Alfido Tech Internship – Task 3

Unit Testing & Logging in Spring Boot
Name: Revathi S
Role: Java Developer Intern
Task Number: Task 3
Organization: Alfido Tech
Technology: Java, Spring Boot

1. Task Overview
The objective of Task 3 is to implement unit testing for service layer classes and configure logging in a Spring Boot application. This task focuses on improving code quality, validating business logic using test cases, and ensuring proper logging of important application events.

2. Technologies and Tools Used
The following technologies and tools were used to complete this task:
Java
Spring Boot
JUnit 5
Mockito
Maven
SLF4J
Logback
Git & GitHub
Visual Studio Code

3. Project Description
This Spring Boot project is based on an Employee Management system. The service layer handles business logic related to employee operations. Unit tests were written to validate service methods independently by mocking repository dependencies.
Flyway migrations were used to manage database schema creation and sample data insertion.

4. Unit Testing Implementation
Unit testing was implemented for the service layer using JUnit 5 and Mockito.
Key Highlights:
Mockito is used to mock repository dependencies
JUnit annotations such as @Test and @ExtendWith are used
Service logic is tested independently without database interaction


eServiceImpl.java 3
EmployeeServiceImplTest.java
EmployeeController.java
app
employeeapi > src > test > java > com > example > employeeapi > service > EmployeeServiceImplTes
1 package com.example.employeeapi.service;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.mockito.Mockito.verify;
5 import static org.mockito.Mockito.when;
6
7 import java.util.Arrays;
8 import java.util.List;
9 import java.util.Optional;
10
11 import com.example.employeeapi.entity.Employee;
12 import com.example.employeeapi.repository.EmployeeRepository;
13
14 import org.junit.jupiter.api.Test;
15 import org.mockito.InjectMocks;
16 import org.mockito.Mock;
17 import org.springframework.boot.test.context.SpringBootTest;
18
19 @SpringBootTest
20 public class EmployeeServiceImplTest {
21
22 @Mock
23 private EmployeeRepository repository;
24
25 @InjectMocks
26 private EmployeeServiceImpl employeeService;
27
28 @Test
29 void testGetAllEmployees() {
30 Employee emp1 = new Employee();
31 emp1.setId(id: 1L);
32 emp1.setName(name: "Revathi");
33 emp1.setRole(role: "Developer");
34
35 Employee emp2 = new Employee();
employeeapi main* 0 3 Java: Ready


oyeeServiceImpl.java 3
EmployeeServiceImplTest.java
EmployeeController.java
employeeapi > src > test > java > com > example > employeeapi > service > EmployeeServiceImplTest.java
20 public class EmployeeServiceImplTest {
29 void testGetAllEmployees() {
40 when(repository.findAll()).thenReturn(Arrays.asList(emp1, emp2));
41
42 List<Employee> result = employeeService.getAllEmployees();
43
44 assertEquals(2, result.size());
45 }
46
47 @Test
48 void testGetEmployeeById() {
49 Employee emp = new Employee();
50 emp.setId(id: 1L);
51 emp.setName(name: "Revathi");
52 emp.setRole(role: "Developer");
53
54 when(repository.findById(1L)).thenReturn(Optional.of(emp));
55
56 Employee result = employeeService.getEmployeeById(id: 1L);
57
58 assertEquals("Revathi", result.getName());
59 }
60
61 @Test
62 void testCreateEmployee() {
63 Employee emp = new Employee();
64 emp.setName(name: "Revathi");
65 emp.setRole(role: "Developer");
66
67 when(repository.save(emp)).thenReturn(emp);
68
69 Employee saved = employeeService.createEmployee(emp);
70
71 assertEquals("Developer", saved.getRole());
72 }
73
74 @Test
75 void testDeleteEmployee() {
76 employeeService.deleteEmployee(id: 1L);
77 verify(repository).deleteById(1L);
78 }
79 }
employeeapi main* 0 3 Java: Ready

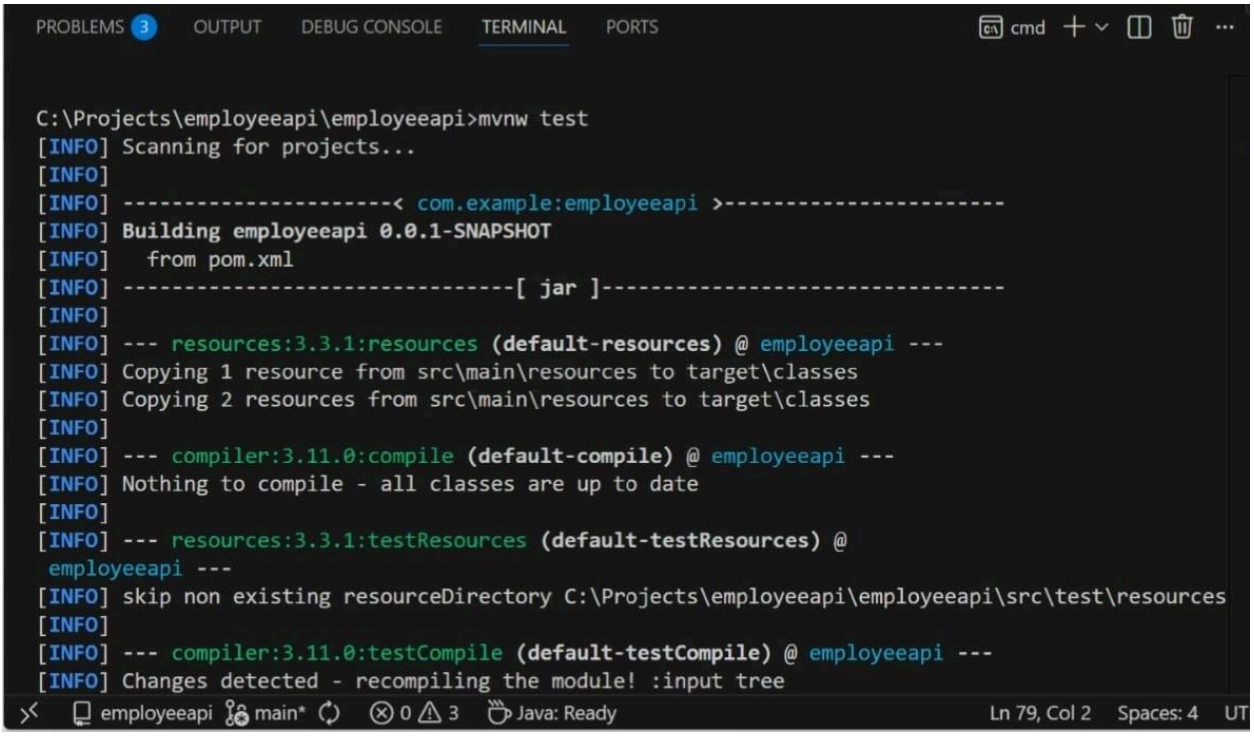
5. Logging Configuration
Logging was configured using SLF4J with Logback to capture important application events.
Logging Features:
Informational logs for service operations
Error logs for exception handling
Console logging configured via Logback

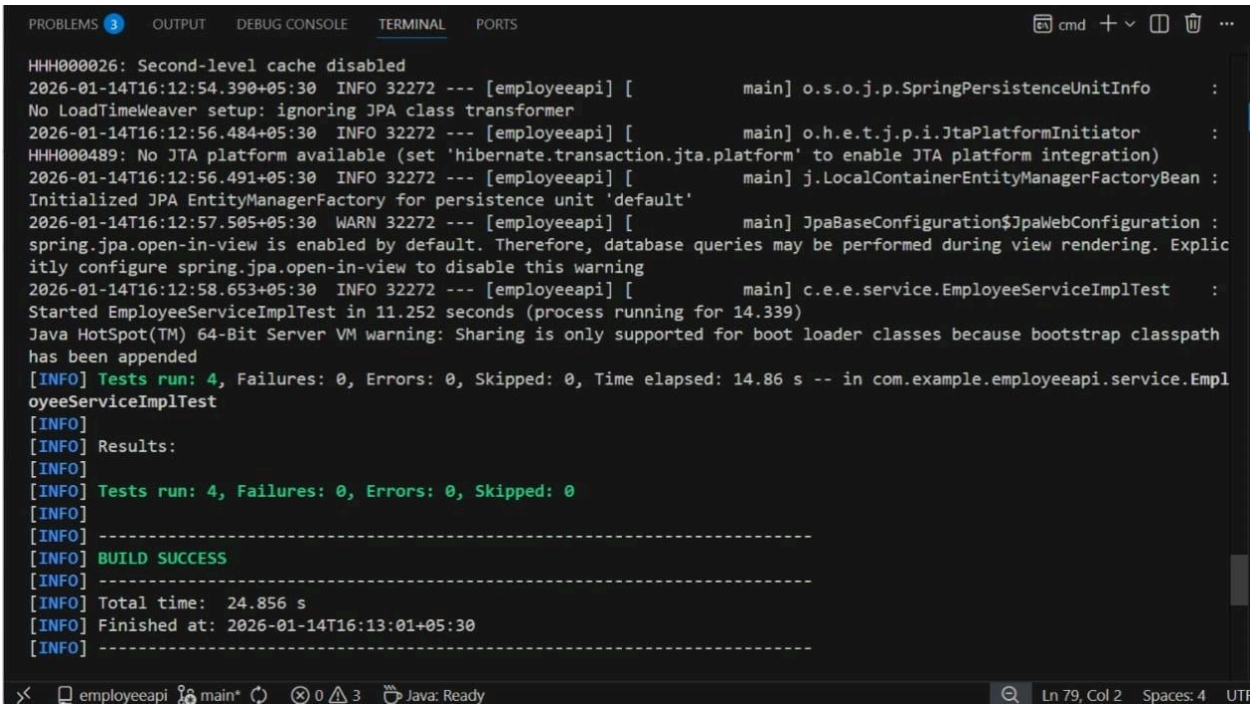
Sample Logback Configuration:

Xml
```
<root level="INFO">
    <appender-ref ref="CONSOLE"/>
</root>
```

This ensures better monitoring and debugging of application behavior.
6. Database Migration
Flyway was used for database migration and version control.
Migration Files:
V1_CreateEmployeeTable.sql – Creates the employee table
V2_InsertSampleData.sql – Inserts sample employee data

7. Test Execution and Results
Unit tests were executed using Maven.
Command Used:
mvn test





Terminal output showing successful test execution.

8. GitHub Repository Link
The complete source code for Task 3 is available at the following GitHub repository:
🔗 GitHub Link:

https://github.com/Revathi2006-op/Alfido-Tech-Internship-Task-3

9. Conclusion
Through Task 3, I gained hands-on experience in:
Writing unit tests using JUnit and Mockito
Mocking dependencies for isolated testing
Configuring logging using SLF4J and Logback
Managing database migrations using Flyway
Improving application reliability and maintainability
This task strengthened my understanding of professional Spring Boot development practices.