**Online Shopping Platform**

**Team 2**

**Team members:**

|  |  |
| --- | --- |
| Name | Id |
| **أحمد اديب معضماني** | **20201701044** |
| **احمد اسامة عبد الرزاق** | **20201700020** |
| **مصعب ثابت محمد عبد الآخر** | **20201700848** |
| **أميره ياسر إبراهيم حسين** | **20201700150** |
| **الاء مصطفى صادق على** | **20201700137** |
| **شهد اشرف احمد عبداللاه** | **20201700407** |

A diagram of a product service

Description automatically generated

**Project Overview**

The online shopping platform is designed to provide a seamless shopping experience for customers. The platform is composed of multiple microservices handling various functionalities such as user authentication, product catalog management, order processing, and payment processing. To ensure scalability, reliability, and efficient deployment, Docker, Docker Compose, and Kubernetes are utilized

**Architecture**

**Microservices**

**Architecture**

**Microservices**

1. **API Gateway**
   * **Acts as an entry point to the microservices.**
   * **Routes requests to appropriate services.**
   * **Manages CORS configurations.**
2. **User Service**
   * **Handles user authentication and registration.**
   * **Provides endpoints for user login and signup.**
3. **Product Service**
   * **Manages product catalog.**
   * **Provides endpoints for adding, updating, deleting, and retrieving products.**
4. **Order Service**
   * **Manages orders and order details.**
   * **Provides endpoints for creating, retrieving, and deleting orders.**
5. **Payment Service**
   * **Manages payment processing.**
   * **Provides endpoints for making and refunding payments.**

**Communication**

* **Services communicate through RESTful APIs using HTTP.**
* **API Gateway routes the requests to the appropriate services.**

**Architecture Diagram**

**Below is a simplified architecture diagram showing the communication between components:**

**Technology Stack**

* **Backend: Spring Boot**
* **Frontend: (Not provided in the details)**
* **Database: (Not specified)**
* **Containerization: Docker**
* **Orchestration: Kubernetes**

**Dockerization**

**Dockerfile for API Gateway**

FROM maven:3.8.4-openjdk-17 as builder  
  
WORKDIR /app  
  
COPY pom.xml .  
RUN mvn dependency:go-offline  
  
COPY src src  
RUN mvn clean package -DskipTests  
  
FROM openjdk:17  
  
EXPOSE 8080  
  
COPY --from=builder /app/target/api-gateway-0.0.1-SNAPSHOT.jar api-gateway.jar  
  
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "/api-gateway.jar"]

**Docker Compose**

server:  
 port: 8080  
  
spring:  
 *# main:  
 # web-application-type: reactive* cloud:  
 gateway:  
 globalcors:  
 add-to-simple-url-handler-mapping: true  
 cors-configurations:  
 '[/\*\*]':  
 allowedOrigins: "\*"  
 allowedMethods: "\*"  
 allowedHeaders: "\*"  
  
 routes:  
 - id: user  
 uri: http://user:8081  
 predicates:  
 - Path=/api/users/\*\*  
  
 - id: product  
 uri: http://product:8082  
 predicates:  
 - Path=/api/products/\*\*  
  
 - id: order  
 uri: http://order:8083  
 predicates:  
 - Path=/api/orders/\*\*  
  
 - id: payment  
 uri: http://payment:8085  
 predicates:  
 - Path=/api/payments/\*\*

**Environment Setup**

**Development Environment**

1. **Prerequisites**
   * Docker installed.
   * Docker Compose installed.
2. **Setup**
   * Clone the repository.
   * Navigate to the project directory.
   * Run **docker-compose up --build** to start all services.

**Staging and Production Environments**

* Use Kubernetes for orchestration.
* Apply Kubernetes deployment and service files to the cluster using **kubectl apply -f <filename>**.

**Running the Application**

1. **Local Setup**
   * Follow the steps in the development environment setup.
   * Access the API Gateway at **http://localhost:8080**.
2. **Staging and Production**
   * Deploy the services to the Kubernetes cluster.
   * Ensure the services are running and accessible via the API Gateway.

The parameters for each classifier were meticulously tuned using a grid search approach, focusing on achieving the highest accuracy in classifying the motor imagery tasks.