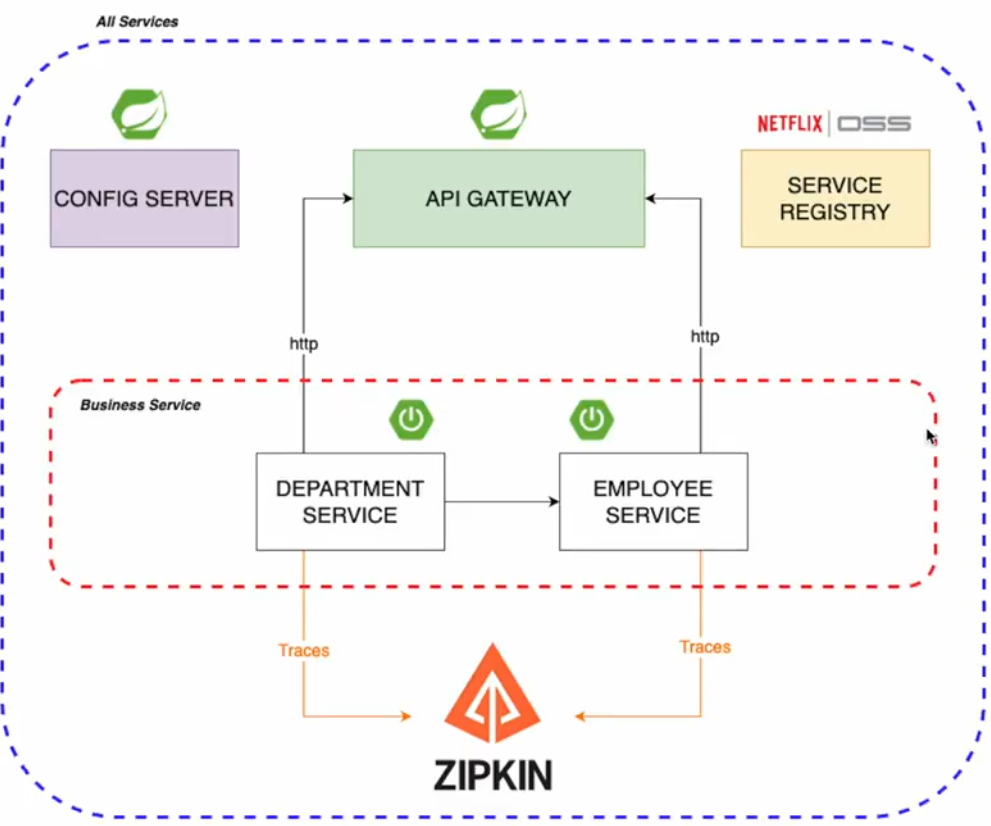
|  |
| --- |
| **COURSE- SEZG533 SERVICE ORIENTED COMPUTING**  **ASSIGNMENT- DESIGN AND IMPLEMENTATION OF AN SOA APPLICATION** |

**OBJECTIVE:**

To design and implement a service-based application for Customer Care Portal.

**PROBLEM STATEMENT:**

PART 1: DESIGN



### 1. Identify the basic capabilities of the application:

The basic capabilities of the application revolve around managing employees and departments. Key functionalities include:

- \*\*Employee Management:\*\*

- Add a new employee

- Retrieve information about an employee

- Update employee details

- Delete an employee

- \*\*Department Management:\*\*

- Add a new department

- Retrieve information about a department

- Update department details

- Delete a department

- Retrieve a list of all employees in a department

### 2. Entities and Models:

- \*\*Department:\*\*

- Properties: `id`, `name`, `employees` (this is a list of employees in the department)

|  |
| --- |
| Department.java |
|  |

- \*\*Employee:\*\*

- Properties: `id`, `name`, `age`, `position`, `departmentId`

|  |
| --- |
| Employee.java |
|  |

### 3. Resources and Operations:

#### Employee Resource:

- \*\*Resource Name:\*\* `/employees`

- \*\*Operations:\*\*

- `GET /employees`: Retrieve all employees

- `GET /employees/{employeeId}`: Retrieve a specific employee

- `POST /employees`: Create a new employee

- `PUT /employees/{employeeId}`: Update a specific employee

- `DELETE /employees/{employeeId}`: Delete a specific employee

#### Department Resource:

- \*\*Resource Name:\*\* `/departments`

- \*\*Operations:\*\*

- `GET /departments`: Retrieve all departments

- `GET /departments/{departmentId}`: Retrieve a specific department

- `POST /departments`: Create a new department

- `PUT /departments/{departmentId}`: Update a specific department

- `DELETE /departments/{departmentId}`: Delete a specific department

- `GET /departments/{departmentId}/employees`: Retrieve all employees in a specific department

### 4. URI Design, HTTP Methods, Headers:

- URI Design: Follows the RESTful conventions outlined above.

- HTTP Methods: `GET`, `POST`, `PUT`, `DELETE`.

- Headers: Common headers like `Content-Type` for specifying the format of the request or response body (e.g., JSON).

### 5. Representations:

- \*\*Accepted Representations (Request):\*\*

- For operations like `POST` and `PUT`, the client may send JSON representations of the entities in the request body.

- \*\*Served Representations (Response):\*\*

- Responses for `GET` operations may include JSON representations of the retrieved entities.

- Successful `POST`, `PUT`, and `DELETE` operations may return a simple success message or the updated/deleted entity.

### 6. Resource Categorization:

- \*\*Employee Resource Type:\*\*

- `Collection Resource`: `/employees` (for managing a collection of employees)

- `Singleton Resource`: `/employees/{employeeId}` (for managing a single employee)

- \*\*Department Resource Type:\*\*

- `Collection Resource`: `/departments` (for managing a collection of departments)

- `Singleton Resource`: `/departments/{departmentId}` (for managing a single department)

- `Subordinate Resource`: `/departments/{departmentId}/employees` (for managing employees within a specific department)

PART 2.1: IMPLEMENTATION OF SERVICE ENDPOINTS

Technology Stack used :

* Spring Boot 3
* Java 17
* Spring Framework
* Spring Cloud Netflix OSS
* API Gateway
* Spring Config Server
* Spring Eureka discovery Server

|  |
| --- |
| 5 microservices designed as 5 Spring Boot Maven Projects |
|  |

|  |
| --- |
| DepartmentController.java |
|  |

|  |
| --- |
| EmployeeController.java |
|  |

|  |
| --- |
| 7 Endpoints implemented |
|  |

**Endpoints Request & Response Screenshots –**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**PART 2.2: API GATEWAY**

As part of the implementation process for the service-oriented application, I have designed API Gateway which is running on 8060.

Department Service is running on 8081

Employee Service is running on 8082

Eureka Discovery Service is running on 8761

And API GATEWAY is running on 8060, and all routing is defined in API GATEWAY application.yaml file.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Important Links**

Github Link : <https://github.com/ShelkeAkshay-2022mt93331/SOC-Assignment-2022mt93331>

Demo Video Link (Google Drive Folder) : <https://drive.google.com/drive/folders/1qwNI56-kG9ihDYcu_jbXeMsG_hKiTbuS?usp=sharing>