**End-to-End Microservices Case Study: Digital Complaint Box**

**Project Overview**

Develop a Digital Complaint Box platform that allows users (citizens, employees, or customers) to submit, track, escalate, and resolve complaints using microservices architecture. This system enables issue categorization, role-based routing, response analytics, and notification services.

**Sprint 1: Database & Java Backend Fundamentals**

**Sprint Goal: Lay the groundwork by creating normalized schema, domain models, and database integration using JDBC.**

**Tasks and User Stories**

| **Member** | **User Story** | **Task Description** |
| --- | --- | --- |
| 1 | As a DB designer, I want to define tables for complaints, users, departments, categories, and responses. | Design normalized schema with appropriate constraints, indexing, and foreign keys. |
| 2 | As a SQL developer, I want to implement queries to log and retrieve complaint records. | Write DML, DQL, JOINS, subqueries, and rollback mechanisms. |
| 3 | As a Java developer, I want to model complaint data and user entities. | Implement POJOs using OOP, abstract classes, constructors, wrapper classes, and collections. |
| 4 | As a backend developer, I want to integrate JDBC for CRUD operations. | Connect Java to DB using JDBC, perform inserts, updates, deletes, and result processing. |
| 5 | As a concurrency tester, I want to simulate multiple complaint submissions. | Use threads to mimic concurrent user complaints and ensure thread safety. |
| 6 | As a logger, I want to capture input/output logs and exceptions. | Use Scanner, try-catch, SLF4J for structured logging. |

**Deliverables**

* SQL scripts
* Java domain classes
* JDBC CRUD code
* Thread simulation logs

**Evaluation Rubric**

| **Criteria** | **Weight** |
| --- | --- |
| Schema Design | 25% |
| Java Object Modeling | 20% |
| JDBC Functionality | 20% |
| Multithreading Simulation | 20% |
| Logging & Validation | 15% |

**Sprint 2: Service Design and Web API Development**

**Sprint Goal: Design the system architecture and implement complaint submission and tracking APIs.**

**Tasks and User Stories**

| **Member** | **User Story** | **Task Description** |
| --- | --- | --- |
| 1 | As a software architect, I want to define microservice boundaries and interactions. | Create UML diagrams (use case, sequence, component, class) for complaint lifecycle. |
| 2 | As a SOAP developer, I want to implement department info as a SOAP service. | Create JAX-WS service for department lookup with WSDL access. |
| 3 | As a REST developer, I want to build APIs for complaint submission and status tracking. | Create JAX-RS REST services for POST and GET endpoints with JSON payloads. |
| 4 | As a security engineer, I want to protect APIs using basic auth or tokens. | Add JWT or Basic authentication filters. |
| 5 | As a system deployer, I want to host services using web server. | Deploy SOAP/REST services on Tomcat/Jetty. |
| 6 | As a QA tester, I want to validate service accuracy. | Use Postman and SOAP UI to test endpoints, status codes, and data. |

**Deliverables**

* UML design files
* REST and SOAP APIs
* Authentication filter logic
* Deployed services
* Test scripts and results

**Evaluation Rubric**

| **Criteria** | **Weight** |
| --- | --- |
| Architecture & Design | 20% |
| API Quality & Functionality | 25% |
| Authentication Implementation | 20% |
| Deployment Readiness | 20% |
| API Testing | 15% |

**Sprint 3: Spring Boot Microservices & AI-Driven Routing**

**Sprint Goal: Modularize services using Spring Boot and add AI-based complaint category prediction.**

**Tasks and User Stories**

| **Member** | **User Story** | **Task Description** |
| --- | --- | --- |
| 1 | As a developer, I want to build Complaint Service using Spring Boot. | Create REST APIs for create, update, and view complaints using Spring Data JPA. |
| 2 | As a dev, I want to build User Service for registration and authentication. | Implement secure user management with role-based access. |
| 3 | As a dev, I want to create Category & Department Service. | Provide APIs to get categories and resolve responsible departments. |
| 4 | As a DevOps engineer, I want to enable service health checks and logs. | Integrate Actuator endpoints and SLF4J logging for all services. |
| 5 | As an AI integrator, I want to create a GenAI service that predicts complaint categories. | Mock a GenAI API that receives complaint text and returns a category suggestion. |
| 6 | As a data engineer, I want to support XML-based complaint export. | Use JAXB to convert complaint data to XML, with schema validation. |

**Deliverables**

* Spring Boot services: Complaint, User, Category
* GenAI Category Suggestion Service
* XML export logic
* Monitoring endpoints and logging

**Evaluation Rubric**

| **Criteria** | **Weight** |
| --- | --- |
| Spring Boot Implementation | 30% |
| AI Integration & Response Accuracy | 20% |
| XML Handling & Schema Validation | 15% |
| Logging & Monitoring | 15% |
| Documentation & Swagger | 20% |

**Sprint 4: Messaging, Containerization, and CI/CD**

**Sprint Goal: Integrate messaging for escalation, containerize services, and automate deployments.**

**Tasks and User Stories**

| **Member** | **User Story** | **Task Description** |
| --- | --- | --- |
| 1 | As a container engineer, I want to Dockerize all services. | Write Dockerfiles and compose files to run services locally. |
| 2 | As a K8s engineer, I want to deploy the platform on a Kubernetes cluster. | Use manifests for deployments, services, secrets, and configmaps. |
| 3 | As a messaging dev, I want to notify departments on high-priority complaints. | Use JMS (e.g., ActiveMQ) to publish messages for escalations. |
| 4 | As a gateway developer, I want to centralize API access. | Configure Spring Cloud Gateway to route requests to respective microservices. |
| 5 | As a DevOps engineer, I want to automate build and deploy. | Setup GitHub Actions/Jenkins pipeline for Docker build, test, and deploy to K8s. |
| 6 | As a demo lead, I want to run an end-to-end presentation. | Coordinate a final demo, submit documentation, and validate system performance. |

**Deliverables**

* Docker images & compose files
* K8s YAMLs
* JMS messaging system
* API Gateway config
* CI/CD pipeline
* Final demo presentation

**Evaluation Rubric**

| **Criteria** | **Weight** |
| --- | --- |
| Containerization Quality | 30% |
| Messaging Integration | 20% |
| Gateway Setup | 20% |
| CI/CD Pipelines | 15% |
| Demo & Docs | 15% |

**Final Evaluation Summary**

| **Category** | **Description** | **Weight** |
| --- | --- | --- |
| System Architecture | Modular, scalable, and API-driven design | 30% |
| Code Quality & Best Practices | Readable, maintainable, well-tested code | 20% |
| Deployment Strategy | Docker, Kubernetes, CI/CD ready | 20% |
| Innovation | GenAI categorization, XML, JMS alerts | 15% |
| Presentation & Documentation | Demo walkthrough, diagrams, repo structure | 15% |