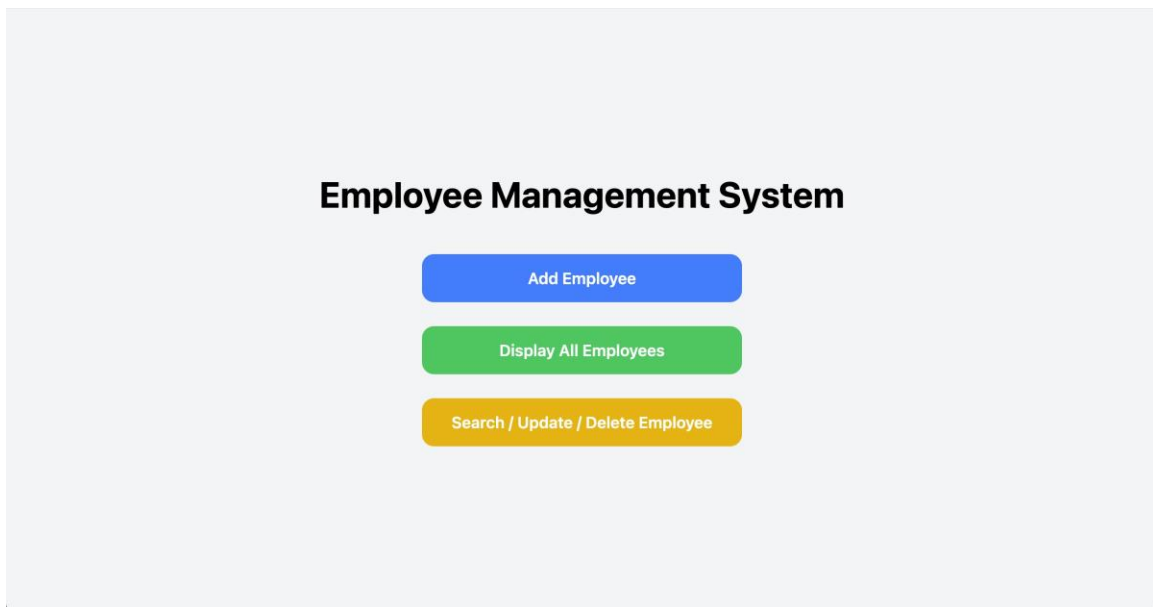
4. Data Validation Rules

Field	Validation	Error Message
name	Not Blank	Name is Mandatory
email	Not Blank, Valid Email Format	Email should be valid
role	Not Blank	Role is Mandatory
salary	Must be Positive	Salary must be positive

5. Screenshots

Frontend Screenshots:

1.HOME PAGE:



2. ADD EMPLOYEE:

Add Employee

Name

Email

Role

Salary

Employee added successfully!

3. ADD EMPLOYEE VALIDATION:

Add Employee

Name

Email

Email should be valid and must contain a domain (e.g., '.com')

Role

Salary

4. STORED IN A DATABASE:

All Employees

ID	Name	Email	Role	Salary
1	MOHAMED YASIN	yasin@gmail.com	software developer	70000

[Back to Home](#)

5. SEARCH EMPLOYEE BY ID:

Search Employee by ID

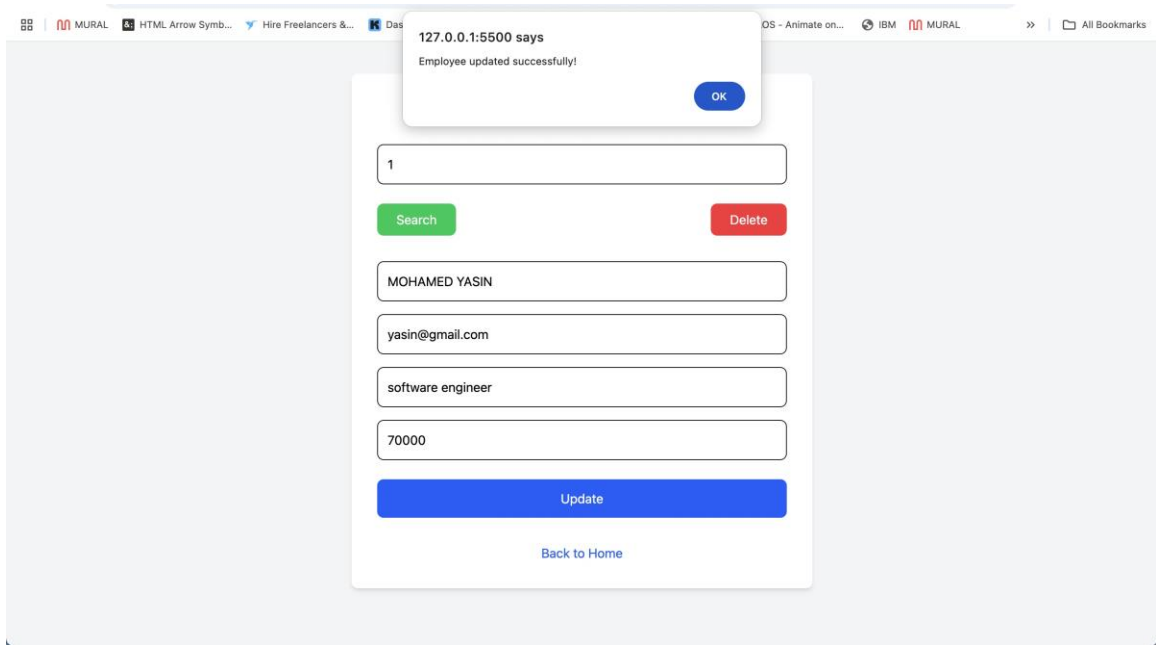
Search

Delete

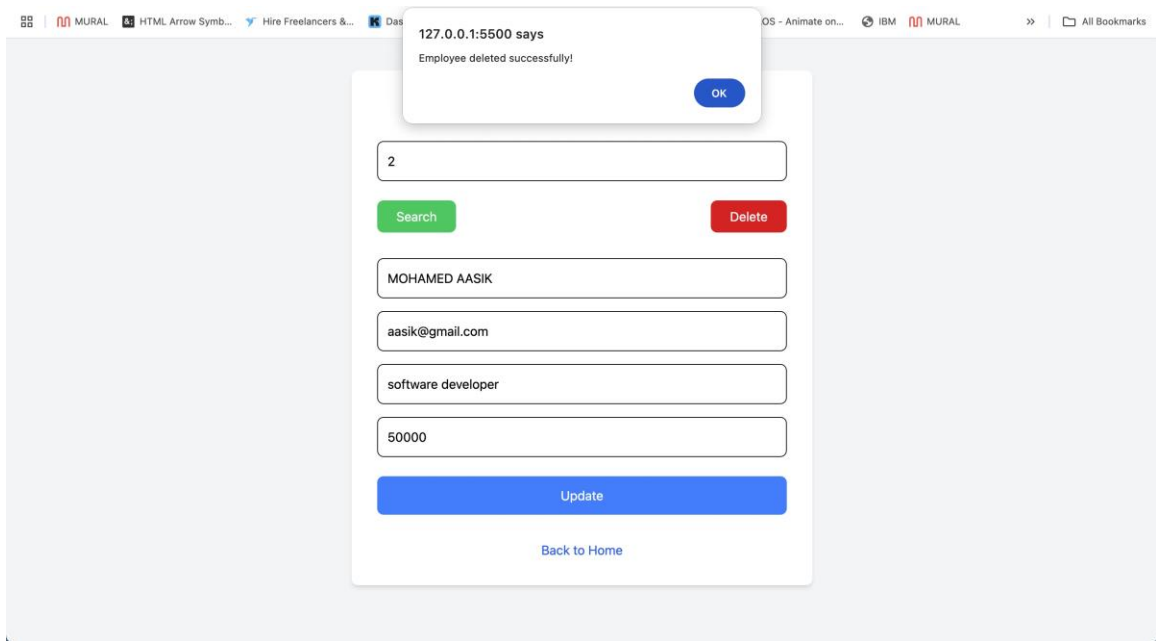
Update

[Back to Home](#)

6. UPDATE EMPLOYEE BY ID:



7. DELETE EMPLOYEE BY ID:



Postman API Screenshots:

1. ADD EMPLOYEE:

The screenshot shows the Postman interface with a workspace named "My Workspace". On the left, there's a sidebar with "Collections", "Environments", "Flows", and "History". The "Collections" section is active, showing a collection named "My first collection" with two folders: "First folder inside collection" and "Second folder inside collection". A "Create a collection for your requests" dialog is open, explaining that a collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it. A "Create Collection" button is visible.

The main area shows a request to `http://localhost:8080/api/employee/add` with a `POST` method. The request body is a JSON object:

```
1 {
2   "name": "mohamed haris",
3   "email": "haris@gmail.com",
4   "role": "frontend developer",
5   "salary": 50000
6 }
```

The response is a `200 OK` status with a response time of 244 ms and a body size of 355 B. The response body is a JSON object:

```
1 {
2   "name": "mohamed haris",
3   "email": "haris@gmail.com",
4   "role": "frontend developer",
5   "salary": 50000.0,
6   "id": 3
7 }
```

The bottom status bar shows "Online", "Find and replace", "Console", "Postbot", "Runner", "Start Proxy", "Cookies", "Vault", "Trash", and a help icon.

2. DISPLAY ALL EMPLOYEE

The screenshot shows the Postman interface with a workspace named "My Workspace". On the left, there's a sidebar with "Collections", "Environments", "Flows", and "History". The "Collections" section is active, showing a collection named "My first collection" with two folders: "First folder inside collection" and "Second folder inside collection". A "Create a collection for your requests" dialog is open, explaining that a collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it. A "Create Collection" button is visible.

The main area shows a request to `http://localhost:8080/api/employee/displayAll` with a `GET` method. The request body is empty. The response is a `200 OK` status with a response time of 179 ms and a body size of 459 B. The response body is a JSON array of two employee objects:

```
1 [
2   {
3     "name": "MOHAMED YASIN",
4     "email": "yasin@gmail.com",
5     "role": "software engineer",
6     "salary": 70000.0,
7     "id": 1
8   },
9   {
10    "name": "mohamed haris",
11    "email": "haris@gmail.com",
12    "role": "frontend developer",
13    "salary": 50000.0,
14    "id": 3
15  }
16 ]
```

The bottom status bar shows "Online", "Find and replace", "Console", "Postbot", "Runner", "Start Proxy", "Cookies", "Vault", "Trash", and a help icon.

3. DISPLAY EMPLOYEE BY ID:

My Workspace | New | Import

Overview | GET http://localhost:8080/api/employee/display/1

Save

GET | http://localhost:8080/api/employee/display/1 | Send

Params | Authorization | Headers (8) | Body | Scripts | Settings

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body | Cookies | Headers (8) | Test Results | 200 OK · 80 ms · 354 B

JSON | Preview | Visualize

```
1 {
2   "name": "MOHAMED YASIN",
3   "email": "yasin@gmail.com",
4   "role": "software engineer",
5   "salary": 70000.0,
6   "id": 1
7 }
```

Online | Find and replace | Console | Postbot | Runner | Start Proxy | Cookies | Vault | Trash

4. UPDATE EMPLOYEE BY ID:

My Workspace | New | Import

Overview | PUT http://localhost:8080/api/employee/update/3

Save

PUT | http://localhost:8080/api/employee/update/3 | Send

Params | Authorization | Headers (8) | Body | Scripts | Settings

none | form-data | x-www-form-urlencoded | **raw** | binary | GraphQL | JSON

Beautify

```
1 {
2   "name": "mohamed haris",
3   "email": "haris@gmail.com",
4   "role": "backend developer",
5   "salary": 50000
6 }
```

Body | Cookies | Headers (8) | Test Results | 200 OK · 81 ms · 354 B

JSON | Preview | Visualize

```
1 {
2   "name": "mohamed haris",
3   "email": "haris@gmail.com",
4   "role": "backend developer",
5   "salary": 50000.0,
6   "id": 3
7 }
```

Online | Find and replace | Console | Postbot | Runner | Start Proxy | Cookies | Vault | Trash

5. DELETE EMPLOYEE BY ID:

My Workspace

NewImport

Overview

DEL http://localhost:8080/api

+

No environment

Collections

Environments

Flows

History

My first collection

First folder inside collection

Second folder inside collection

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

http://localhost:8080/api/employee/delete/3

DELETE

http://localhost:8080/api/employee/delete/3

Send

Save

Params

Authorization

Headers (8)

Body

Scripts

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

{

"name": "mohamed haris",

"email": "harris@gmail.com",

"role": "backend developer",

"salary": 50000

}

Body

Cookies

Headers (8)

Test Results

200 OK

67 ms

282 B

Raw

Preview

Visualize

1 Employee deleted successfully

Online

Find and replace

Console

Postbot

Runner

Start Proxy

Cookies

Vault

Trash