Document outlining the architecture of your application (before and after)

Table of Contents:

1. Introduction

This document outlines the architecture of our Bakery Application, which started as a monolithic application and was later transformed into a microservices-based architecture. The document will provide an in-depth look at both architectures, highlighting the components, data flow, and deployment strategies.

2. Monolithic Architecture (Before)

Components

User Management: Handles user authentication, profiles, and roles.

Product CatLog: Manages the list of bakery items and their details.

Inventory: Keeps track of stock levels.

Order Management: Manages customer orders.

Payment: Processes payments.

Data Flow:

The frontend communicates directly with a backend server.

The backend server accesses a single, monolithic database for all operations.

Deployment:

The entire application is packaged into a single deployment unit, often a single Docker container.

Deployed on a single server or multiple servers behind a load balancer.

3. Microservices Architecture (After)

Components

API Gateway: Routes incoming requests to appropriate microservices.

User Service: Manages user accounts, profiles, and authentication.

Database: User DB

Product Catalog Service: Manages bakery products, categories, and prices.

Database: Product DB

Inventory Service: Manages stock levels.

Database: Inventory DB

Order Service: Manages customer orders.

Database: Order DB

Payment Service: Processes payments.

Database: Payment DB

Notification Service: Sends email notifications.

Queue: Notification Queue

Data Flow

Users interact with an API Gateway.

The API Gateway routes the request to the appropriate microservice.

Each microservice interacts with its own separate database.

Asynchronous operations, like notifications, are handled through a message queue.

Deployment

Each microservice is containerized into its own Docker container.

Each microservice can be independently deployed and scaled.

Kubernetes is used for orchestration, allowing for automated deployments, scaling, and management of containerized microservices.

4. Conclusion

The transformation from a monolithic to a microservices architecture has allowed for greater flexibility, scalability, and maintainability. Each microservice can be developed, deployed, and scaled independently, providing numerous operational and developmental advantages.