Employee Management System

Problem Statement

Hilton Hotel administrators want to automate their employee data by having an employee management system. As an employee(admin/user) he should be able to access the employee data to perform various functions.

In this scenario, develop a REST API for managing employee data in an Employee Management System. The API will provide CRUD operations (Create, Read, Update, Delete) for the "Employee" entity, along with pagination and sorting capabilities.

Objectives of the application

- Design and implement a Spring MicroService Application.
- Enable CRUD operations for the "Employee" entity.
- Implement pagination to handle large datasets.
- Enable sorting functionality for retrieving employee data.
- Validate input data to ensure data integrity.
- Implement error(Exception) handling mechanisms for informative error responses. ● Add Leave entity.
- Create Leave and Employee separate microservice applications.

Time required: 60 mins

Project Structure

Gitlab link:

https://gitlab-in.globallogic.com/m.m/employeemanagementsystemmicroserviceapp

Software Requirements

1. OS (Windows/Linux)
2. IDE of your choice (STS/Intellij IDEA)
3. JDK 17
4. MySql workbench
5. PostMan

Instruction to be followed

1. Connect with VPN first to execute second statement
2. Clone the project structure from the gitlab link given above
3. Install and start the MySql workbench database
4. Create the database schema with the name emsdb

5. Fields of the entity class:

Table Structure

1) Employee

Column Datatype Description
empId bigint(PK) Stores employee Id
first_name varchar(255) Stores firstName
last_name varchar(255) Stores lastName
email varchar(255) Stores email
department varchar(255) Stores department details

position varchar(255) Stores the designation salary bigint Stores

2) Leave

Column Datatype Description
Leave_id bigint(PK) Stores leave id
empId bigint(FK) Stores empId
applicationDate date Stores application date
leaveStartDate date Stores start date
leaveEndDate date Stores end date

leaveType varchar(255) Stores leave type

reason varchar(255) Stores the reason for taking a leave

Project Backend Implementation

i) API Table Documentation:

S No	Method	API Endpoints Description	Response Status
	Employee API		
1	POST	/api/employees — Create a new employee	201 Created
2	GET	/api/employees — Get all employee from the database	200 OK
3	GET	/api/employees/{id} — Get a specific employee by ID	200 OK
4	PUT	/api/employees/{id} — Update a specific employee by ID	200 OK
5	DELETE	/api/employees/{id} — Delete a specific employee by ID	200 OK
	Leave API		
6	POST	/api/employees/leaves — Employee can add or apply for a Leave	201 Created
7	GET	/api/employees/leaves/{empId} — Retrieve the details Of leaves applied by an employee	200 OK
8	GET	/api/employees/leaves/details/{leaveId} - Retrieve the details Of employee based on leaveId	200 OK

1. Endpoints :

a. As HR he should able to create the employee

i. Endpoint: POST/employees

ii. Description: Create a new employee

iii. Status Code will be 201 Created

iv. Request body:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "department": "Engineering",
  "position": "Software Engineer",
  "salary": 50000
}
```

v. Response body:

```
{
  "id": "1",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "department": "Engineering",
  "position": "Software Engineer",
  "salary": 50000
}
```

b. Get all employees with Pagination and Sorting

i. Endpoint: GET/employees

ii. Description: Retrieve a list of employees with pagination and sorting iii. Query Parameters:

1. pageNo: The page number to retrieve (default: 1)
2. pageSize: The number of employees per page (default: 10)
3. sortBy: The field to sort the employees by (default: EmployeeID)
4. sortDir: The sort order, either "asc" (ascending) or "desc" (descending) (default: asc).

iv. Response body:

```
{
  "page": 1,
  "pageSize": 10,
```

```
"totalPages": 3,
"totalEmployees": 27,
"employees": [
 {
  "id": "1",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "department": "Engineering",
  "position": "Software Engineer",
  "salary": 50000
 },
 // ... other employees
]
}
```


c. Get Employee by ID:

i. Endpoint: GET /employees/{id}

ii. Description: Retrieve an employee by their ID.

iii. Response body:


```
{
  "id": "1",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "department": "Engineering",
  "position": "Software Engineer",
  "salary": 50000
}
```


d. Update an Employee

i. Endpoint: PUT /employees/{id} (employee_id) ii. Description: Update an existing employee by their ID.

iii. Request body:


```
{
  "firstName": "Jane",
  "lastName": "Smith",
  "email": "jane.smith@example.com",
  "department": "Marketing",
```

"position": "Marketing Specialist",
"salary": 55000
}

iv. Response body:

{
"id": "1",
"firstName": "Jane",
"lastName": "Smith",
"email": "jane.smith@example.com",
"department": "Marketing",
"position": "Marketing Specialist",
"salary": 55000
}

e. Delete an Employee:

i. Endpoint: DELETE /employees/{id} (employee_id)

ii. Description: Delete an employee by their ID.

iii. Response Body:

{
"message": "Employee with ID 1 has been deleted successfully."
}

2. Endpoints for Leave:

i) As Employee he should able to add/apply for a leave

i. Endpoint: POST/employees/leaves

ii. Description: ADD/Apply for Leave

iii. Status Code will be 201 Created

iv. Request Body(Sample)

{
"Leave_id":103,
"applicationDate":"02/03/2023",
"startDate":"05/05/2023",
"endDate":"10/05/2023",
"leaveType":"Sick",
"reason":"Personal",
"id":1
}

v)ResponseBody(Sample)


```
{
 "leave_id": 103,
"applicationDate": "02/03/2023",
 "startDate": "05/05/2023",
"endDate": "10/05/2023",
 "leaveType": "Sick",
  "reason": "Personal",
  "id": 1
}
```


i i) As an HR/Employee he should able to get the leave details of employee based on empId

b. Endpoint: GET/employees/leave/{id} (employee_id)

c. Description: Retrieve the Leave details Of employee based on the employee id

d. Request body:NA

e. Response body:


```
{
  “Leave_id :101,
   “ApplicationDate”:”02-May-2023”,
   “LeaveStartDate”: “05-May-2023”,
    “LeaveEndDate”: “10-May-2023”,
  “employee”:
 {
 "id": "1",
 "firstName": "John",
 "lastName": "Doe",
 "email": "john.doe@example.com",
 "department": "Engineering",
 "position": "Software Engineer",
 "salary": 50000
}

}
```

iii) As an HR/Employee he should be able to get the employee details based on leaveId b. Endpoint: GET/leave/employees/{leaveId} (leave_id)

c. Description: Retrieve the employee details based on the leave id . Request body:NA

e. Response body:


```
{
  "id": "1",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "department": "Engineering",
  "position": "Software Engineer",
  "salary": 50000
}
```


ii).User Stories for creating services:

Serial No	User Stories (EMS) Acceptance Remarks/Notes

	Criteria
Theme: Employee	

US01	As HR i should be able to create the
	
	profile of the new joinee in the
	
	employee management system
		Constraints:
		
		Employee ID will
		be primary key and
		auto increment
		should be there,
		Valid Email should
		be there with 180
		character only,
		First Name, Last
		Name should have
		60 character only,
		Department,
		Position should be
		60 characters each
		
		Validation:
		
		Use Validation
		annotations to
		validate the field if
		not validated then
		throw custom
		
		Request Body(Sample):
		
		{
		"firstName": "John",
		"lastName": "Doe",
		"email":
		"john.doe@example.com",
		"department": "Engineering",
		"position": "Software
		Engineer",
		"salary": 50000
		}
		
		Response Body(Sample):
		
		{
		"id": "1",
		"firstName": "John",
		"lastName": "Doe",
		"email":
		"john.doe@example.com",
		"department": "Engineering",

		exception with proper message	"position": "Software Engineer", "salary": 50000 }

US02 **US03**	As employee i should be able to view the profiles of the existing employee in the system and should be able to get the sorting capability with pagination too. As an employee i should be able to view a specific profile of the employee it may be my profile or other people profile	Pagination and Sorting concept are must **Exception:** Handle the exception using a custom exception if invalid id is

Response Body(Sample):


```
{
  "page": 1,
"pageSize": 10,
"totalPages": 3,
"totalEmployees": 27,
"employees": [
  {
  "id": "1",
"firstName": "John",
"lastName": "Doe",
  "email":
"john.doe@example.com",
"department": "Engineering",
 "position": "Software
  Engineer",
 "salary": 50000
  },
// ... other employees
  ]

  }
```


Response Body(Sample):


```
  {
  "id": "1",
"firstName": "John",
"lastName": "Doe",
```

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

	"department": "Engineering", "position": "Software Engineer", "salary": 50000 }

US04	As a system admin i should be able to update the profile of the existing employee if they require changes	**Validation:** Use Validation annotations to validate the field if not validated then throw custom exception with proper message **Exception:** Handle the exception using a custom exception if invalid id is provided.

Request body(Sample):

```
{
"firstName": "Jane",
"lastName": "Smith",
"email":
"jane.smith@example.com",
"department": "Marketing",
"position": "Marketing
Specialist",
"salary": 55000
}
```

Response body(Sample):

```
{
"id": "1",
"firstName": "Jane",
"lastName": "Smith",
"email":
"jane.smith@example.com",
"department": "Marketing",
"position": "Marketing
Specialist",
"salary": 55000
}
```

US05	As a system admin i should be able to
	
	delete the profile of the employee
	
	when they are resigning or getting out
	
	of the system
		Exception:
		
		Handle the
		
		exception using
		
		custom exception if
		
		invalid id is
		
		provided
		
		Response Body(Sample):
		
		{
		"message": "Employee with ID
		
		1 has
		
		been deleted successfully."
		}

Theme:Leaves

US06	As an Employee i should be able to add
	
	or apply for a Leave
		Exception:
		
		Handle the exception using custom exception if invalid id is provided
		
		Request Body(Sample)
		
		{
		"leave_id":103,
		"applicationDate":"02/03/2023",
		"startDate":"05/05/2023",
		"endDate":"10/05/2023",
		"leaveType":"Sick",
		"reason":"Personal",
		"id":1
		}
		
		ResponseBody(Sample)
		
		{
		"leave_id": 103,
		"applicationDate": "02/03/2023",
		"startDate": "05/05/2023",
		"endDate": "10/05/2023",
		"leaveType": "Sick",
		"reason": "Personal",
		"id": 1
		}

US07 As an HR/Employee I can see the details of leaves based on the emp_id

Exception:

Handle the exception using a custom exception if

Response Body

Response body:

{

“Id”:”1” ///EmployeeID

	invalid id is provided.	“employee”:	{ **Constraints:** "firstName": "John",
		Leave ID will be primary key and auto increment should be there Emp_Id will be Foreign Key which refers the employee table	"lastName": "Doe", "email": "john.doe@example.com", "department": "Engineering", "position": "Software Engineer", }, [{ "leave_id": 101, "applicationDate": "02/03/2023", "startDate": "03/02/2023", "endDate": "07/02/2023", "leaveType": "Sick", "reason": "Personal", }, { "leave_id": 102, "applicationDate": "02/03/2023", "startDate": "05/05/2023", "endDate": "10/05/2023", "leaveType": "Sick",

	"reason": "Personal",

	} }] }

US08 As an HR/Employee I can see the details of Employee based on the leave_id

Exception:

Handle the exception using a custom exception if invalid id is provided.

Response Body

Response body:

{

“Id”:”1” ///leaveID

“employee”:

{

"firstName": "John",

Constraints:

		Leave ID will be primary key and auto increment should be there Emp_Id will be Foreign Key which refers the employee table	"lastName": "Doe", "email": "john.doe@example.com", "department": "Engineering", "position": "Software Engineer", } [{ "leave_id": 101, "applicationDate": "02/03/2023", "startDate": "03/02/2023", "endDate": "07/02/2023", "leaveType": "Sick", "reason": "Personal", }, { "leave_id": 102, "applicationDate": "02/03/2023", "startDate": "05/05/2023", "endDate": "10/05/2023", "leaveType": "Sick", "reason": "Personal",

	} }] }

Activity1

- ❖ Create MicroServices for the user stories given above.
  - ➢ Apply the Microservice Design principle to decide on the number of microservices
- ❖ Use Centralized Exception Handling to handle the exceptions.
- ❖ Use Interservice communication using Rest template or Feign for service interaction(like Employee and leave can be separate modules and they can communicate with each other to get the leave details based on employee_id).

Activity2

- ❖ Register the above created microservices like Employee and Leave as microservice on a discovery server(eureka server) which will track the information of all the client services and the ports.
- ❖ Route requests through API Gateway
- ❖ Use centralized configuration.
- ❖ Generate Logs using log4j (Slf4j or logback).

Push To Gitlab -Once the endpoints are created and tested successfully using PostMan push the code by creating a new branch with your name appended with your empId.