**1. 5 use cases of your previous project**

## 1. Order Creation and Management

**Actor:** Volkswagen partner (service center, refurbishing unit)

**Flow:**

- Partner places an order for spare accessories, selecting items and specifying quantities.
- The system checks inventory, processes the order, and generates a confirmation.

**Integration Points:** Orders API, Inventory API, Payment API.

## 2. Inventory Status Monitoring

**Actor:** Volkswagen partner

**Flow:**

- Partner queries real-time inventory status (availability and restock dates).

**Integration Points:** Inventory API, MySQL.

## 3. Order Payment Processing

**Actor:** Volkswagen partner

**Flow:**

- Partner selects a payment method, the system processes the transaction, and a confirmation is generated once payment is successful.

**Integration Points:** Payments API, Business API, Orders API.

## 4. Service Unit Registration and Management

**Actor:** Volkswagen partner (service center)

**Flow:**

- Partner registers or updates service unit details (e.g., name, location, contact info).

**Integration Points:** Service Unit API, MySQL.

## 5. Real-time Data Sync via Kafka (Data Ingestion)

**Actor:** System (background process)

**Flow:**

- External data (e.g., inventory updates) is ingested via Kafka to update MySQL/MongoDB in real time.

**Integration Points:** Kafka, MongoDB, MySQL.

**2. Technology used in your project**

I worked with **Java 8** and **Spring Boot** to build RESTful services, using **MySQL** and **MongoDB** for data storage. We used **Apache Kafka** for real-time data sync, **Swagger** for API documentation, and **AWS Parameter Store** for secure config management. The project followed **Agile** principles, and I wrote tests with **JUnit** and **Mockito**. CI/CD pipelines were automated using **Jenkins** and **Docker**.

**3. Tech preparation topic wise**

## 1. Core Java (Java 8)

- **Key Concepts**: Streams API, Lambda Expressions, Functional Interfaces, Multithreading.
- **Prepare**: Writing efficient stream operations, handling concurrency, and using lambda expressions.

## 2. Spring Framework (Spring Boot, Spring MVC, Spring Data JPA)

- **Key Concepts**: Dependency Injection, Spring Boot Auto-Configuration, REST APIs, JPA Repositories.

- **Prepare**: Building REST APIs with Spring Boot, securing APIs with Spring Security, and database interaction with Spring Data JPA.

### 3. Databases (MySQL, MongoDB)

- **Key Concepts**: SQL (MySQL), CRUD operations, Data Modeling, NoSQL (MongoDB).
- **Prepare**: Writing efficient SQL queries, understanding ACID properties in MySQL, and MongoDB basics (CRUD, documents).

### 4. Apache Kafka

- **Key Concepts**: Producers, Consumers, Topics, Partitions, Event-Driven Architecture.
- **Prepare**: Real-time data ingestion, message delivery guarantees, and configuring Kafka consumers and producers.

### 5. Swagger / OpenAPI

- **Key Concepts**: API documentation, Swagger UI, OpenAPI Specification.
- **Prepare**: Documenting REST APIs with Swagger and integrating it into Spring Boot.

### 6. AWS (AWS Parameter Store)

- **Key Concepts**: Configuration Management, Secrets Management, IAM Roles.
- **Prepare**: Externalizing configurations and securing sensitive data with AWS Parameter Store.

### 7. DevOps / CI/CD

- **Key Concepts**: Jenkins, GitLab CI, Docker, Kubernetes (optional).
- **Prepare**: Automating build and deployment pipelines, containerization with Docker, and using Jenkins/GitLab for CI/CD.

### 8. Test-Driven Development (TDD)

- **Key Concepts**: JUnit, Mockito, PowerMock.
- **Prepare**: Writing unit tests with JUnit, mocking dependencies using Mockito.

## 9. Agile Methodology

- **Key Concepts**: Scrum, User Stories, Sprint Planning, Jira.
- **Prepare**: Understanding Scrum roles, ceremonies, and using Jira for project management.