

Employee Management System

Introduction

The Employee Management System (EMS) is a software application designed to streamline the process of managing employee data within an organization. It provides functionalities for creating, reading, updating, and deleting employee records efficiently.

Technologies Used

Frontend: HTML, CSS, JavaScript

Backend: Spring Boot3

Database: MySQL

API: Axios

CRUD Operations

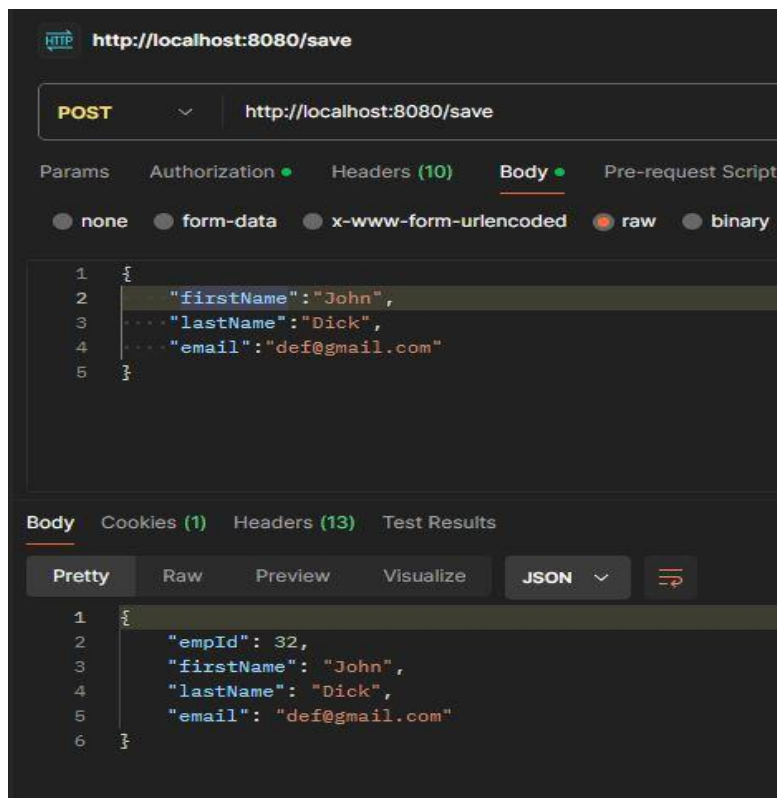
1. Create

Endpoint: `http://localhost:8080/save`

Method: POST

Request Payload: Employee details (firstName, lastName, email)

Response: empId, firstName, lastName, email.



2. Read

Endpoint: `http://localhost:8080/employees`

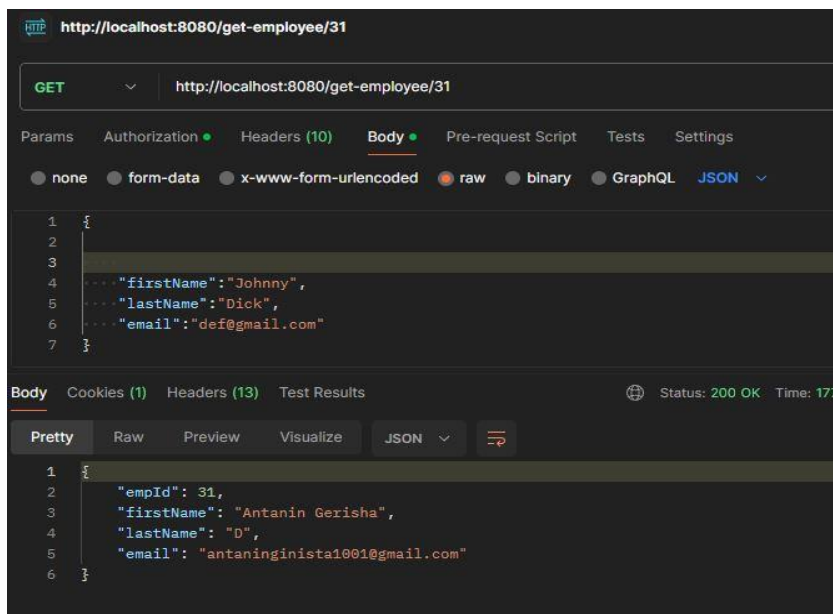
Method: GET

Response: List of employee records.

Endpoint: `http://localhost:8080/get-employee/{id}`

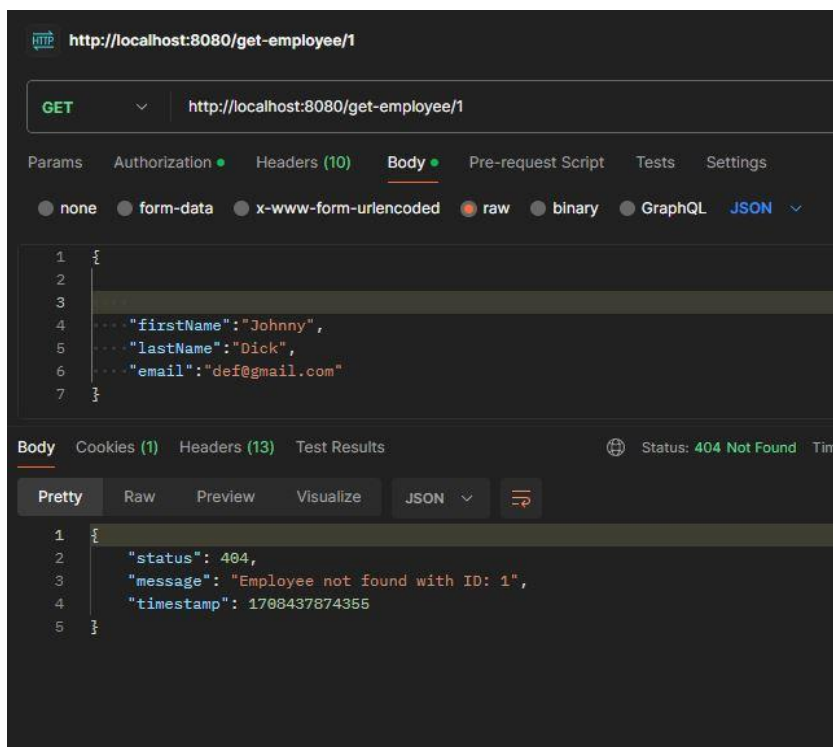
Response: Employee details corresponding to the provided ID (if employee is present)

Status, message, timestamp (if employee is not present)



```
HTTP http://localhost:8080/get-employee/31
GET http://localhost:8080/get-employee/31
Body
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1 {
2   ...
3   ...
4   ... "firstName": "Johnny",
5   ... "lastName": "Dick",
6   ... "email": "def@gmail.com"
7 }
```

```
Body Cookies (1) Headers (13) Test Results Status: 200 OK Time: 177
Pretty Raw Preview Visualize JSON
1 {
2   "empId": 31,
3   "firstName": "Antanin Gerisha",
4   "lastName": "D",
5   "email": "antaninista1001@gmail.com"
6 }
```



```
HTTP http://localhost:8080/get-employee/1
GET http://localhost:8080/get-employee/1
Body
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1 {
2   ...
3   ...
4   ... "firstName": "Johnny",
5   ... "lastName": "Dick",
6   ... "email": "def@gmail.com"
7 }
```

```
Body Cookies (1) Headers (13) Test Results Status: 404 Not Found Time: 177
Pretty Raw Preview Visualize JSON
1 {
2   "status": 404,
3   "message": "Employee not found with ID: 1",
4   "timestamp": 1708437874355
5 }
```

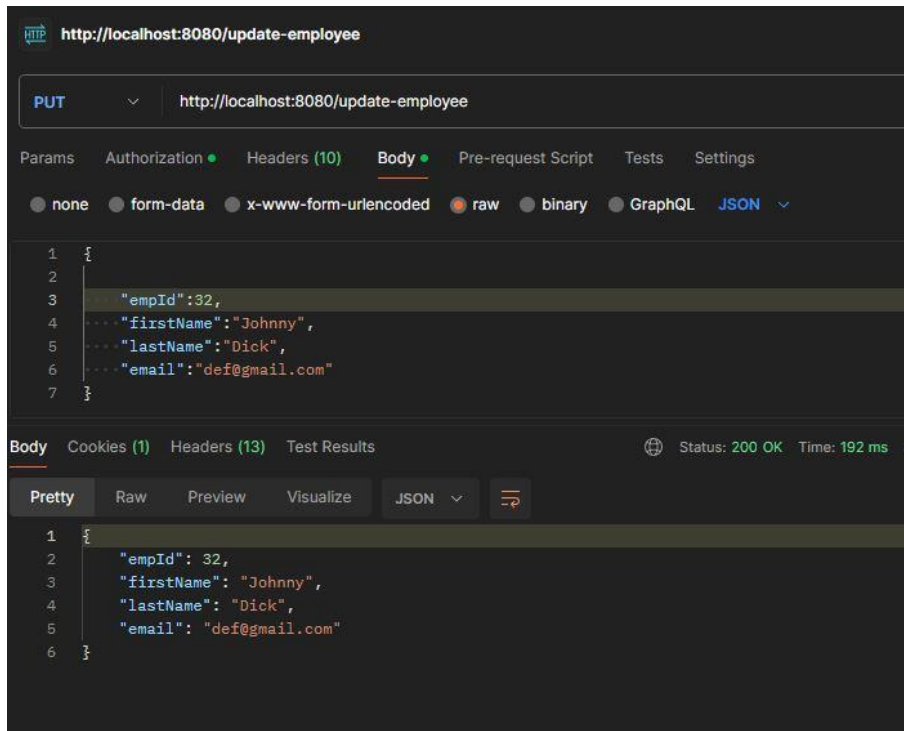
3. Update

Endpoint: `http://localhost:8080/update-employee`

Method: PUT

Request Payload: Updated employee details.

Response: Updated employee details.

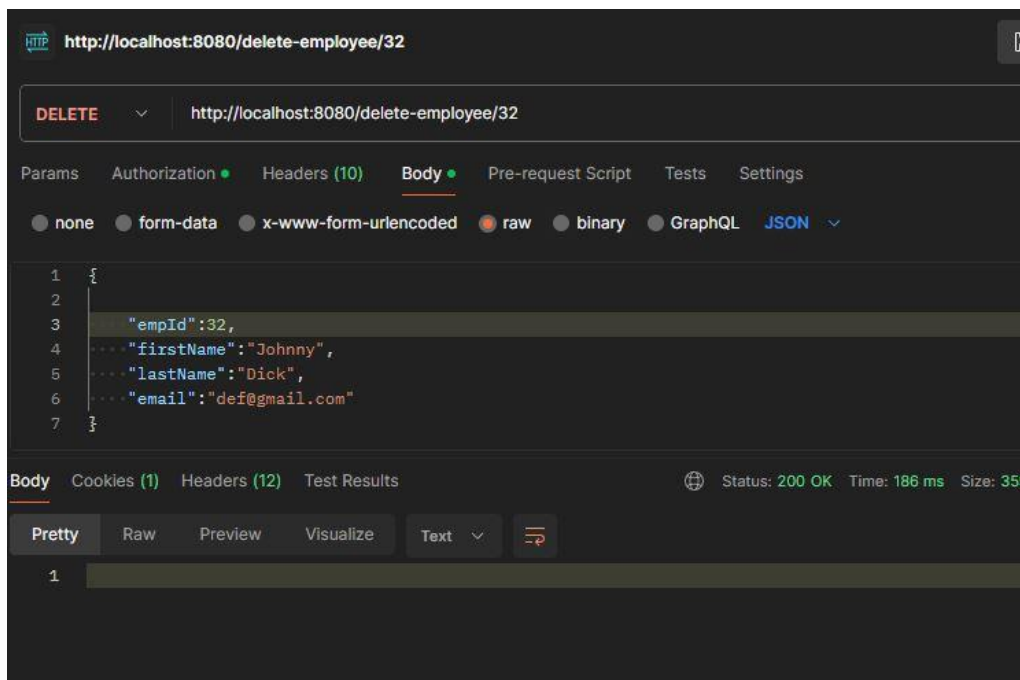


4. Delete

Endpoint: `http://localhost:8080/employees /delete-employee/ { id }`

Method: DELETE

Response: Success message.



Authentication Details

User Details:

Username: user

Password: password

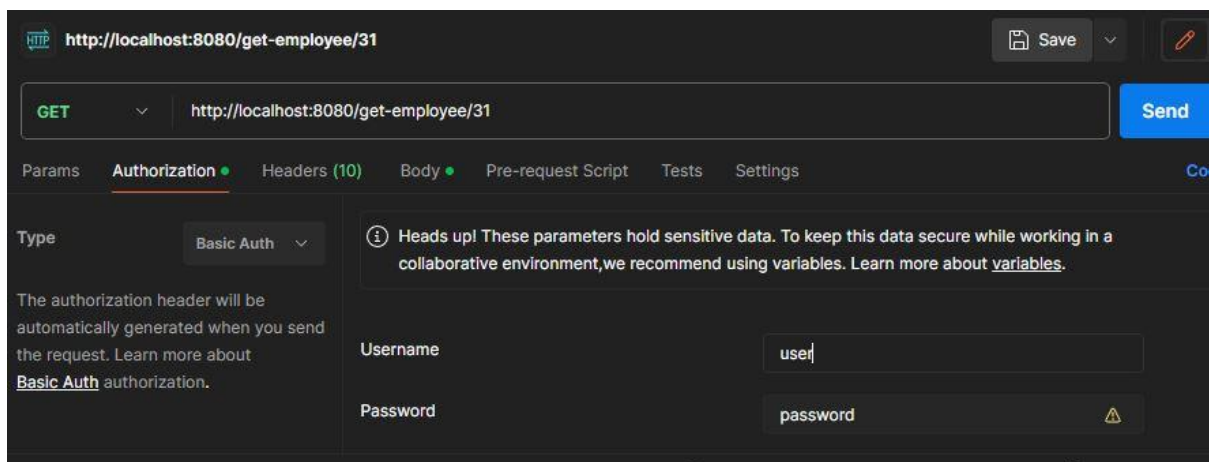
Roles: group1

Admin Details:

Username: admin

Password: password

Roles: group1



```

package com.guvi.employeeManagement.config;

import org.springframework.context.annotation.Bean;

@Configuration
@EnableWebSecurity
public class ApplicationSecurityConfig {

    @Bean
    public UserDetailsService users() {
        // The builder will ensure the passwords are encoded before saving in memory
        UserBuilder users = User.withDefaultPasswordEncoder();
        UserDetails user = users.username("user").password("password").roles("group1").build();
        UserDetails admin = users.username("admin").password("password").roles("group1").build();
        return new InMemoryUserDetailsManager(user, admin);
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.authorizeHttpRequests((authz) -> authz.anyRequest().authenticated())
            .httpBasic()
            .and()
            .csrf()
            .disable()
            .headers().frameOptions().disable();

        return http.build();
    }
}

```

ORM Mapping

```

package com.guvi.employeeManagement.entity;

import jakarta.persistence.Column;

@Entity
@Table(name = "employees")
@Getter
@Setter
@NoArgsConstructor
public class Employee {

    @Id
    @Column(name = "empId")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int empId;

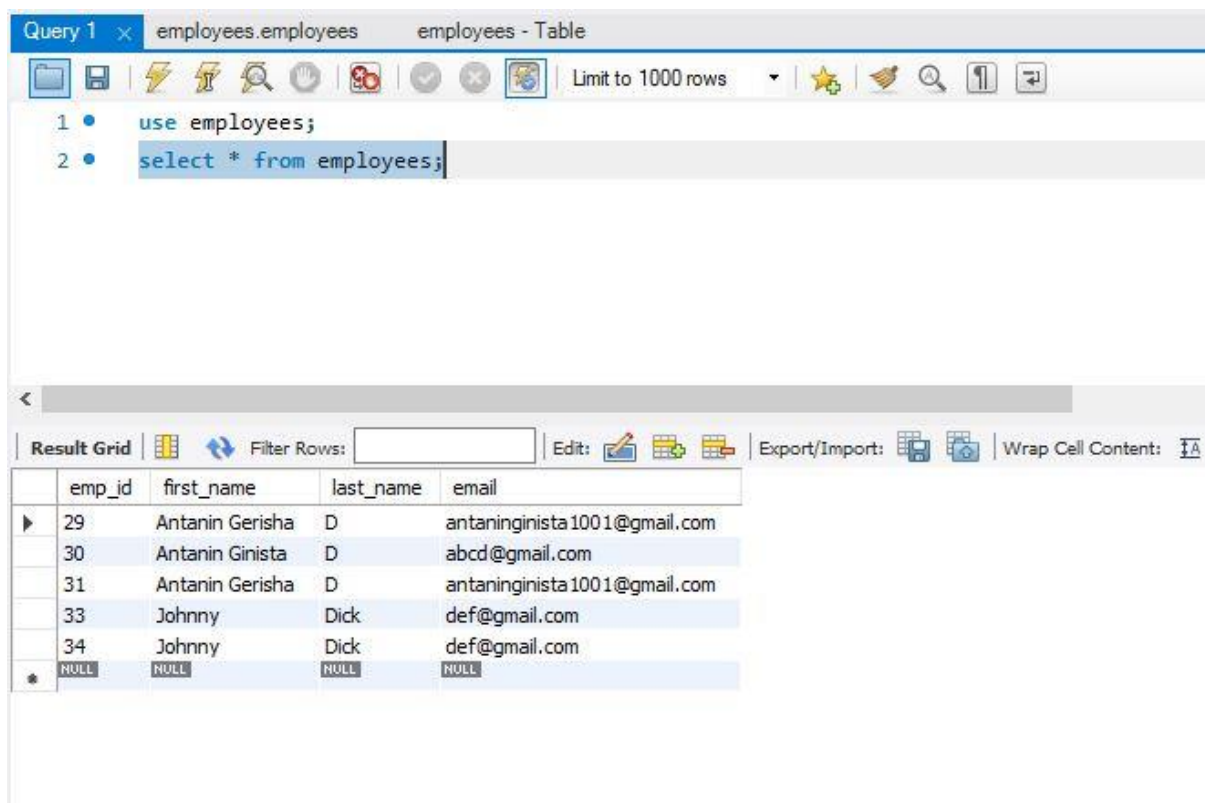
    @NotNull(message = "First Name is required")
    @Column(name = "firstName")
    private String firstName;

    @NotNull(message = "Last Name is required")
    @Column(name = "lastName")
    private String lastName;

    @Email(message = "Invalid Email Address")
    @Column(name = "email")
    private String email;
}

```

Database



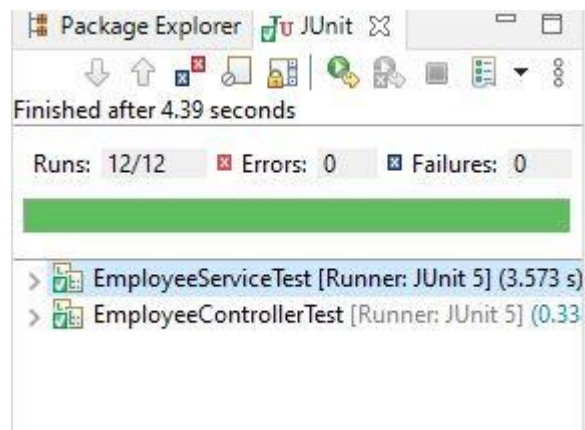
The screenshot shows a database query tool interface. At the top, there's a tab labeled "Query 1" and another labeled "employees - Table". Below the tabs is a toolbar with various icons. The main area contains a SQL query:

```
1 • use employees;  
2 • select * from employees;
```

Below the query, there's a "Result Grid" section. It includes a "Filter Rows:" input field, an "Edit:" button, and an "Export/Import:" button. The "Wrap Cell Content:" checkbox is also visible. The result grid displays the following data:

emp_id	first_name	last_name	email
29	Antanin Gerisha	D	antaninista1001@gmail.com
30	Antanin Ginista	D	abcd@gmail.com
31	Antanin Gerisha	D	antaninista1001@gmail.com
33	Johnny	Dick	def@gmail.com
34	Johnny	Dick	def@gmail.com
NULL	NULL	NULL	NULL

Testing : Employee controller and Employee service



The screenshot shows a JUnit test runner interface. At the top, there's a "Package Explorer" and a "JUnit" icon. Below them is a toolbar with various icons. The main area displays the test results:

Finished after 4.39 seconds

Runs: 12/12 Errors: 0 Failures: 0

The test results are listed below:

- > EmployeeServiceTest [Runner: JUnit 5] (3.573 s)
- > EmployeeControllerTest [Runner: JUnit 5] (0.33 s)

Conclusion

The Employee Management System provides a comprehensive solution for managing employee data efficiently

Demo Url: https://drive.google.com/file/d/1BaU9X-xoLg2pPWHEPbirt4vzZjPzLxmY_/view

