

**CAPSTONE PROJECT**  
**EMPLOYEE MANAGEMENT SYSTEM**  
**BATCH-8**  
**JAVA J2EE**

**Name: SARAGADAM VARA SAI RAM**

**Email: [sairamsaragadam@gmail.com](mailto:sairamsaragadam@gmail.com)**

**Date: September 2, 2024**

**Trainer: Ramakrishna (RK)**

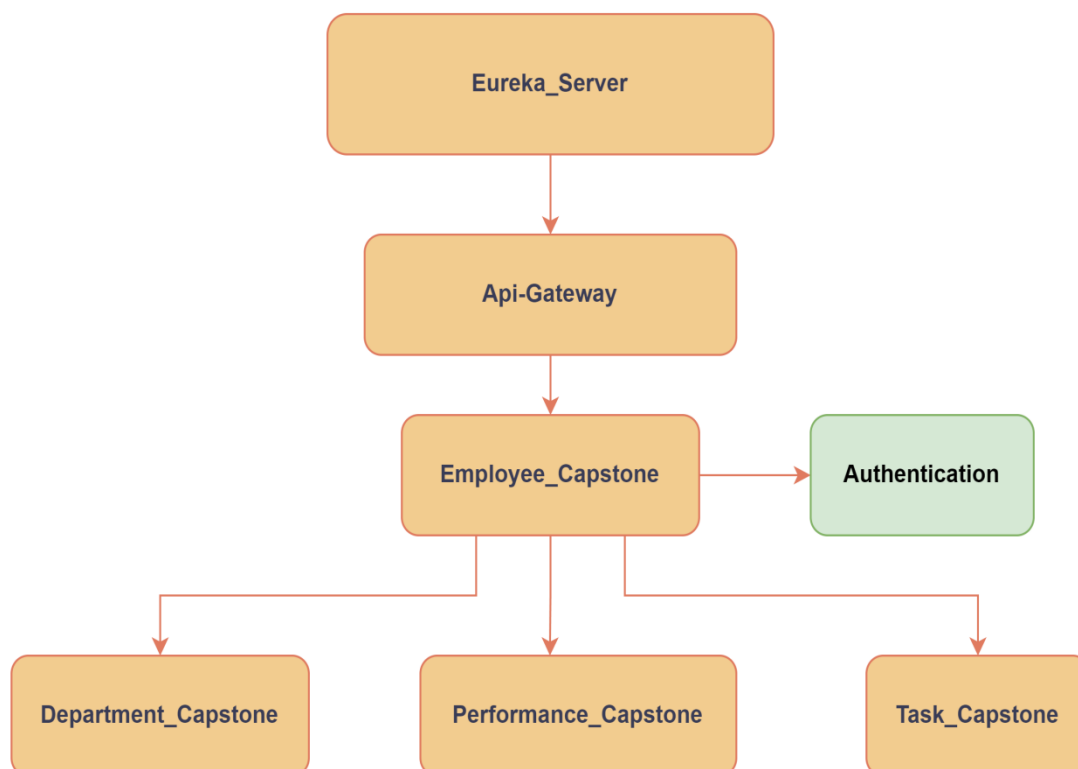
## Table Of Contents:

<b>SL No</b>	<b>Name of Content</b>	<b>Pg No</b>
1	<b>Introduction</b>	3
2	<b>Technologies Used</b>	4
3	<b>Problem Statement</b>	6
4	<b>Problem Statement</b>	6
5	<b>Project Flow Admin Module</b>	14
6	<b>Project Flow Employee Module</b>	15
7	<b>Eclipse Project Workspace-Backend:</b>	16
8	<b>Running the Application as SpringbootApp:</b>	17
9	<b>Data Base MySQL Workbench:</b>	17
10	<b>Eureka Server page:</b>	20
11	<b>Swagger UI</b>	20
12	<b>Postman API</b>	23
13	<b>Testing and Refinement</b>	24
14	<b>Conclusion</b>	27
15	<b>Future Enhancement</b>	27

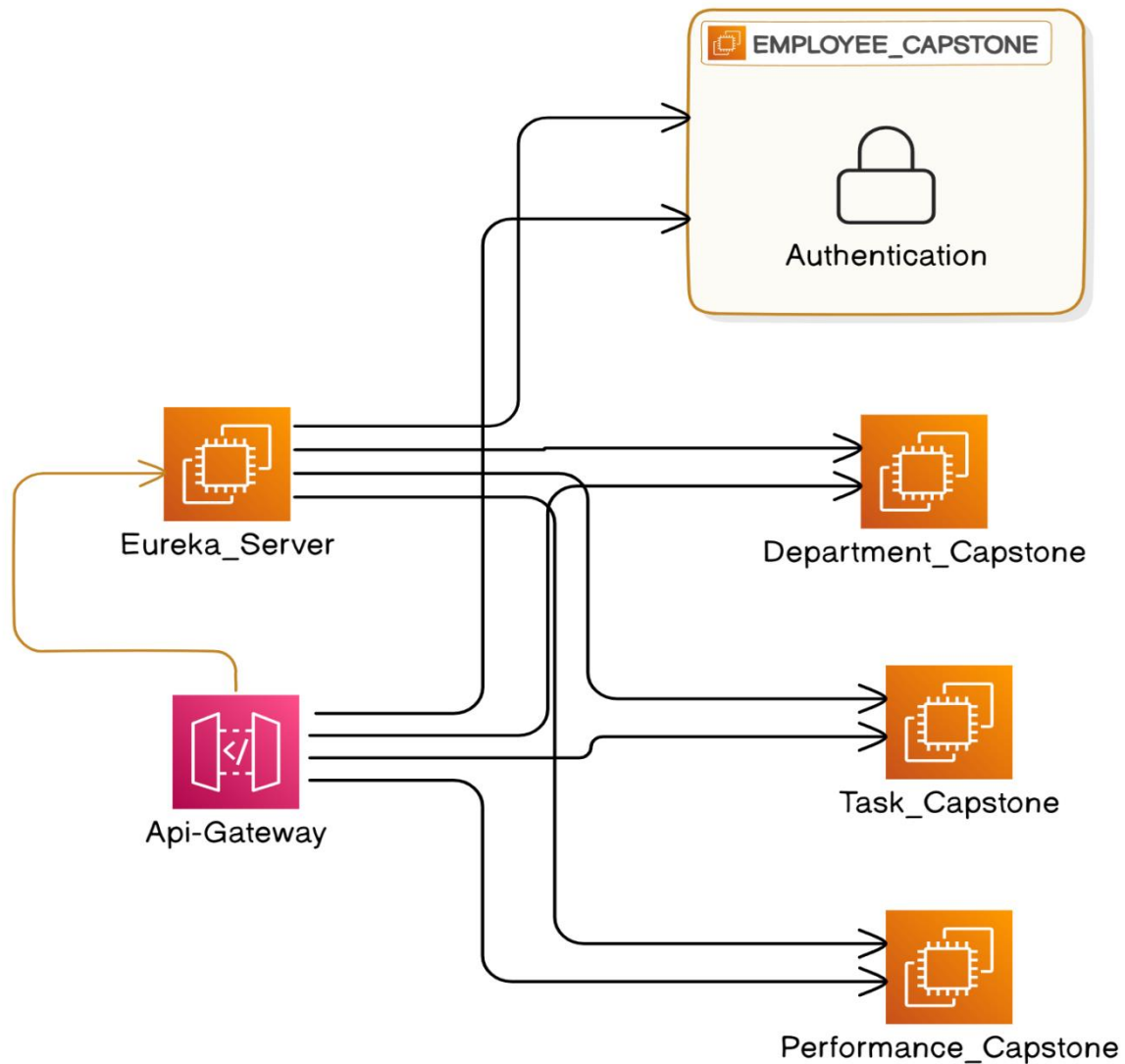
# 1. Introduction

## 1.1 Project Overview

The Employee Management System is a distributed application designed to manage employee-related information within an organization efficiently. The system allows administrators to manage employees, departments, tasks, and performance evaluations. The architecture of the system is based on microservices, which ensures scalability and flexibility, enabling the organization to adapt and grow as needed.



# Employee Management System Architecture



## 2. Technologies Used

**2.1 Java:** Core programming language used for application development.

**2.2 Spring Boot:** Framework that simplifies the creation of production-ready Spring applications, allowing for easy setup and rapid deployment.

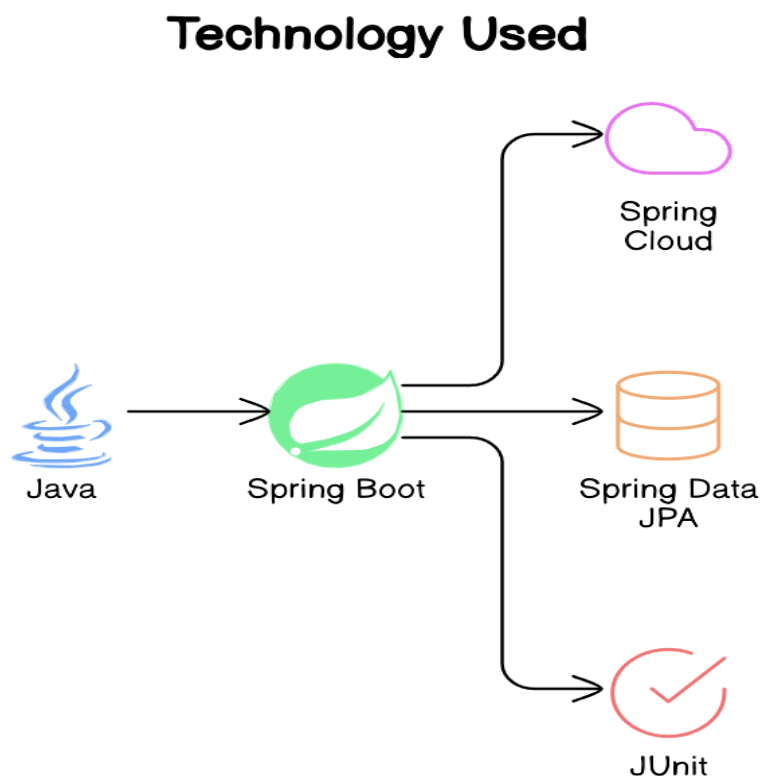
**2.3 Spring Cloud:** Handles cross-cutting concerns like configuration management, service discovery, circuit breakers, and distributed tracing, enabling microservices architecture.

**2.4 Spring Data JPA:** Simplifies data access and provides a standard API for database interactions.

**2.5 MySQL DB:** The relational database management system used for storing the application's data.

**2.6 RESTful APIs:** The architectural style used for designing networked applications and providing communication between different microservices.

**2.7 JUnit:** A testing framework used for unit testing the application's components, ensuring code quality and reliability.



### **3. Problem Statement:**

The primary goal of this project is to create a robust Employee Management System that caters to the following requirements.

#### **3.1 For Admins:**

- A centralized system to manage employees, departments, and roles.
- CRUD (Create, Read, Update, Delete) operations on employee records.
- Assignment of roles to employees and organizing them into departments.
- Management of tasks and performance evaluations.

#### **3.2 For Employees:**

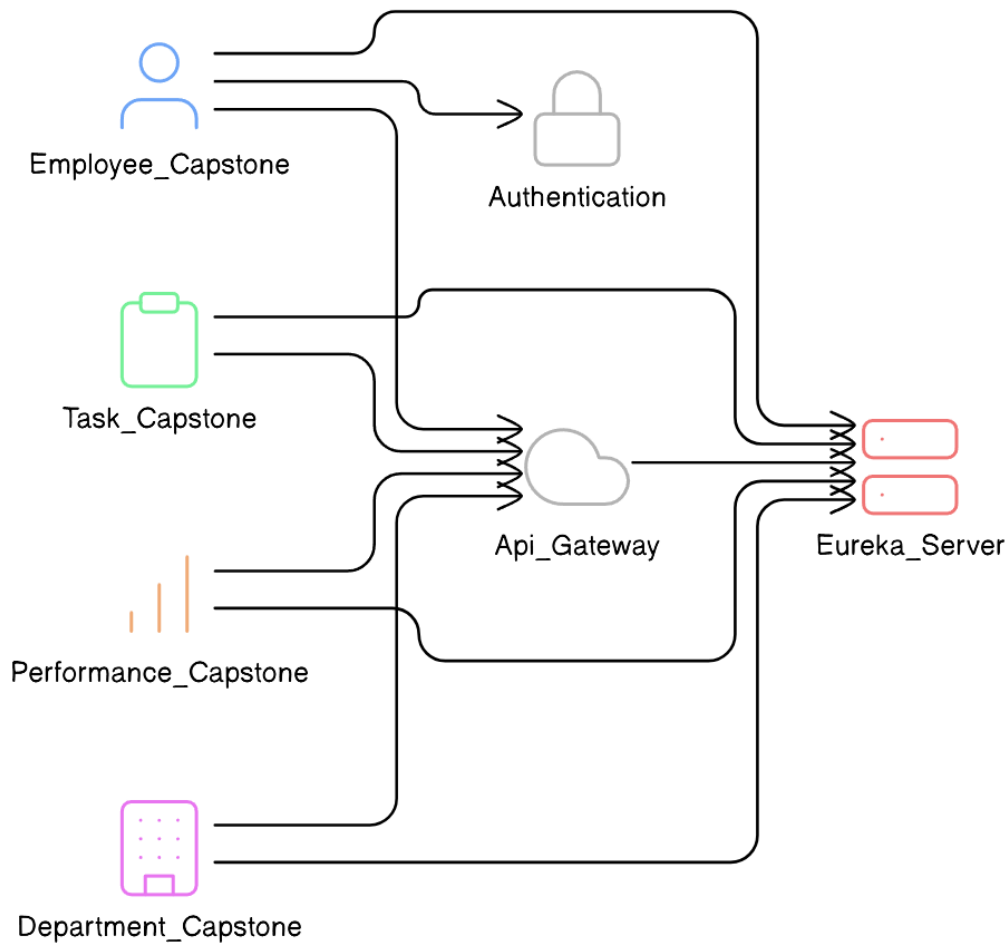
- Ability to manage personal profiles.
- View assigned tasks and update task statuses.
- Track performance and receive feedback from managers.

## **4. Microservices Architecture**

### **4.1 Overview**

Microservices architecture breaks down the application into small, independent services, each responsible for specific business functionalities. These services communicate through APIs, allowing for greater flexibility, scalability, and maintainability.

## Employee Management System Architecture



## 4.2 Service Registry & Discovery

### Eureka Server

**4.2.1 Role:** Eureka is a service registry used to keep track of all available microservices and their instances within the system. Each service registers itself with Eureka upon startup and periodically sends heartbeats to confirm its availability.

**4.2.2 Functionality:** Eureka provides a central directory where services can look up the locations (IP addresses and ports) of other services. This enables dynamic scaling and ensures that services can

discover and communicate with one another even if they are deployed on different servers or cloud instances.

**4.2.3 Failover:** Eureka can work in a high-availability mode where multiple Eureka servers are deployed, ensuring that the service registry remains available even in case of a server failure.

## 4.3 API Gateway

### Spring Cloud Gateway

**4.3.1 Role:** The API Gateway acts as the single entry point for client requests, abstracting the complexities of the microservices architecture from the client. It routes requests to the appropriate backend services based on the configured routes.

#### 4.3.2 Functionality:

**Routing:** Directs client requests to the corresponding microservices based on URL patterns or other request properties.

**Security:** Integrates with authentication services to enforce security policies, including authentication and authorization.

**Load Balancing:** Distributes incoming requests across multiple instances of a microservice, improving performance and reliability.

**Rate Limiting:** Controls the rate at which requests are processed, protecting the system from overload by limiting the number of requests per client within a specified time frame.

**Request Validation and Transformation:** Validates incoming requests and can transform them before forwarding them to the backend services.



## 4.4 Authentication Service

**4.4.1 Role:** Manages the authentication and authorization processes across the application, ensuring that only authorized users can access certain features and data.

### 4.4.2 Functionality:

**User Registration and Login:** Handles user registration and login, securely storing user credentials.

**JWT (JSON Web Token) Generation:** Issues JWTs upon successful authentication, which are then used to secure communications between the client and server.

**Role-Based Access Control (RBAC):** Enforces role-based access policies, ensuring that Admins and Employees have access only to the functionalities they are permitted to use.

**OAuth2 Support:** Optionally integrates with OAuth2 for external authentication providers, such as Google or GitHub, providing flexibility in authentication methods.

## 4.5 Employee Management Microservice

### Employee Directory:

criteria, such as department, job title, or performance metrics.

**4.5.1 Role:** This service is responsible for all CRUD operations related to employee data, including personal information, job titles, department affiliations, and roles within the organization.

### 4.5.2 Functionality:

**Employee Records Management:** Supports the creation, update, retrieval, and deletion of employee records, ensuring that the organization's employee data is always up-to-date.

**Role Management:** Allows Admins to assign and update roles for each employee, dictating their permissions and access levels within the system.

**Search and Filter:** Provides APIs for searching and filtering employee records based on various

**Integration with Other Services:** Seamlessly integrates with the Task and Performance Management microservices, ensuring that changes to employee roles or departments are reflected across the system.

Variables	Data Type
Id	Long
Name	String
Email	String
PhoneNumber	Long
JobRole	String
Salary	Double
DepartmentCode	String
PerformanceId	Long
TaskId	Long

## 4.6 Department Management Microservice

### Department Catalog:

**4.6.1 Role:** Manages department-related data, allowing for the organization and categorization of employees within specific departments.

#### 4.6.2 Functionality:

**Department CRUD Operations:** Provides APIs to create, update, delete, and retrieve department records, ensuring that the organizational structure is accurately represented.

**Employee Assignment:** Manages the assignment of employees to departments, facilitating clear organizational hierarchies and reporting structures.

**Departmental Analytics:** Offers analytics and reports on departmental performance, including headcount, productivity, and inter-departmental collaboration metrics.

Variables	Data Type
Id	Long
Name	String
Description	String
DepartmentCode	Long

### 4.7 Task Management Microservice

#### Task Assignment:

**4.7.1 Role:** This microservice is central to the management of tasks within the organization, enabling efficient task distribution and tracking.

#### 4.7.2 Functionality:

**Task Creation and Assignment:** Admins can create tasks and assign them to specific employees, setting deadlines and priorities to ensure timely completion.

**Progress Tracking:** Tracks the progress of tasks, allowing both Admins and Employees to monitor the status of each task in real-time.

**Task Notifications:** Sends notifications to employees when new tasks are assigned or when deadlines are approaching, helping to keep tasks on track.

**Integration with Performance Management:** Links task completion data to the Performance Management microservice, providing input for employee evaluations.

Variables	Data Type
Id	Long
Status	String
Description	String
Title	String

## 4.8 Performance Management Microservice

### Employee Performance Tracking:

. **4.8.1 Role:** Evaluates and tracks employee performance based on various metrics, including task completion, punctuality, and quality of work

### 4.8.2 Functionality:

**Performance Reviews:** Allows Admins and managers to conduct regular performance reviews, providing structured feedback to employees.

**Feedback Mechanism:** Employees can receive and respond to feedback, facilitating continuous improvement and career development.

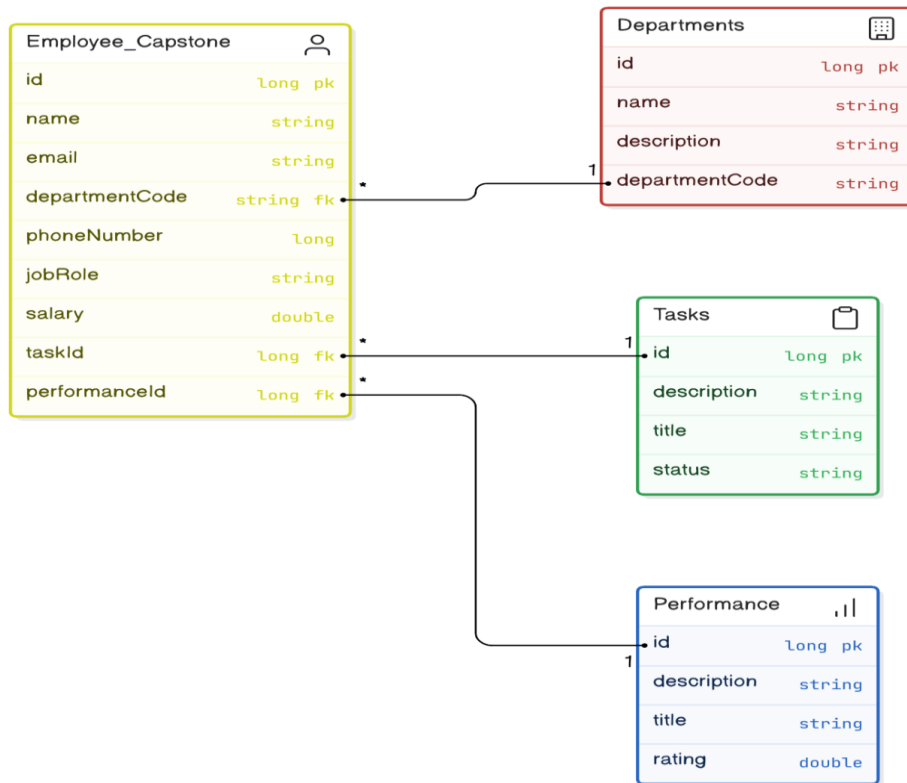
**Evaluation Reports:** Generates detailed reports based on employee performance data, which can be used for decisions on promotions, training, or disciplinary actions.

Variables	Data Type
Id	Long
Title	String
Description	String
Rating	Double

**Goal Setting and Tracking:** Admins can set performance goals for employees and track progress against these goals, aligning individual performance with organizational objectives.

**Note:**We can perform CRUD operation using Postman and Swagger.

## Class Diagram of microservices:



## 5. Project Flow

### 5.1 Admin Module

#### Admin Dashboard

**5.1.1 Role:** Acts as the centralized interface for Admins to manage the organization.

**5.1.2 Features:** Provides analytics, reports on employee performance, and departmental efficiency.

## **5.2 Employee Management**

**5.2.1 Role:** Enables Admins to perform CRUD operations on employee records.

**5.2.2 Features:** Role assignment, department organization, and employee record management.

## **5.3 Department Management**

**5.3.1 Role:** Allows Admins to manage departments by creating, viewing, editing, or deleting department records.

**5.3.2 Features:** Organizes employees within departments and manages departmental data.

## **5.4 Task Management**

**5.4.1 Role:** Enables Admins to create and assign tasks to employees, monitor progress, and update task statuses.

**5.4.2 Features:** Task tracking and monitoring of completion rates and deadlines.

## **5.5 Performance Management**

**5.5.1 Role:** Allows Admins to track and evaluate employee performance.

**5.5.2 Features:** Performance reviews, feedback provision, and generation of evaluation reports.

## **6. Employee Module**

### **Employee Registration & Authentication**

**6.1.1 Role:** Manages employee registration and login processes, ensuring secure access to the system.

**6.1.2 Features:** Profile management, access to assigned roles and departments.

## 6.2 Task Management

**6.2.1 Role:** Enables employees to view and manage assigned tasks.

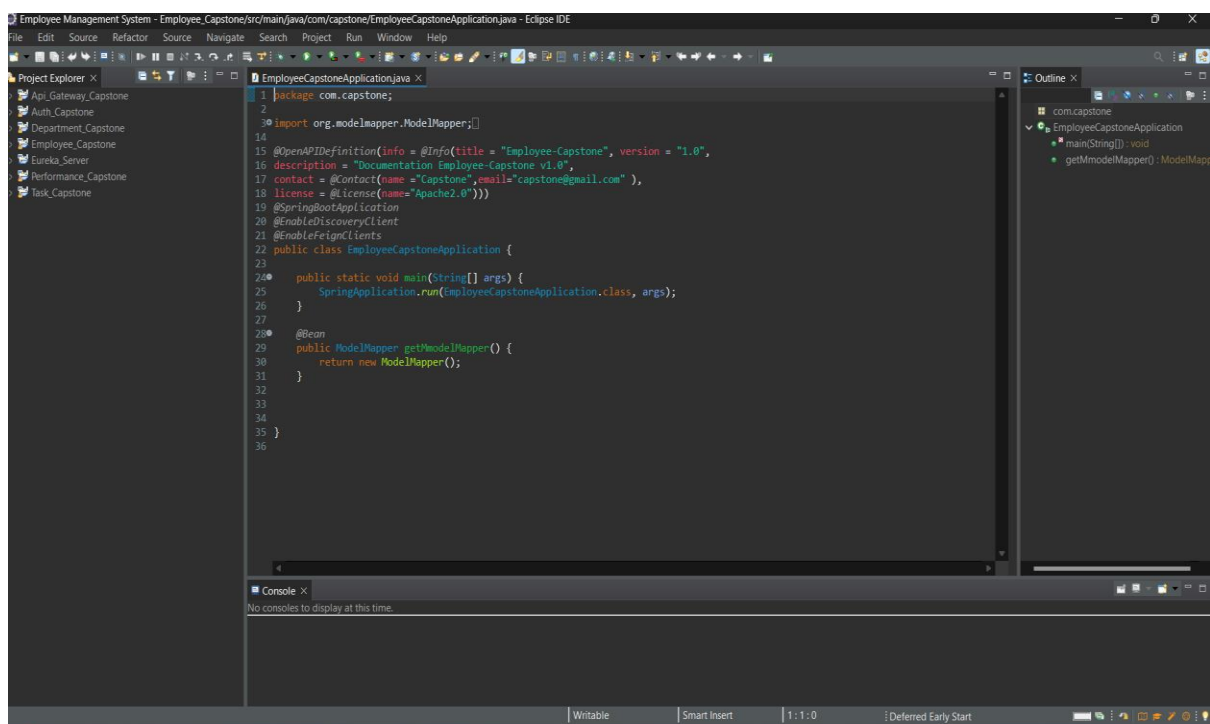
**6.2.2 Features:** Task progress tracking and workload management.

## 6.3 Performance Tracking

**6.3.1 Role:** Allows employees to view performance metrics and feedback.

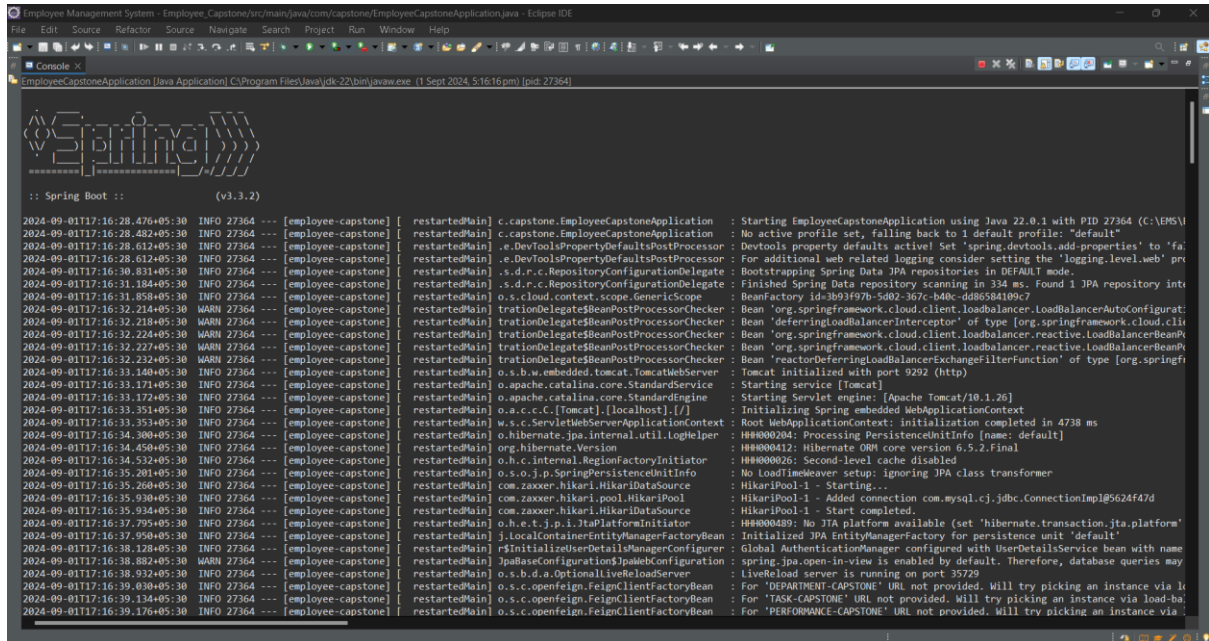
**6.3.2 Features:** Participation in performance reviews and tracking career development goals.

## 7. Eclipse Project Workspace-Backend:





## 8. Running the Application as SpringbootApp:



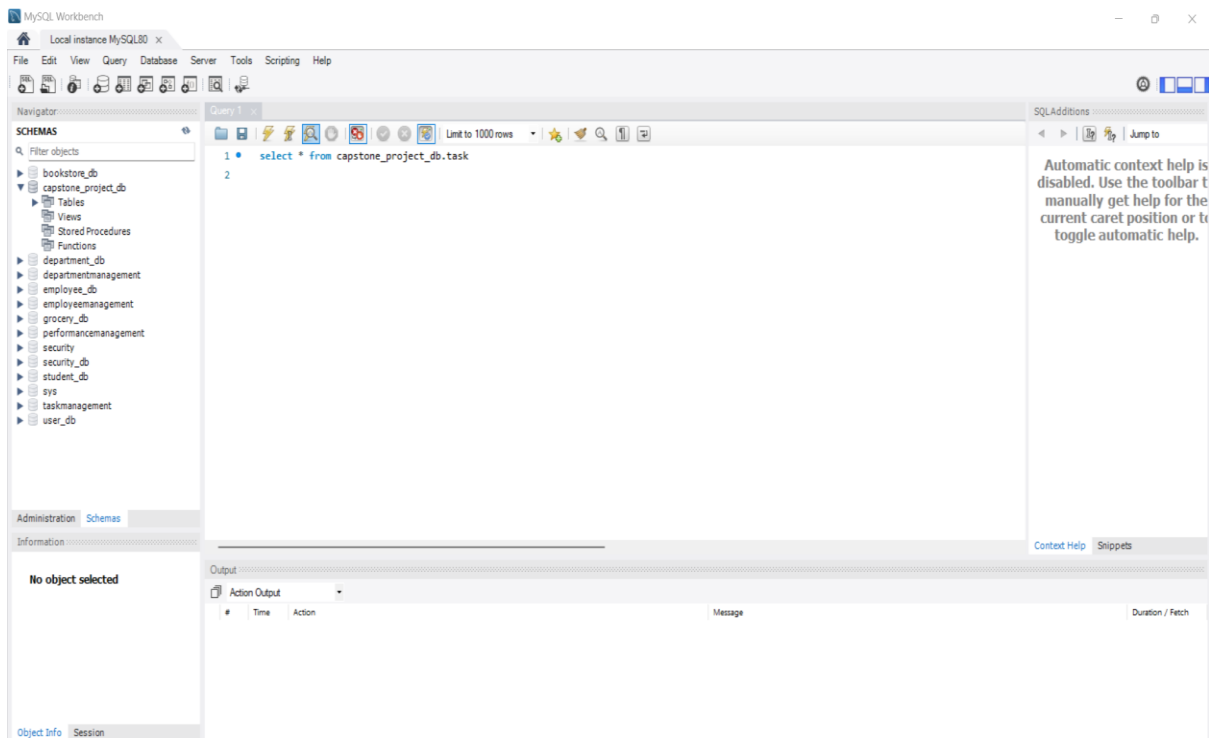
```
Employee Management System - Employee_Capstone\src\main\java\com\capstone\EmployeeCapstoneApplication.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help

EmployeeCapstoneApplication [Java Application] C:\Program Files\Java\jdk-22\bin\java.exe (1 Sept 2024, 5:16:16pm) [pid: 27364]

:: Spring Boot ::
(v3.3.2)

2024-09-01T17:16:28.476+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] c.capstone.EmployeeCapstoneApplication : Starting EmployeeCapstoneApplication using Java 22.0.1 with PID 27364 (C:\EMS\
2024-09-01T17:16:28.482+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] c.capstone.EmployeeCapstoneApplication : No active profile set, falling back to 1 default profile: "default"
2024-09-01T17:16:28.612+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'fa
2024-09-01T17:16:28.612+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' pr
2024-09-01T17:16:30.831+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-09-01T17:16:31.184+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 334 ms. Found 1 JPA repository int
2024-09-01T17:16:31.858+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.s.cloud.context.scope.GenericScope : BeanFactory id=3b93f97b-5d02-367c-b40c-dd86584109c7
2024-09-01T17:16:32.214+05:30 WARN 27364 --- [employee-capstone] [ restartedMain ] trationDelegateBeanPostProcessorChecker : Bean 'org.springframework.cloud.client.loadbalancer.LoadBalancerAutoConfigurat
2024-09-01T17:16:32.218+05:30 WARN 27364 --- [employee-capstone] [ restartedMain ] trationDelegateBeanPostProcessorChecker : Bean 'org.springframework.cloud.client.loadbalancer.reactive.LoadBalancerBeanP
2024-09-01T17:16:32.227+05:30 WARN 27364 --- [employee-capstone] [ restartedMain ] trationDelegateBeanPostProcessorChecker : Bean 'org.springframework.cloud.client.loadbalancer.reactive.LoadBalancerBeanP
2024-09-01T17:16:32.232+05:30 WARN 27364 --- [employee-capstone] [ restartedMain ] trationDelegateBeanPostProcessorChecker : Bean 'reactorDeferringLoadBalancerExchangeFilterFunction' of type [org.springfi
2024-09-01T17:16:33.148+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 9292 (http)
2024-09-01T17:16:33.171+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-09-01T17:16:33.172+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.26]
2024-09-01T17:16:33.351+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-09-01T17:16:33.353+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 4738 ms
2024-09-01T17:16:34.380+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.hibernate.jpa.internal.util.LogHelper : HH0000284: Processing PersistenceUnitInfo [name: default]
2024-09-01T17:16:34.450+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] org.hibernate.Version : HH0000412: Hibernate ORM core version 5.5.2.Final
2024-09-01T17:16:34.532+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.h.c.internal.RegionFactoryInitiator : HH000026: Second-level cache disabled
2024-09-01T17:16:35.201+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2024-09-01T17:16:35.268+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-09-01T17:16:35.930+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@5624f47d
2024-09-01T17:16:35.924+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-09-01T17:16:37.795+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.h.e.t.j.p.i.JtaPlatformInitiator : HH0000489: No JTA platform available (set 'hibernate.transaction.jta.platform'
2024-09-01T17:16:37.950+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-09-01T17:16:38.128+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] r.InitializeUserDetailsServiceConfigurer : Global AuthenticationManager configured with UserDetailsService bean with name
2024-09-01T17:16:38.880+05:30 WARN 27364 --- [employee-capstone] [ restartedMain ] JpaBasicConfigurationJpaMethodConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may
2024-09-01T17:16:38.932+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2024-09-01T17:16:39.030+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.s.c.openfeign.FeignClientFactoryBean : For 'DEPARTMENT-CAPSTONE' URL not provided. Will try picking an instance via lo
2024-09-01T17:16:39.134+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.s.c.openfeign.FeignClientFactoryBean : For 'TASK-CAPSTONE' URL not provided. Will try picking an instance via load-ba
2024-09-01T17:16:39.176+05:30 INFO 27364 --- [employee-capstone] [ restartedMain ] o.s.c.openfeign.FeignClientFactoryBean : For 'PERFORMANCE-CAPSTONE' URL not provided. Will try picking an instance via
```

## 9. Data Base MySQL Workbench:



## 9.1 Employee Table in MySQL:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'capstone\_project\_db' selected. The main query editor contains the SQL query: `select * from capstone_project_db.employee`. The 'Result Grid' shows 5 rows of employee data. The 'Output' pane at the bottom shows the execution message: '1 16:45:30 select \* from capstone\_project\_db.employee LIMIT 0, 1000' with a message '5 row(s) returned' and a duration of '0.015 sec / 0.000 sec'.

id	department_code	email	job_role	name	performance_id	phone_number	salary	task_id
1	101	rs@gmail.com	HR	ram	77	7894561450	35000	352
2	103	sai@gmail.com	SAP	sai	78	6894561450	35000	354
3	102	rv@gmail.com	Fullstack	vasr	88	7294561450	45000	564
4	102	hlm@gmail.com	Fullstack	hlm	88	7294561470	45000	564
5	105	aap@gmail.com	tester	aappa	89	7894561211	52500	458

## 9.2 Department Table in MySQL

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'capstone\_project\_db' selected. The main query editor contains the SQL query: `select * from capstone_project_db.department`. The 'Result Grid' shows 7 rows of department data. The 'Output' pane at the bottom shows the execution message: '3 01:09:17 select \* from capstone\_project\_db.department LIMIT 0, 1000' with a message '7 row(s) returned' and a duration of '0.016 sec / 0.000 sec'.

id	department_code	description	name
2	101	department deals with hiring procedd	HR
3	102	department deals with java	Fullstack
4	103	department deals with Security	SAP
5	159	department deals with networks	Networking
6	159	department deals with networks	Networking
7	string	string	string
8	157	department deals with Management	Management

## 9.3 Performance Table in MySQL

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'capstone\_project\_db' expanded, showing tables like 'department', 'employee', 'performance', 'task', and 'users'. The main query editor contains the query: `select * from capstone_project_db.performance`. The 'Result Grid' shows the following data:

id	description	rating	title
1	works with java	3	Associate
2	works with management	4	HR
3	works with developers	4	SAP
4	string	5	string
5	works with developers	4	SAP

The 'Output' pane at the bottom shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
1	16:45:30	select * from capstone_project_db.employee LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec
2	01:06:19	select * from capstone_project_db.employee LIMIT 0, 1000	7 row(s) returned	0.078 sec / 0.000 sec
3	01:09:17	select * from capstone_project_db.department LIMIT 0, 1000	7 row(s) returned	0.016 sec / 0.000 sec
4	01:10:58	select * from capstone_project_db.performance LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec

## 9.4 Tasks Table in MySQL

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'capstone\_project\_db' expanded, showing tables like 'department', 'employee', 'performance', 'task', and 'users'. The main query editor contains the query: `select * from capstone_project_db.task`. The 'Result Grid' shows the following data:

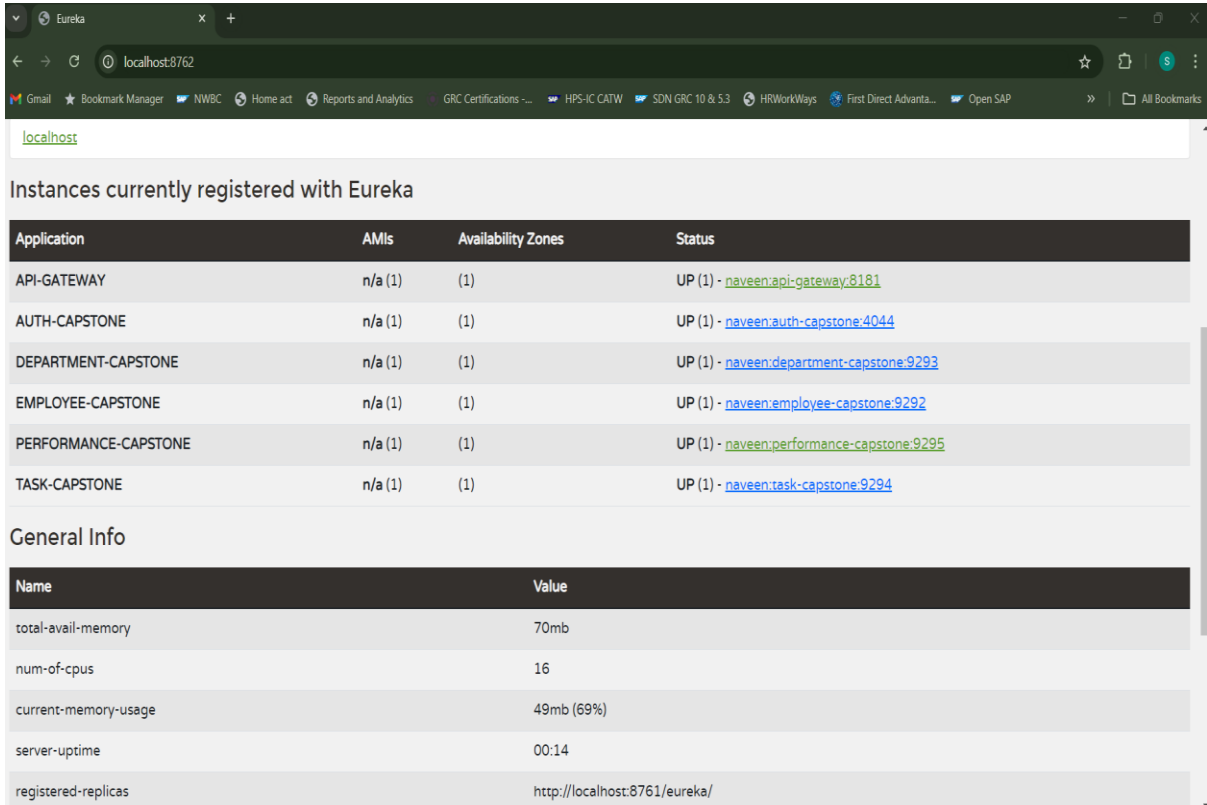
id	description	status	title
1	coding	In Review	Implementation
2	debugging	Invited	Testing
3	access prevention	Completed	SAP
4	accessing networks	Completed	Networking
5	string	string	string
6	accessing networks	Completed	Networking
7	accessing networks	Completed	Networking

The 'Output' pane at the bottom shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
1	16:45:30	select * from capstone_project_db.employee LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec
2	01:06:19	select * from capstone_project_db.employee LIMIT 0, 1000	7 row(s) returned	0.078 sec / 0.000 sec
3	01:09:17	select * from capstone_project_db.department LIMIT 0, 1000	7 row(s) returned	0.016 sec / 0.000 sec
4	01:10:58	select * from capstone_project_db.performance LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
5	01:12:29	select * from capstone_project_db.task LIMIT 0, 1000	7 row(s) returned	0.016 sec / 0.000 sec

## 10. Eureka Server page:

All micorservices register on Eureka Server.



The screenshot shows the Eureka Server web interface in a browser. The address bar shows 'localhost:8762'. The page title is 'localhost'. Below the title, it says 'Instances currently registered with Eureka'. There is a table with 4 columns: Application, AMIs, Availability Zones, and Status. The table lists six applications: API-GATEWAY, AUTH-CAPSTONE, DEPARTMENT-CAPSTONE, EMPLOYEE-CAPSTONE, PERFORMANCE-CAPSTONE, and TASK-CAPSTONE. Each application has 'n/a (1)' for AMIs, '(1)' for Availability Zones, and 'UP (1)' for Status, with a link to the instance details. Below the table, there is a 'General Info' section with a table showing various system metrics.

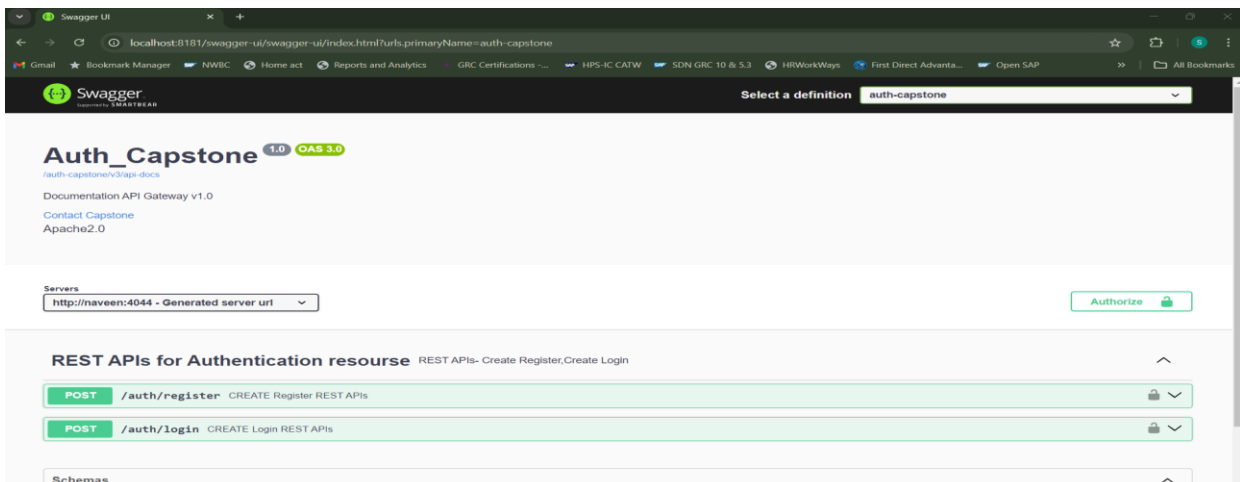
Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - <a href="#">naveen:api-gateway:8181</a>
AUTH-CAPSTONE	n/a (1)	(1)	UP (1) - <a href="#">naveen:auth-capstone:4044</a>
DEPARTMENT-CAPSTONE	n/a (1)	(1)	UP (1) - <a href="#">naveen:department-capstone:9293</a>
EMPLOYEE-CAPSTONE	n/a (1)	(1)	UP (1) - <a href="#">naveen:employee-capstone:9292</a>
PERFORMANCE-CAPSTONE	n/a (1)	(1)	UP (1) - <a href="#">naveen:performance-capstone:9295</a>
TASK-CAPSTONE	n/a (1)	(1)	UP (1) - <a href="#">naveen:task-capstone:9294</a>

General Info

Name	Value
total-avail-memory	70mb
num-of-cpus	16
current-memory-usage	49mb (69%)
server-uptime	00:14
registered-replicas	<a href="#">http://localhost:8761/eureka/</a>

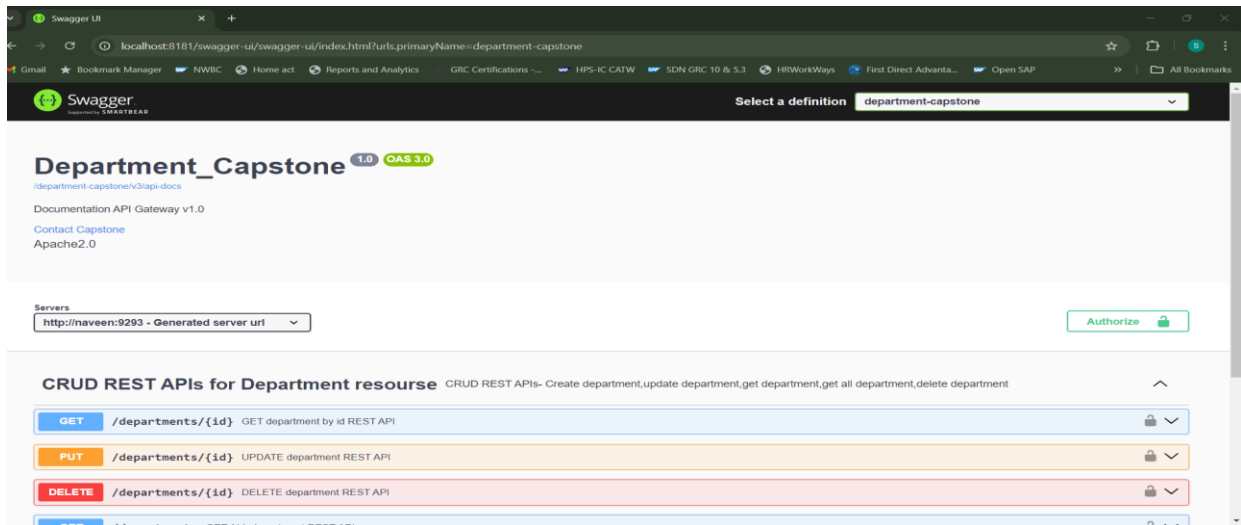
## 11. Swagger UI:

### 11.1 Swagger UI of Auth\_Capstone

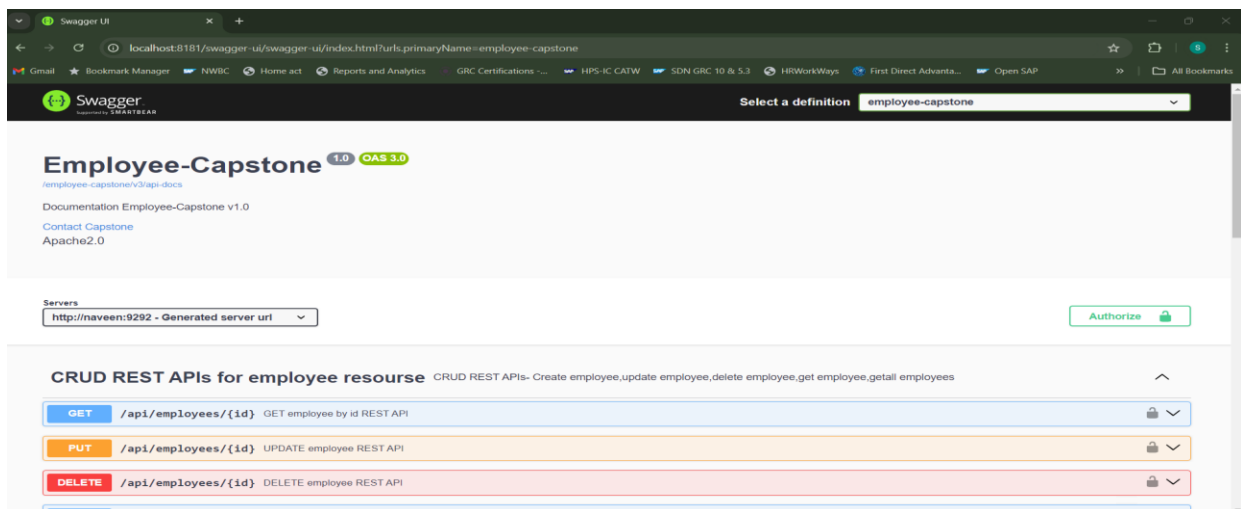


The screenshot shows the Swagger UI for the 'Auth\_Capstone' API. The browser address bar shows 'localhost:8181/swagger-ui/swagger-ui/index.html?urls.primaryName=auth-capstone'. The Swagger logo is in the top left. The title is 'Auth\_Capstone' with version '1.0' and 'OAS 3.0' tags. Below the title, it says 'Documentation API Gateway v1.0' and 'Contact Capstone'. There is a 'Servers' section with a dropdown menu showing 'http://naveen:4044 - Generated server url' and an 'Authorize' button. Below the servers, there is a section for 'REST APIs for Authentication resource' with two endpoints: 'POST /auth/register' and 'POST /auth/login'. Each endpoint has a 'CREATE' button and a 'REST APIs' button. At the bottom, there is a 'Schemas' section.

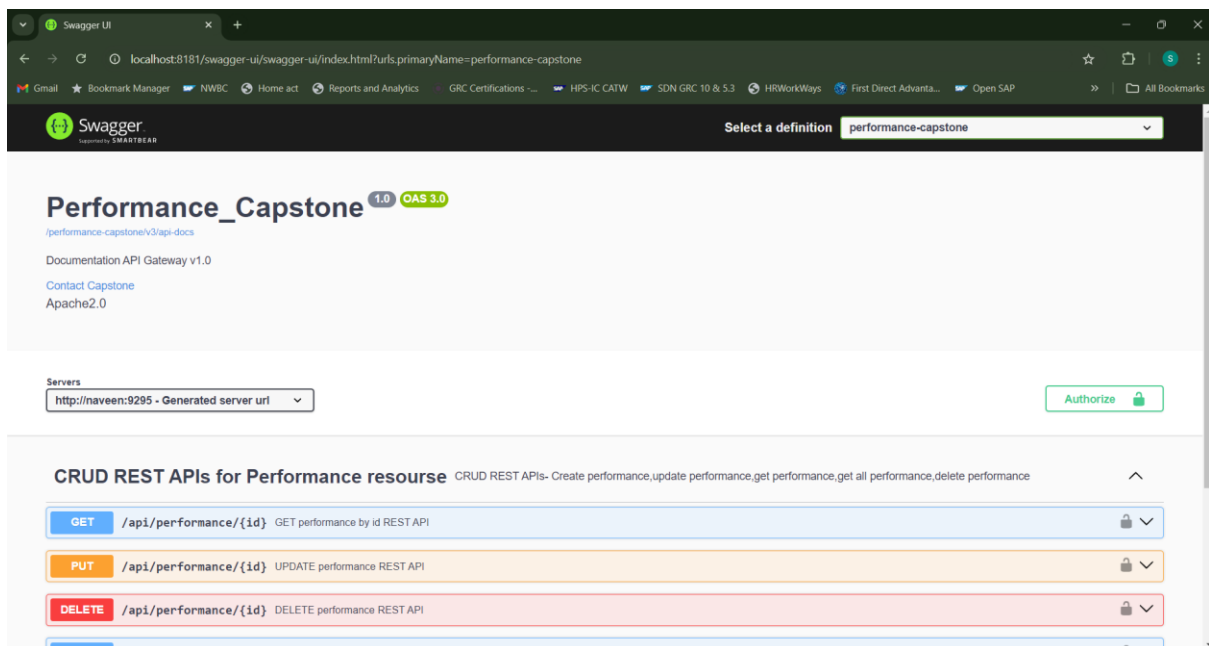
## 11.2 Swagger UI of Department\_Capstone



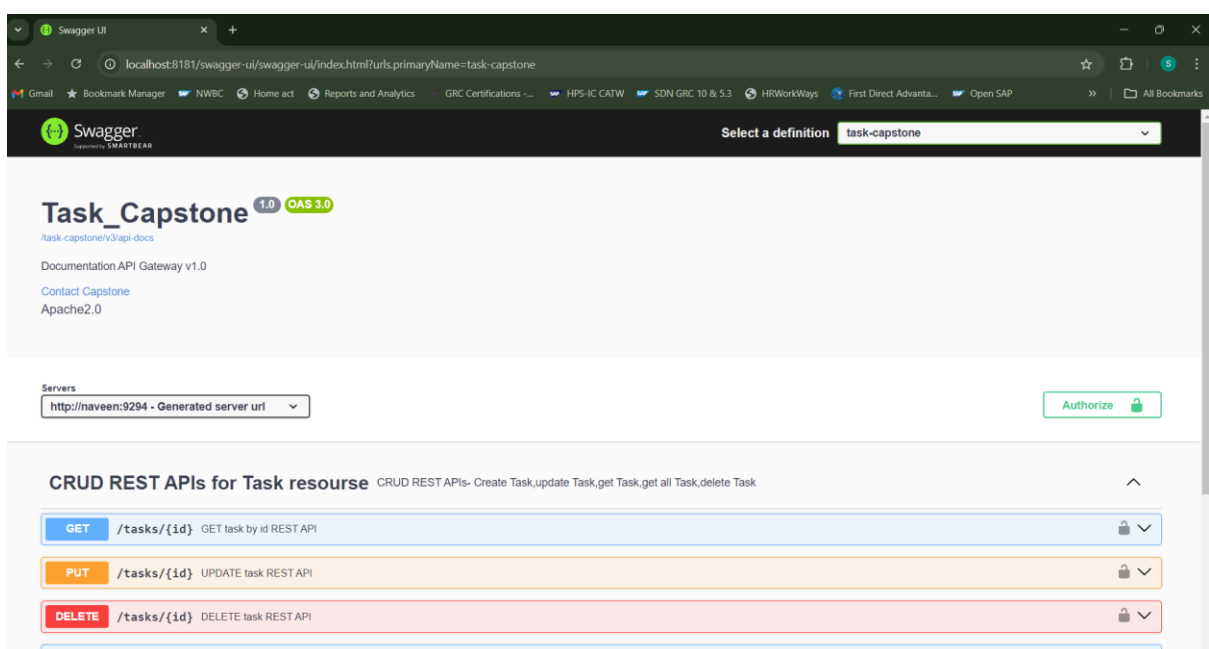
## 11.3 Swagger UI of Employee\_Capstone



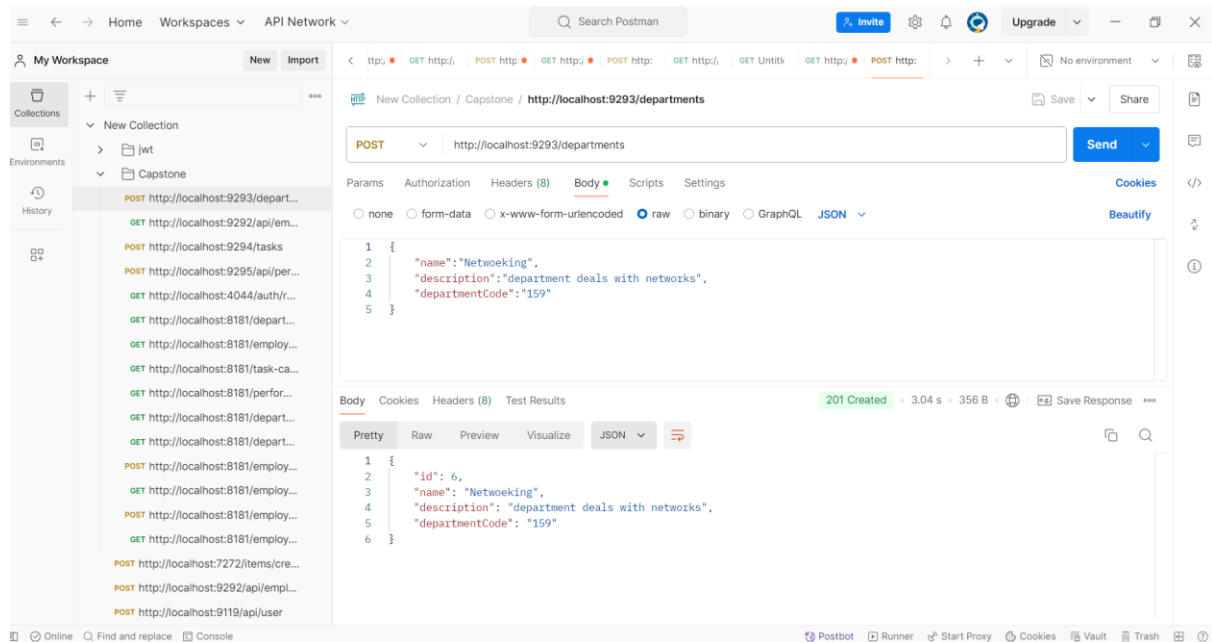
## 11.4 Swagger UI of Performance\_Capstone



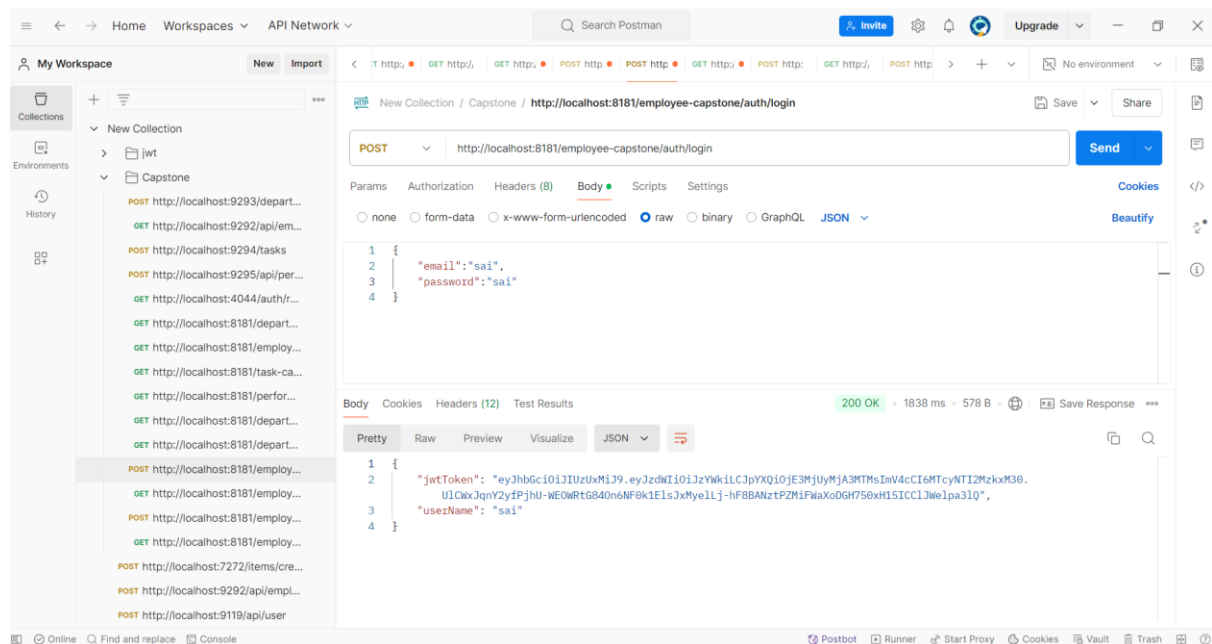
## 11.5 Swagger UI of Task\_Capstone



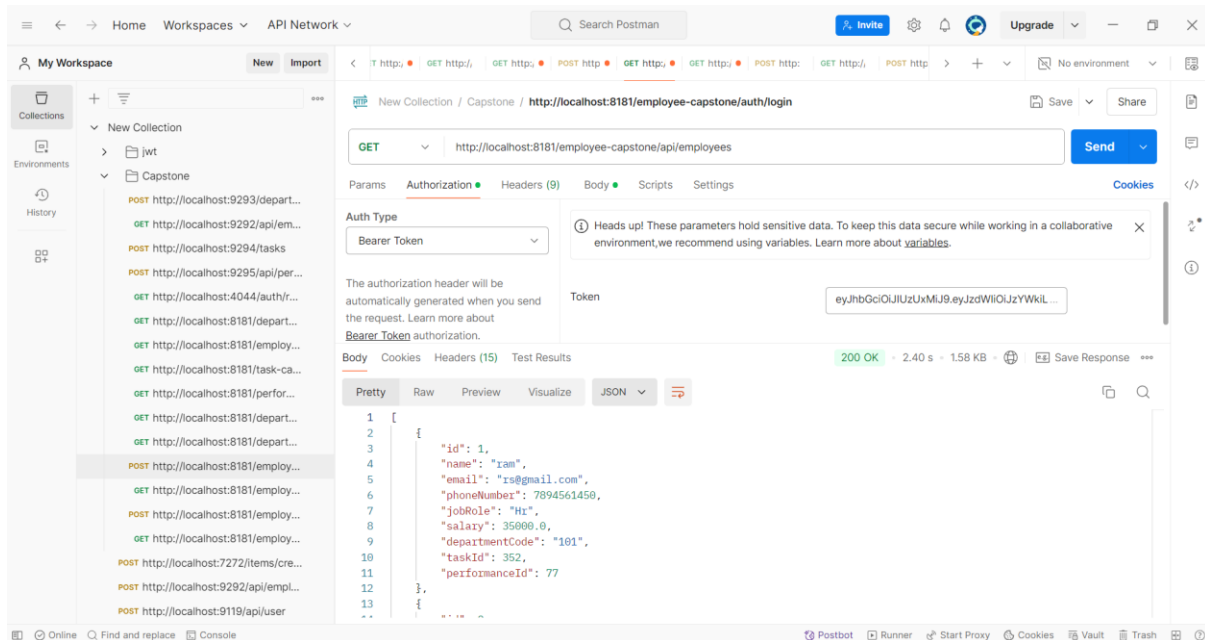
## 12. Postman API:



## 12.1 Authentication for Employee Microservice:



## 12.2 Fetching Employee data using jwtToken:

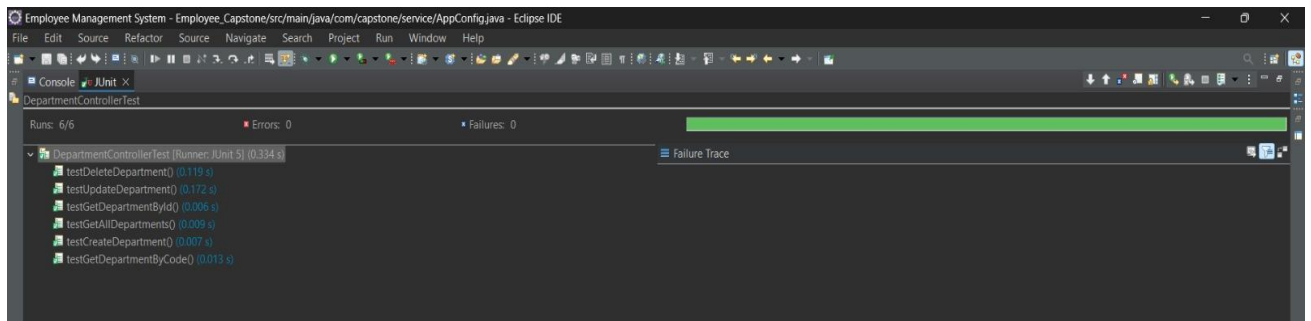


## 13. Testing and Refinement

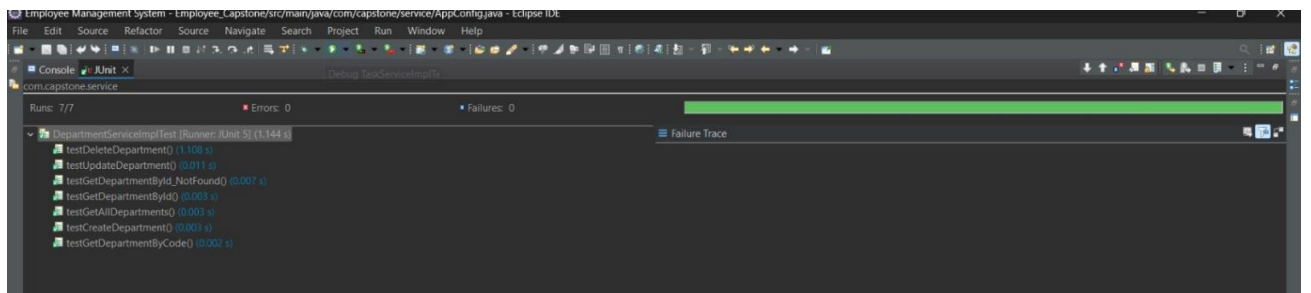
- To ensure the reliability and quality of the Employee Management System, comprehensive testing and refinement were conducted using JUnit.
- Each microservice (Employee Management, Department Management, Task Management, Performance Management) was independently tested using JUnit to validate the functionality of individual components.
- Test cases included scenarios for creating, updating, retrieving, and deleting records (CRUD operations).



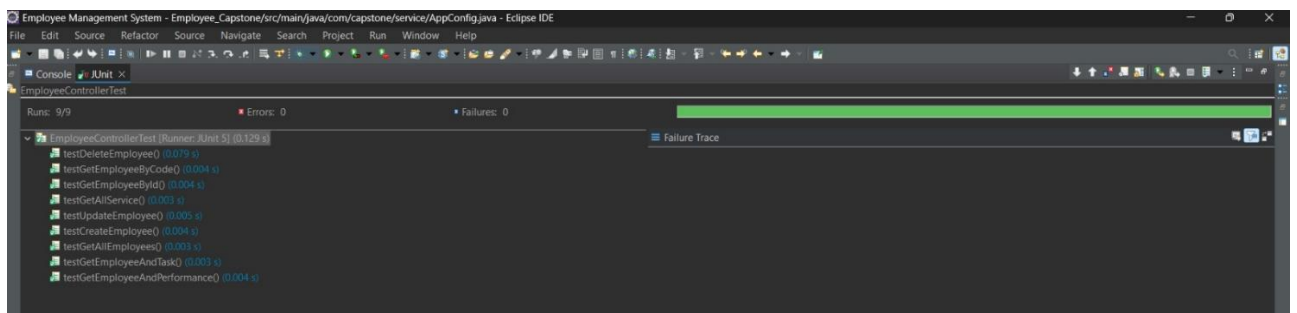
## 13.1 JUnit for Department\_Capstone: Controller Layer



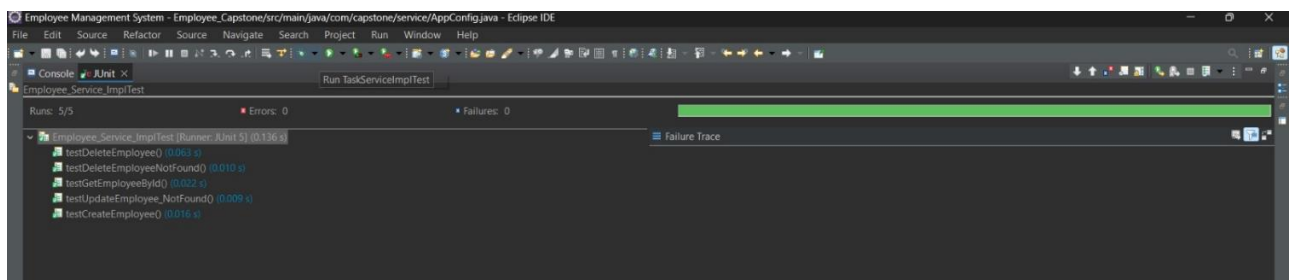
## Service Layer



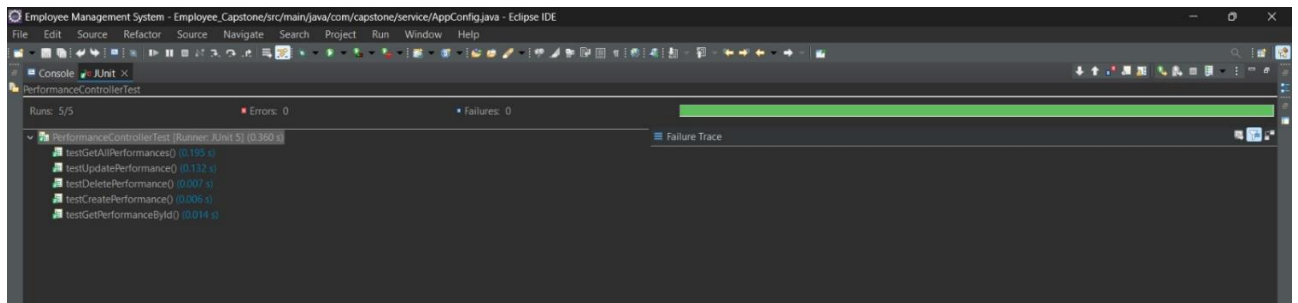
## 13.2 JUnit for Employee\_Capstone. Controller Layer



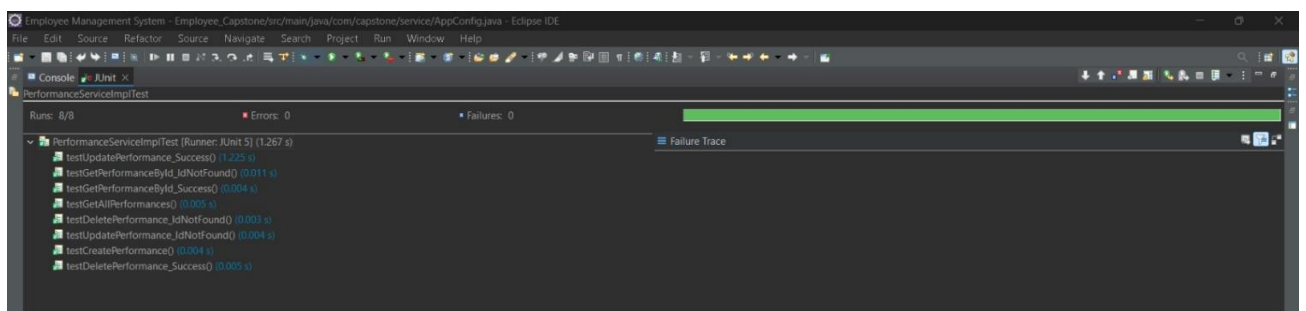
## Service Layer



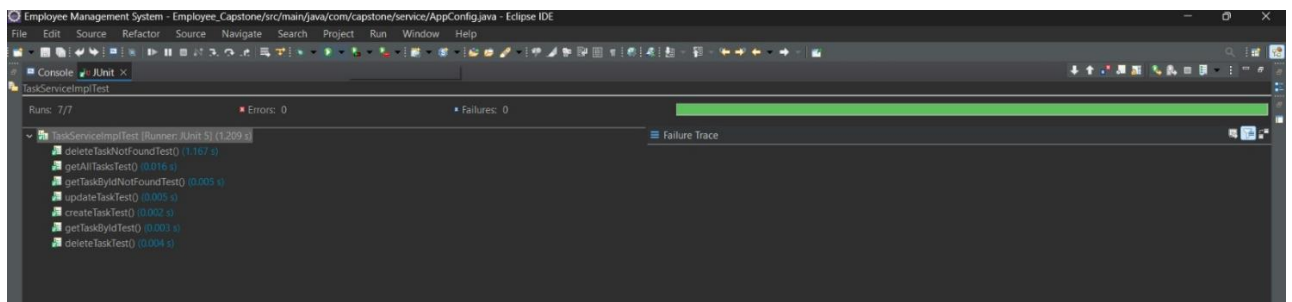
## 13.3 JUnit for Employee\_Capstone: Controller Layer



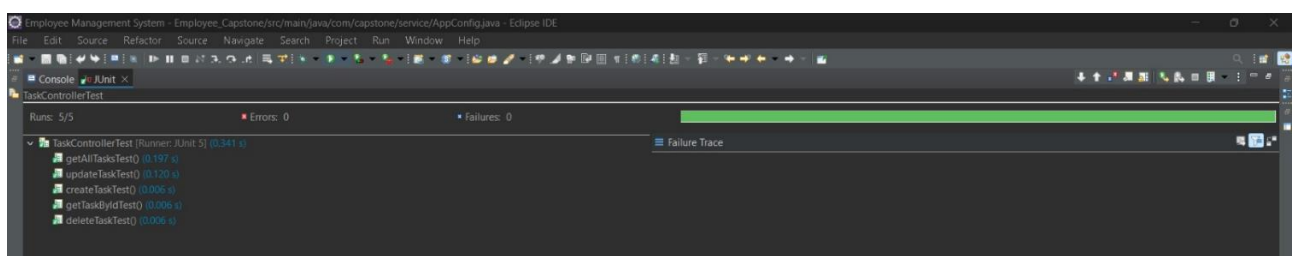
## Service Layer



## 13.4 JUnit for Task\_Capstone. Service Layer



## Controller Layer



## 14. Conclusion:

The Employee Management System project effectively showcases the advantages of using a microservices architecture for scalability and modularity. By utilizing Java, Spring Boot, Spring Cloud, and MySQL/MariaDB, the system efficiently manages employees, departments, tasks, and performance evaluations. It provides secure access via JWT-based authentication, ensuring data integrity and confidentiality. This system lays a solid foundation for further enhancements, including advanced analytics, mobile integration, and machine learning, making it a robust solution for modern human resource management. Employee Management System project effectively showcases the advantages of using a microservices architecture for scalability and modularity. By utilizing Java, Spring Boot, Spring Cloud, and MySQL/MariaDB, the system efficiently manages employees, departments, tasks, and performance evaluations. It provides secure access via JWT-based authentication, ensuring data integrity and confidentiality. This system lays a solid foundation for further enhancements, including advanced analytics, mobile integration, and machine learning, making it a robust solution for modern human resource management.

## 15.Future Enhancement

- **Advanced Analytics:** Integrate advanced analytics for deeper insights into employee performance and organizational efficiency.
- **Mobile Access:** Develop mobile applications for enhanced accessibility and a better user experience.
- **Machine Learning:** Implement predictive analytics to forecast employee performance trends.

- **HR System Integration:** Connect with payroll, attendance, and other HR systems for comprehensive management.
- **Real-Time Notifications:** Add real-time notifications for task updates and performance feedback.