**Software Requirements Specification (SRS) for E-commerce Platform Microservices**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to define the requirements for an e-commerce platform built using a microservices architecture. This document outlines the functionalities, performance, constraints, and interfaces for the system.

**1.2 Scope**

This e-commerce platform will enable users to browse products, manage shopping carts, place orders, process payments, and receive notifications. The system will be composed of multiple independent services, each responsible for a specific domain.

**1.3 Definitions, Acronyms, and Abbreviations**

* **API:** Application Programming Interface
* **JWT:** JSON Web Token
* **SRS:** Software Requirements Specification
* **REST:** Representational State Transfer
* **gRPC:** Google Remote Procedure Call
* **CI/CD:** Continuous Integration/Continuous Deployment

**1.4 References**

* REST API Design Best Practices
* OAuth2.0 Specification
* GDPR Compliance Guidelines

**2. Overall Description**

**2.1 Product Perspective**

This e-commerce platform is an independent product that will be integrated with various third-party services for payment and notifications. It follows a microservices architecture to ensure scalability, maintainability, and ease of deployment.

**2.2 Product Functions**

* User Registration and Authentication
* Product Catalog Management
* Shopping Cart Management
* Order Processing
* Payment Processing
* Notification Management
* Inventory Management
* Shipping Calculation and Management
* Product Reviews and Ratings

**2.3 User Classes and Characteristics**

* **End Users:** Customers who browse products and make purchases.
* **Administrators:** Users who manage the product catalog, orders, and user accounts.
* **Customer Support:** Users who assist customers with their inquiries and issues.

**2.4 Operating Environment**

* **Frontend:** Modern web browsers, mobile devices.
* **Backend:** Hosted on cloud infrastructure (e.g., AWS, Azure, GCP).
* **Database:** Various databases (MongoDB, PostgreSQL, MySQL).
* **API Gateway, Service Discovery, Configuration Management, Logging, and Monitoring:** Managed through cloud-native tools.

**2.5 Design and Implementation Constraints**

* Must comply with GDPR for user data protection.
* Ensure high availability and scalability.
* Use containerization (Docker) and orchestration (Kubernetes) for deployment.

**2.6 Assumptions and Dependencies**

* Reliable internet connection for cloud services.
* Third-party services (Stripe for payment, Twilio for notifications) are available and reliable.

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.1.1 User Service**

* **Registration:** Users must be able to register with an email and password.
* **Login:** Users must be able to log in using their credentials.
* **Profile Management:** Users must be able to update their profile information.
* **Authentication:** Implement JWT for secure user authentication.

**3.1.2 Product Service**

* **Product Catalog:** Ability to add, update, delete, and view products.
* **Product Search:** Search products by name, category, and price.
* **Product Details:** View detailed information about a product.

**3.1.3 Order Service**

* **Shopping Cart:** Add, update, and remove items from the shopping cart.
* **Order Creation:** Create orders from the shopping cart.
* **Order History:** View past orders.

**3.1.4 Payment Service**

* **Payment Processing:** Integrate with Stripe to process payments.
* **Transaction Management:** Record transaction details for each order.

**3.1.5 Notification Service**

* **Order Confirmation:** Send order confirmation via email/SMS.
* **Order Status Updates:** Notify users about order status changes.

**3.1.6 Inventory Service**

* **Stock Management:** Track product stock levels.
* **Stock Update:** Update stock levels based on orders and inventory changes.

**3.1.7 Shipping Service**

* **Shipping Calculation:** Calculate shipping costs based on the destination and product weight.
* **Shipping Management:** Manage shipping details for orders.

**3.1.8 Review Service**

* **Review Submission:** Allow users to submit product reviews.
* **Review Management:** Admins can manage (approve/delete) reviews.
* **Rating System:** Allow users to rate products.

**3.2 Non-Functional Requirements**

**3.2.1 Performance Requirements**

* **Response Time:** The system should respond to user actions within 2 seconds.
* **Scalability:** The system must handle up to 10,000 concurrent users.

**3.2.2 Security Requirements**

* **Data Protection:** Ensure all sensitive data is encrypted in transit and at rest.
* **Authentication:** Use OAuth2.0 and JWT for secure user authentication.
* **Authorization:** Implement role-based access control.

**3.2.3 Usability Requirements**

* **User Interface:** The UI should be intuitive and responsive across devices.
* **Accessibility:** The platform should comply with WCAG 2.1 guidelines.

**3.2.4 Reliability Requirements**

* **Uptime:** Ensure 99.9% uptime for the platform.
* **Backup:** Regular data backups should be performed.

**3.2.5 Maintainability Requirements**

* **Code Quality:** Follow best practices for code quality and documentation.
* **Microservices Architecture:** Ensure services are loosely coupled and independently deployable.

**3.2.6 Portability Requirements**

* **Deployment:** The system should be deployable on any cloud platform.
* **Containerization:** Use Docker for containerization of services.

**3.3 External Interface Requirements**

**3.3.1 User Interfaces**

* **Web Interface:** Responsive web application for end-users.
* **Admin Interface:** Web portal for administrators to manage the platform.

**3.3.2 Hardware Interfaces**

* **Servers:** Cloud-based servers for hosting the backend services.
* **Databases:** Cloud-hosted databases.

**3.3.3 Software Interfaces**

* **Payment Gateway:** Stripe API for payment processing.
* **Notification Services:** Twilio API for sending SMS notifications.
* **Shipping Services:** External shipping API for calculating shipping costs.

**3.3.4 Communication Interfaces**

* **REST/gRPC:** Services will communicate using REST or gRPC protocols.
* **API Gateway:** NGINX or Kong will be used as the API Gateway.

**3.4 System Features**

**3.4.1 User Registration and Authentication**

* **Description:** Allows users to register and log in.
* **Priority:** High
* **Stimulus/Response Sequences:** User registers -> User receives confirmation email -> User logs in -> System generates JWT.
* **Functional Requirements:**
  + Users can register with email and password.
  + Users can log in with email and password.
  + System generates and validates JWT for authenticated sessions.

**3.4.2 Product Catalog Management**

* **Description:** Allows admins to manage the product catalog.
* **Priority:** High
* **Stimulus/Response Sequences:** Admin adds product -> Product is listed in catalog -> User searches for product -> Product details are displayed.
* **Functional Requirements:**
  + Admins can add, update, and delete products.
  + Users can search and view products.

**3.4.3 Order Processing**

* **Description:** Manages user orders from cart to checkout.
* **Priority:** High
* **Stimulus/Response Sequences:** User adds items to cart -> User checks out -> Order is created -> Payment is processed.
* **Functional Requirements:**
  + Users can add, update, and remove items from the cart.
  + Users can create orders from the cart.
  + System processes payments and updates order status.

**3.4.4 Notification Management**

* **Description:** Sends notifications to users.
* **Priority:** Medium
* **Stimulus/Response Sequences:** Order status changes -> Notification is sent to user.
* **Functional Requirements:**
  + System sends order confirmation and status updates via email/SMS.

**4. System Models**

**4.1 Use Case Diagram**

(Create a diagram illustrating key use cases like user registration, product management, order processing, payment processing, and notifications.)

**4.2 Sequence Diagrams**

(Create diagrams showing the interaction between services for key functionalities like user registration, order creation, and payment processing.)

**4.3 Data Flow Diagrams**

(Create diagrams illustrating the data flow between services and databases.)

**4.4 Class Diagrams**

(Create diagrams representing the structure of the major classes and their relationships within each microservice.)

**5. Glossary**

**API:** Application Programming Interface  
**JWT:** JSON Web Token  
**OAuth2.0:** An authorization framework  
**REST:** Representational State Transfer  
**gRPC:** Google Remote Procedure Call  
**CI/CD:** Continuous Integration/Continuous Deployment  
**WCAG:** Web Content Accessibility Guidelines

This SRS document provides a detailed blueprint for building an e-commerce platform using a microservices architecture. Each section outlines specific requirements and design considerations to guide the development process.