**Certification Project:**

Fictional Company GOODs, an e-commerce company, provides online purchasing services and

operates its business using a traditional Java EE-based web application called Customer Order Service. Although the application has been serving the business well, Company GOODs, started struggling with responding to new business requirements. The current customer order service application is not designed to enable changes in business domain and is not open for applying new technologies for accelerating innovation with the current monolithic architecture.

Company GOODs wants to transform the customer order service application to embrace and better handle changes in both business and technical perspectives and has a list of major business requirements:

1. The new system must be evolutionary, meaning it must be flexible for changes.

2. No down time is allowed in moving traffic from the current system to the newly built system.

3. The new application must be able to scale on demand, or automatically, based on the

payload sent to the system, so that it can react to dynamic shopping behavior patterns.

4. The new system must be open for leveraging emerging technologies to embrace innovation.

• Lab Setup and prerequisite

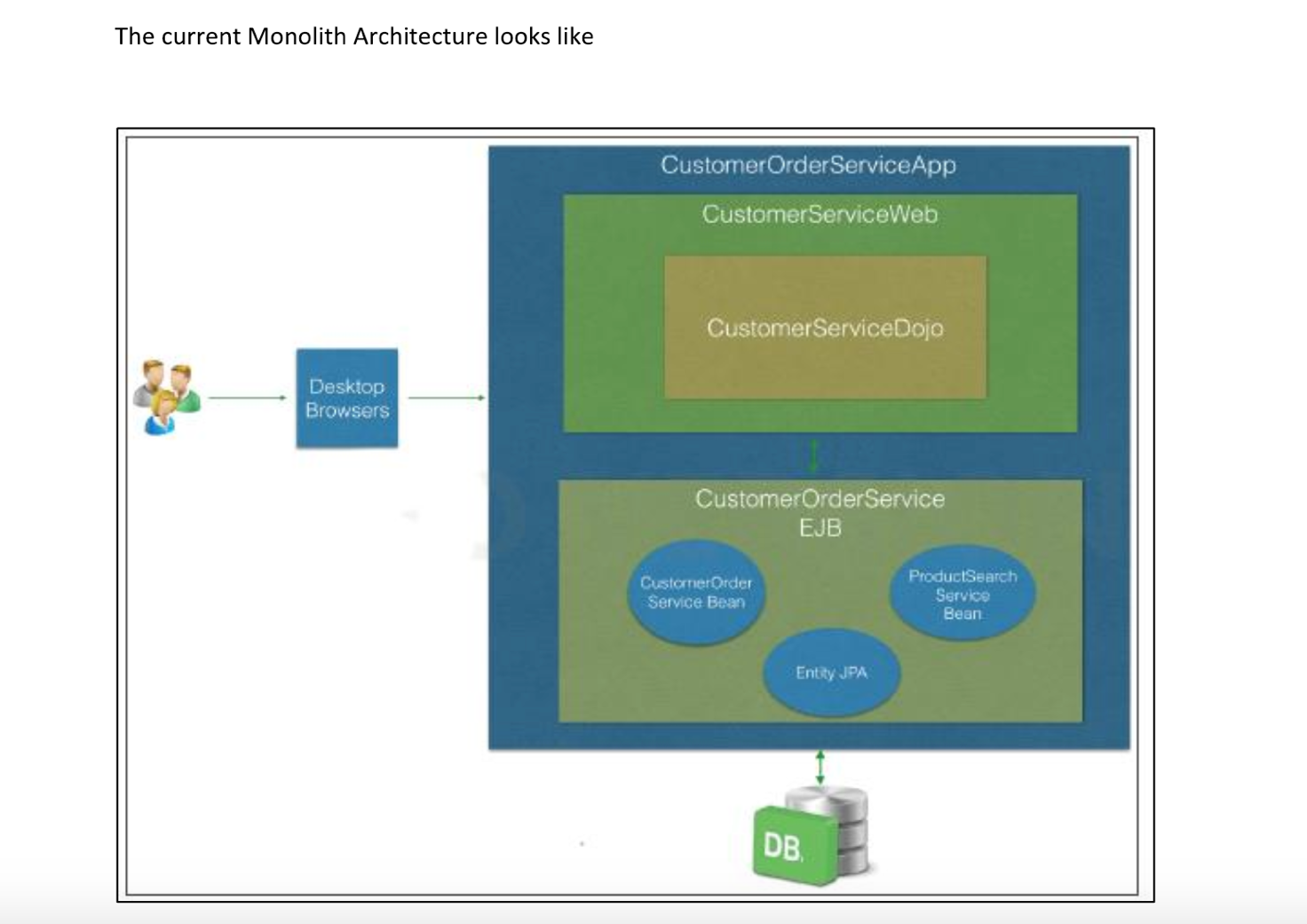
o Case Study

o Identify the Microservices

o Design the Interface

o Develop Microservices

o Test Microservices



**Solution:**

Apply the various rest and Microservice principals we have gone though. Consider you have web front end available to user to place the requests, you do not need to design that. Your focus area should be to identify the business functional and entities and

their interactions.

✓Break the monolith using domain driven design

✓Identify the applications that could exist independently and converted in microservices

✓Identify the right inter service communication pattern

✓Build one of the services using Spring Boot

**Breaking through Domain Driven Design:**

1. Customer/User (Customer will store order details and user information)
2. Order (Customer will order from cart)
3. Product (Customer will search product)
4. Category (Category will be used by product)
5. Cart (Cart can be used for recommendation)
6. Shipment (Order details is sent to Shipment)
7. Payment (Customer will pay the order)
8. Search (Customer will search product)
9. Recommendation (Adding product to cart recommendation of product)
10. Notification (Notify customer about order details and status)

**Identifying microservices:**

1. Customer/User Microservice
2. Order Microservice
3. Product Microservice
4. Catalogue Microservice
5. Shipment Microservice
6. Payment Microservice
7. Search/Recommendation Microservice (Both can be combined and reused)
8. Notification Microservice

**Right communication between services:**

Following microservices uses REST APIs

1. Customer/User Microservice
2. Order Microservice
3. Product Microservice
4. Catalogue Microservice
5. Payment Microservice
6. Search/Recommendation Microservice

Following microservices uses JMS Message Broker/Topic Pub-Sub/Messaging Queue (Active MQ)

1. Shipment Microservice
2. Notification Microservice

I will be using event based SAGA pattern to track the status of the order of the customer.

The design patterns that will be used are:

1. API Gateway (Zuul Proxy)
2. Hystrix Circuit Breaker
3. Aggregator
4. Database pattern- Database per service
5. SAGA – event based pattern
6. Service Discovery
7. Chained
8. RIBBON
9. Eureka Server
10. Authorization Server

Moving from Monolith to Microservices

1. Big Bang

2. Gradual (Netflix is one successful big company to use this approach)

I will be using Gradual approach.

Design:

I have attached the design as a screenshot.

There is a communication with RESTful APIs using HTTP, also queue for shipment and notification as we donot need a synchronous response from them.

Attached code with the submit.

