CS331 Project: Retrieval Augmented Generation Assignment 4

Ahlad Pataparla — 2201017
Anushka Srivastava — 2201030
Arya Sahu — 2201033
Khushi Mandal — 2201108

February 28, 2025

I. Microservices Architecture

A. Justification for Microservices Architecture

- **Granularity**: The system is divided into small, independent services (e.g., User Service, Product Service, Order Service, Payment Service, Recommendation Service) that are fine-grained and therefore suited to the microservices architecture. This promotes reusability and modularity.
- Each service has a single responsibility and communicates via APIs such as REST.
- **Decentralized Governance**: Each service can have its own technology stack and deployment cadence.

B. Why Microservices is the Best Choice (and addressing cost constraints)

- **Scalability**: Each service can be scaled independently based on demand. Certain services such as the Recommendation Service can be scaled independently of the main application. This is crucial because recommendation systems often require vastly different resources than typical application functions.
- **Maintainability**: Services are decoupled, making updates easier. Crucially, this reduces the risk of system-wide failures during updates, a major concern with limited resources.
- **Performance**: Lightweight services with optimized APIs ensure faster response times.
- **Flexibility**: Different technologies can be used for different services.
- **Fault Isolation**: If one service fails, it doesn't bring down the whole system. This is vital for a resource-constrained environment as a failure in one component won't cripple the entire application.

II. Application Components

- **User Service**: Handles user registration, login, and profile management.
- **Product Service**: Manages product data and search functionality.
- **Order Service**: Processes orders, cancellations, and order tracking.
- **Payment Service**: Handles payment processing via payment gateways.
- **Recommendation Service**: Provides personalized product recommendations.
- **Admin Service**: Manages products, users, and orders.
- **Front-end**: User interface for browsing, searching, and placing orders.
- **Database**: Stores user, product, and order data.

III. Hosting Application Components

Host Site

- **Local Development Environment**: Due to resource constraints, the initial development and testing will primarily occur in a local development environment (developer laptops). This avoids incurring cloud costs during the development phase.
- **Cloud Deployment**:
  - **Core Application**: Cloud-based virtual machines (VMs) or container orchestration (Kubernetes) on a provider like Amazon AWS.
  - **Recommendation Engine**: Dedicated cloud-based servers (potentially with GPU acceleration) or cloud ML services (AWS SageMaker).
  - **Database**: Cloud-managed database service (AWS RDS).
  - Frontend can also be deployed using GitHub Pages.

Deployment Strategy

- **Local Development and Testing**: Initial development and testing on developer laptops using Docker containers.
- **Staging Environment**: Deployment to a cloud-based staging environment on VMs for integration and performance testing.
- **Production Deployment**: Deployment to a scalable cloud infrastructure with auto-scaling for production.
- **API Communication**: Use of RESTful APIs for communication between the core application and recommendation engine, managed by an API gateway.
- **Infrastructure as Code (IaC)**: Implementation of IaC for automated deployments.

Security

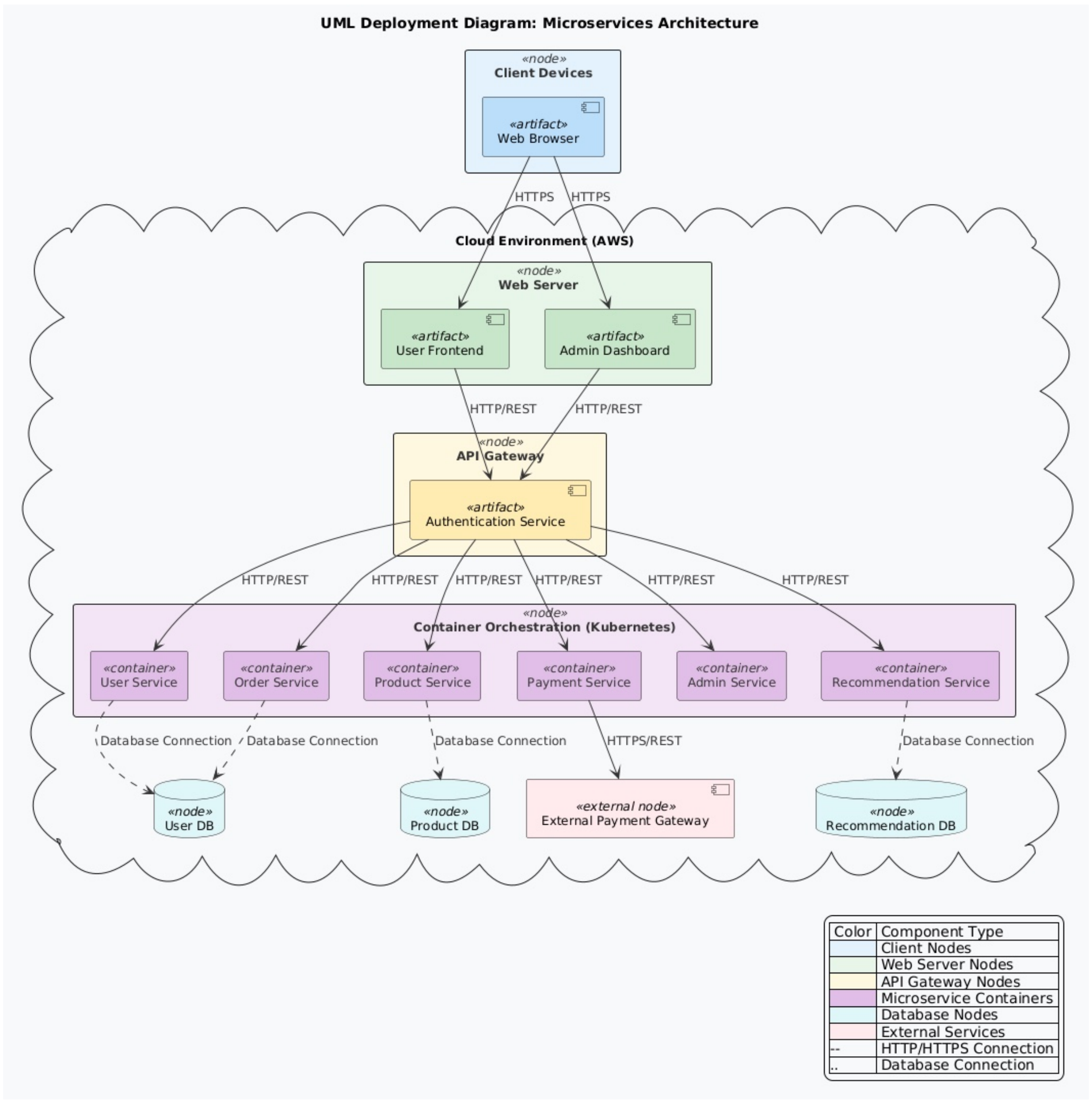
- **API Gateway**: Implementation of an API gateway for authentication and authorization.
- **Cloud Security**: Utilization of cloud provider (AWS) firewalls and network security groups.
- **Data Encryption**: Encryption of data at rest and in transit. JSON Web Tokens used for user sessions and login. Passwords are hashed along with salt and stored. The website will use HTTPS and the data will be encrypted in transit.

IV. User Access and Interaction

How End Users Access Services

- Users access the system via a web browser.
- The front-end interacts with the back-end services via REST APIs.

Pictorial Representation



Color	Component Type
🟦	Client Nodes
🟩	Web Server Nodes
🟨	API Gateway Nodes
🟪	Microservice Containers
🟦	Database Nodes
🟥	External Services
--	HTTP/HTTPS Connection
..	Database Connection