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
}
```

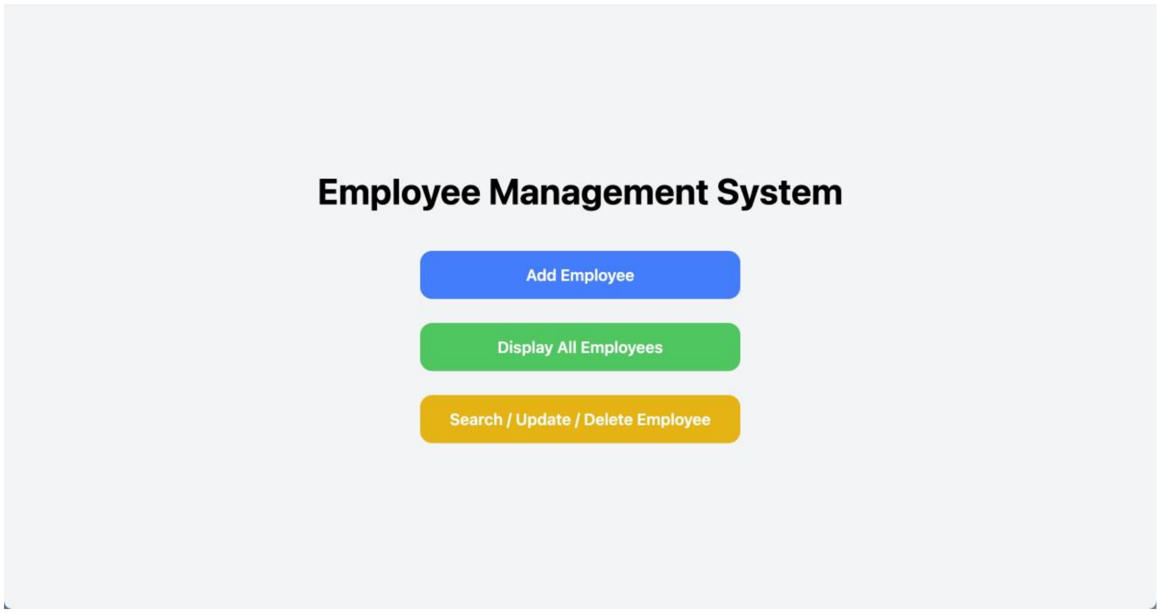
4. Data Validation Rules

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

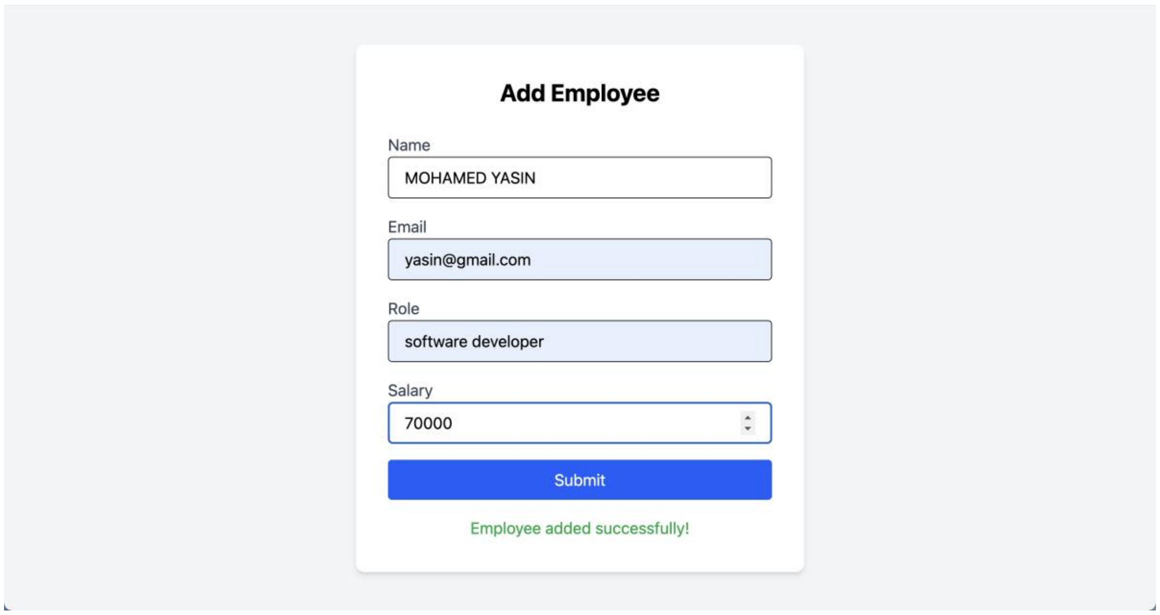
5. Screenshots

Frontend Screenshots:

1.HOME PAGE:



2. ADD EMPLOYEE:


Add Employee

Name
MOHAMED YASIN

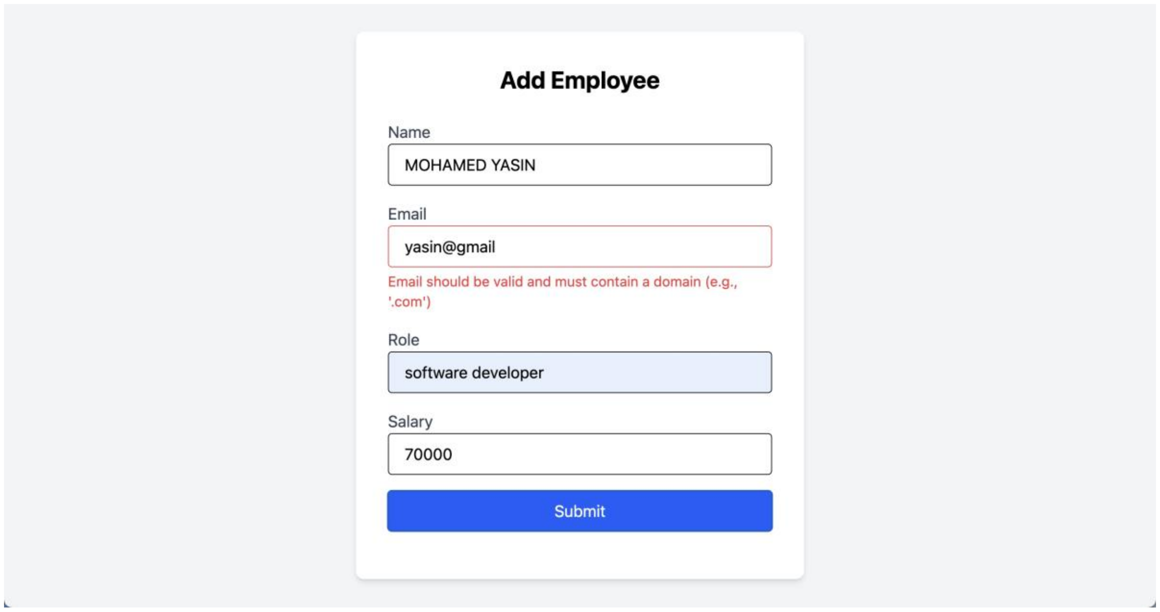
Email
yasin@gmail.com

Role
software developer

Salary
70000

Employee added successfully!

3. ADD EMPLOYEE VALIDATION:


Add Employee

Name
MOHAMED YASIN

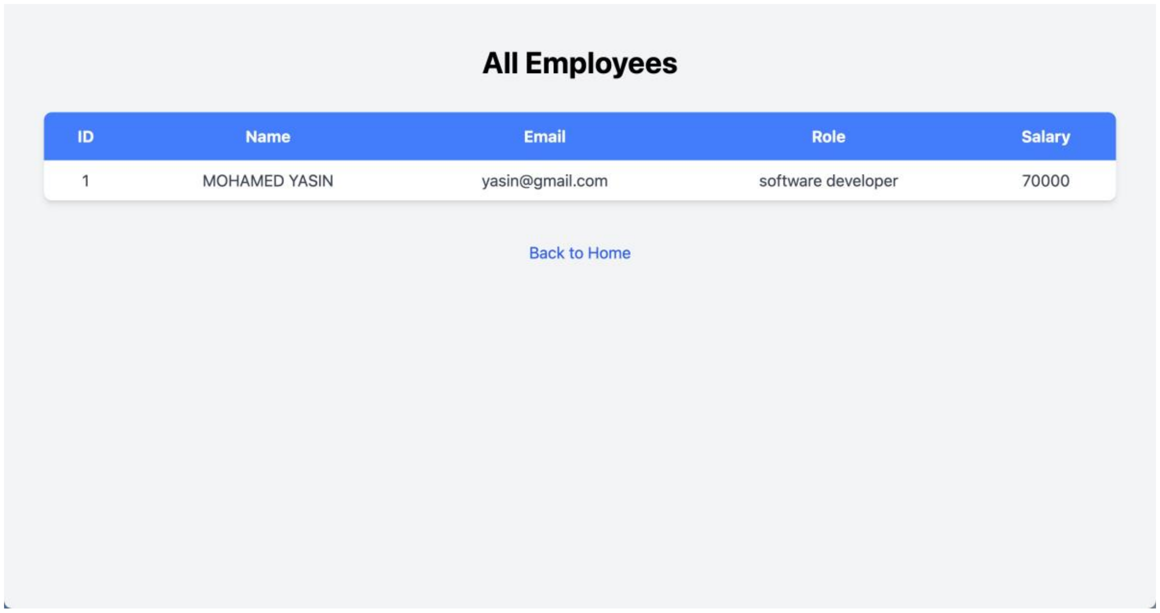
Email
yasin@gmail

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

Role
software developer

Salary
70000

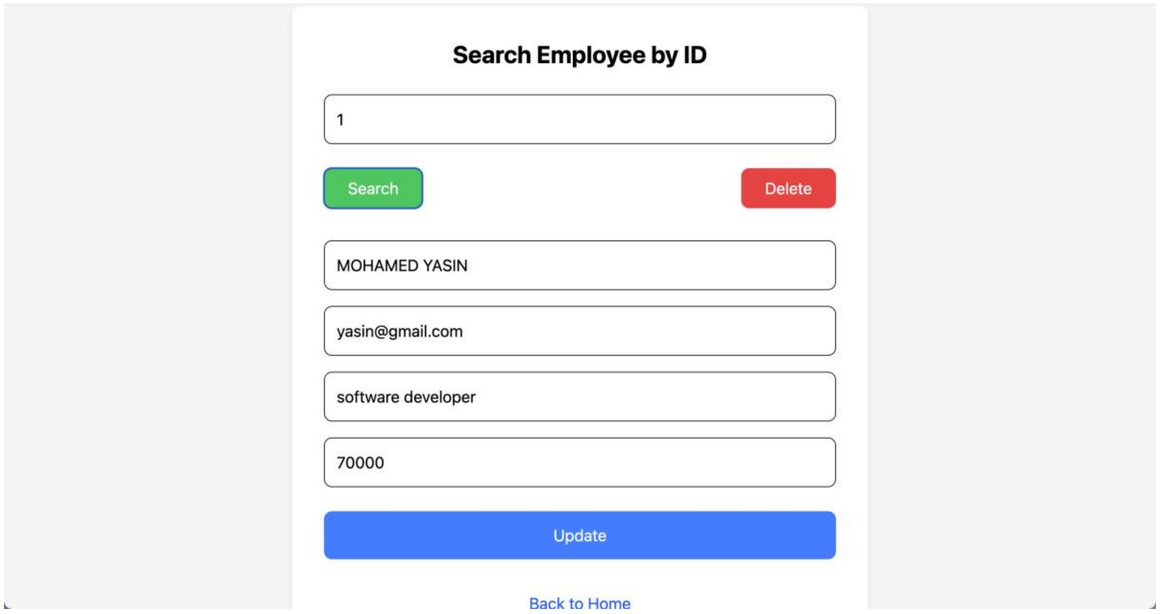
4. STORED IN A DATABASE:



All Employees

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

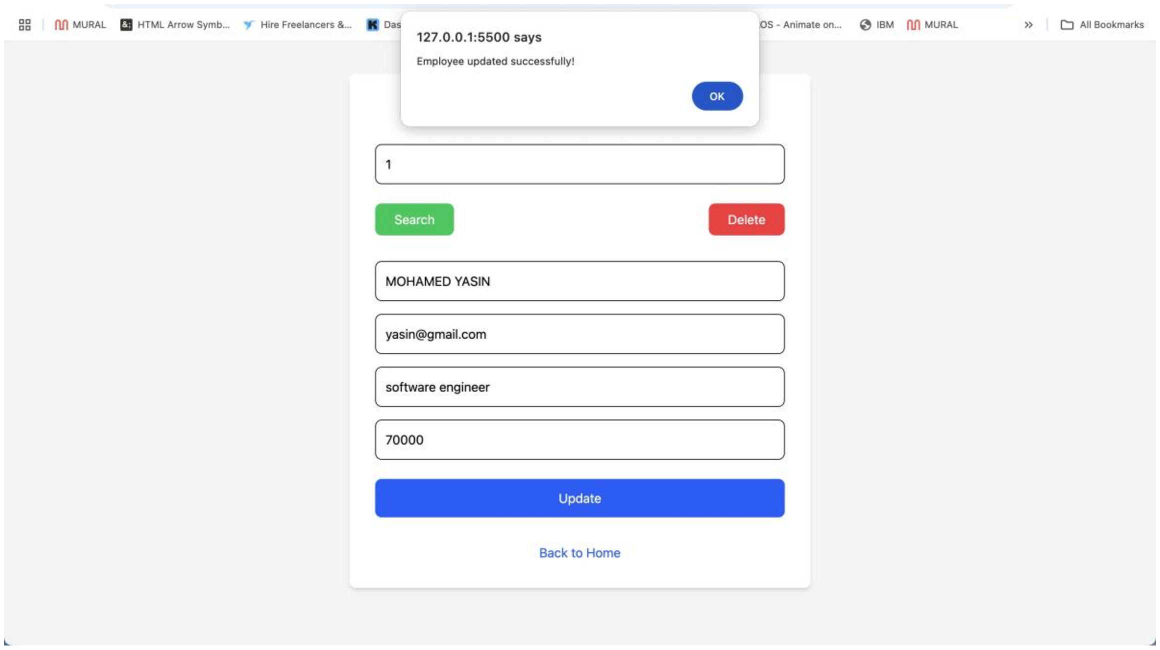
5. SEARCH EMPLOYEE BY ID:



Search Employee by ID

1
MOHAMED YASIN
yasin@gmail.com
software developer
70000

6. UPDATE EMPLOYEE BY ID:



127.0.0.1:5500 says

Employee updated successfully!

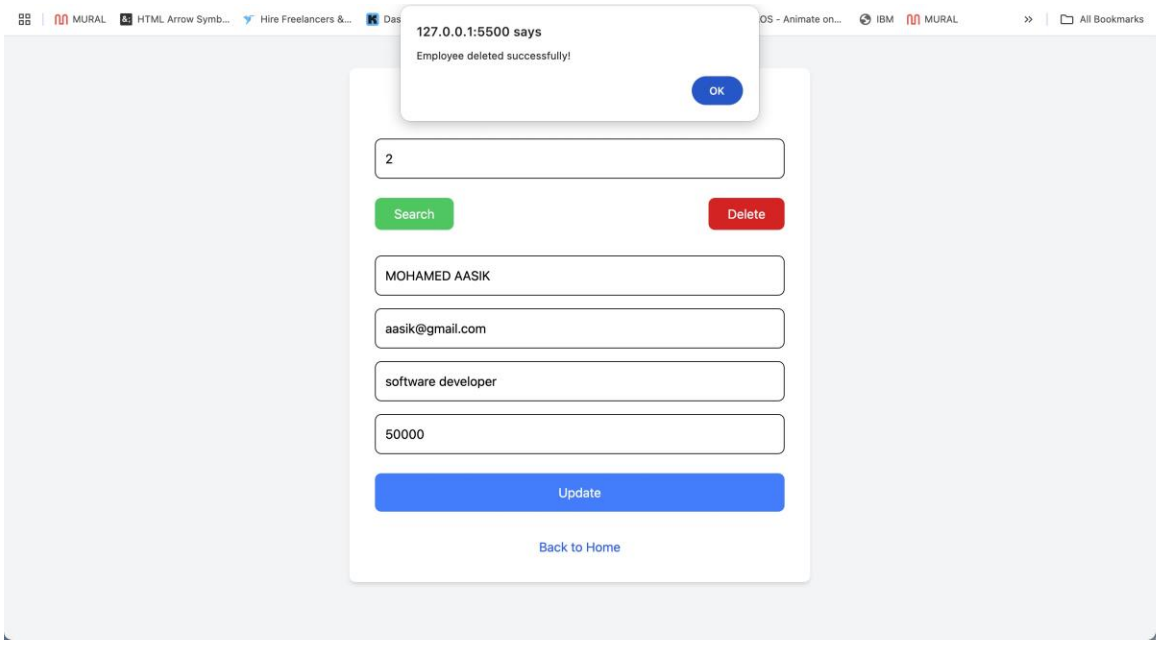
MOHAMED YASIN

yasin@gmail.com

software engineer

70000

7. DELETE EMPLOYEE BY ID:



127.0.0.1:5500 says

Employee deleted successfully!

MOHAMED AASIK

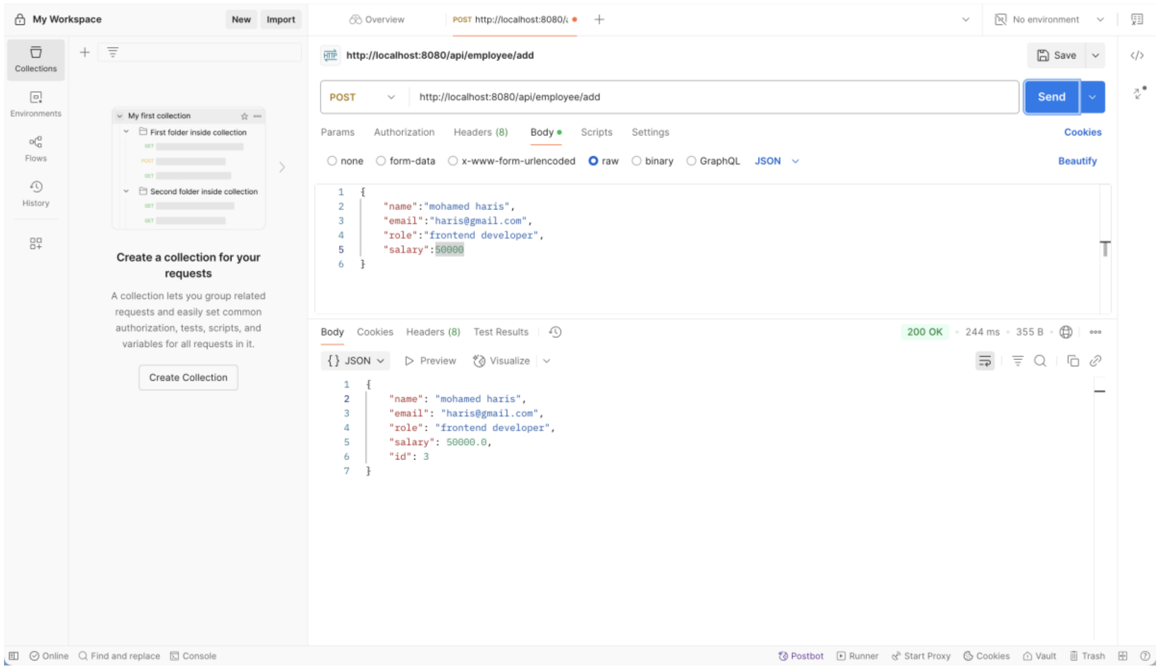
aasik@gmail.com

software developer

50000

Postman API Screenshots:

1. ADD EMPLOYEE:



http://localhost:8080/api/employee/add

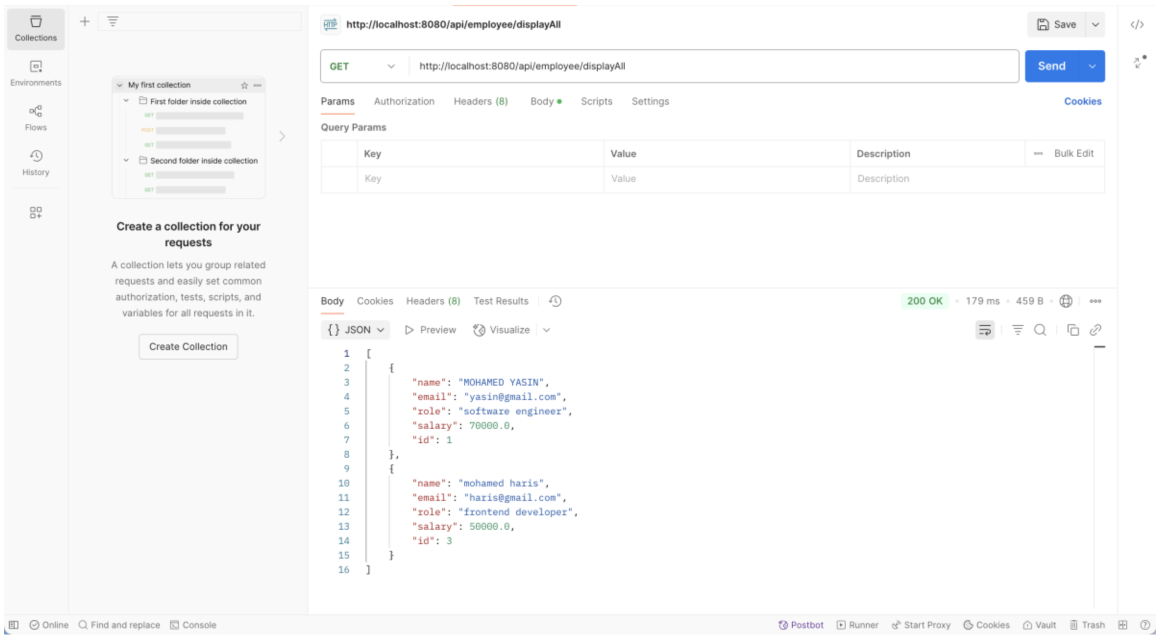
POST http://localhost:8080/api/employee/add

```
{
    "name":"mohamed haris",
    "email":"haris@gmail.com",
    "role":"frontend developer",
    "salary":50000
}
```

200 OK · 244 ms · 355 B

```
{
    "name": "mohamed haris",
    "email": "haris@gmail.com",
    "role": "frontend developer",
    "salary": 50000.0,
    "id": 3
}
```

2. DISPLAY ALL EMPLOYEE



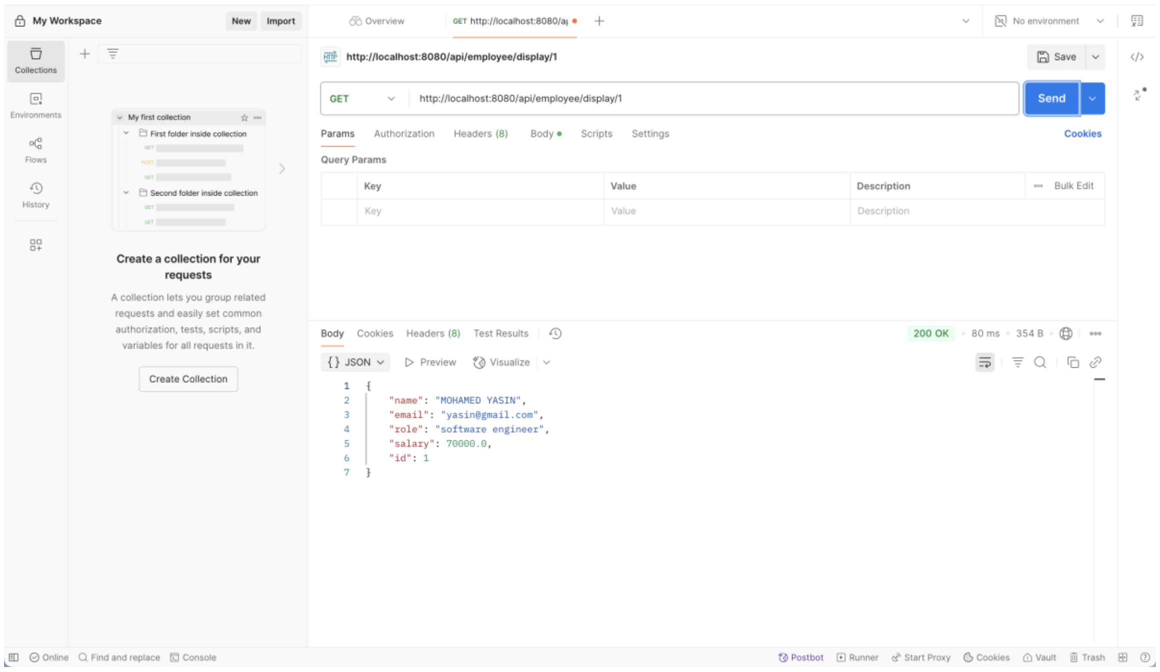
http://localhost:8080/api/employee/displayAll

GET http://localhost:8080/api/employee/displayAll

200 OK · 179 ms · 459 B

```
[
    {
        "name": "MOHAMED YASIN",
        "email": "yasin@gmail.com",
        "role": "software engineer",
        "salary": 70000.0,
        "id": 1
    },
    {
        "name": "mohamed haris",
        "email": "haris@gmail.com",
        "role": "frontend developer",
        "salary": 50000.0,
        "id": 3
    }
]
```

3. DISPLAY EMPLOYEE BY ID:



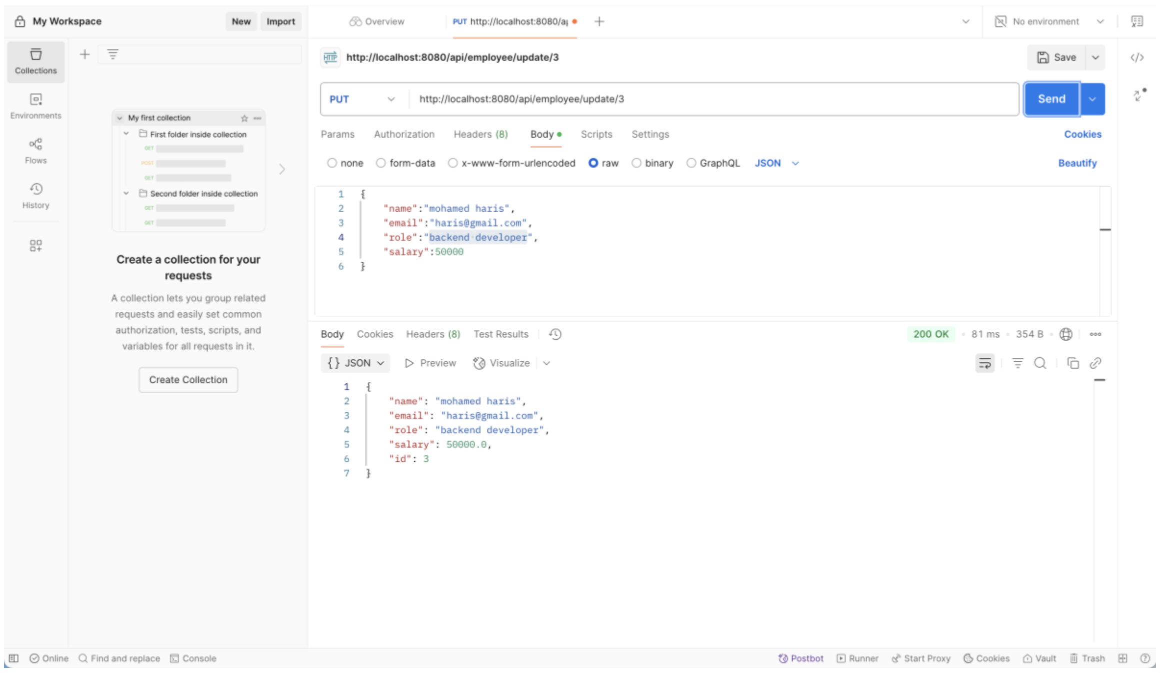
http://localhost:8080/api/employee/display/1

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

200 OK 80 ms 354 B

```
{
    "name": "MOHAMED YASIN",
    "email": "yasin@gmail.com",
    "role": "software engineer",
    "salary": 70000.0,
    "id": 1
}
```

4. UPDATE EMPLOYEE BY ID:



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

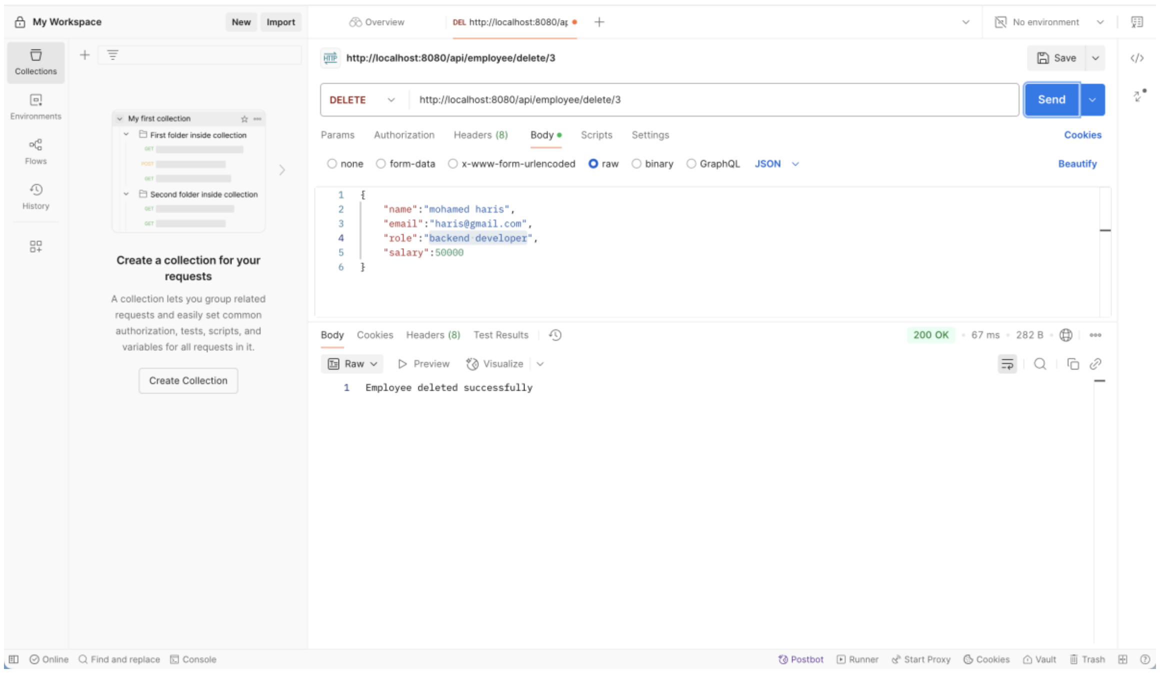
PUT http://localhost:8080/api/employee/update/3

```
{
    "name":"mohamed haris",
    "email":"haris@gmail.com",
    "role":"backend developer",
    "salary":50000
}
```

200 OK 81 ms 354 B

```
{
    "name": "mohamed haris",
    "email": "haris@gmail.com",
    "role": "backend developer",
    "salary": 50000.0,
    "id": 3
}
```

5. DELETE EMPLOYEE BY ID:



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

```
{
    "name":"mohamed haris",
    "email":"haris@gmail.com",
    "role":"backend developer",
    "salary":50000
}
```

200 OK · 67 ms · 282 B

```
Employee deleted successfully
```