**Product Requirements Document (PRD)**

**---**

**DRONE - FAST FOOD DELIVERY**

**1. Problem Alignment**

* Người dùng gặp khó khăn khi đặt món ăn qua nhiều nền tảng khác nhau (web, mobile), trải nghiệm không đồng nhất.
* Khách hàng thiếu công cụ theo dõi đơn hàng minh bạch, thời gian thực.
* Các hệ thống giao đồ ăn hiện nay có quá nhiều bước, gây mất thời gian và dễ nhầm lẫn.
* Quản lý thanh toán và dịch vụ chưa tích hợp đồng bộ, khó mở rộng khi có nhiều đơn hàng.

**1.1. High Level Approach**

* Xây dựng ứng dụng web + mobile sử dụng React.js + React Native để người dùng có thể đặt món thuận tiện.
* Backend theo kiến trúc 3 lớp và chia thành 5 microservices (User, Product, Order, Payment,Delivery(Được bổ sung)).
* Tích hợp CI/CD pipeline để đảm bảo cập nhật nhanh chóng, ổn định.
* Tích hợp giám sát hệ thống real-time để theo dõi hiệu năng từng service.

**1.2. Narrative**

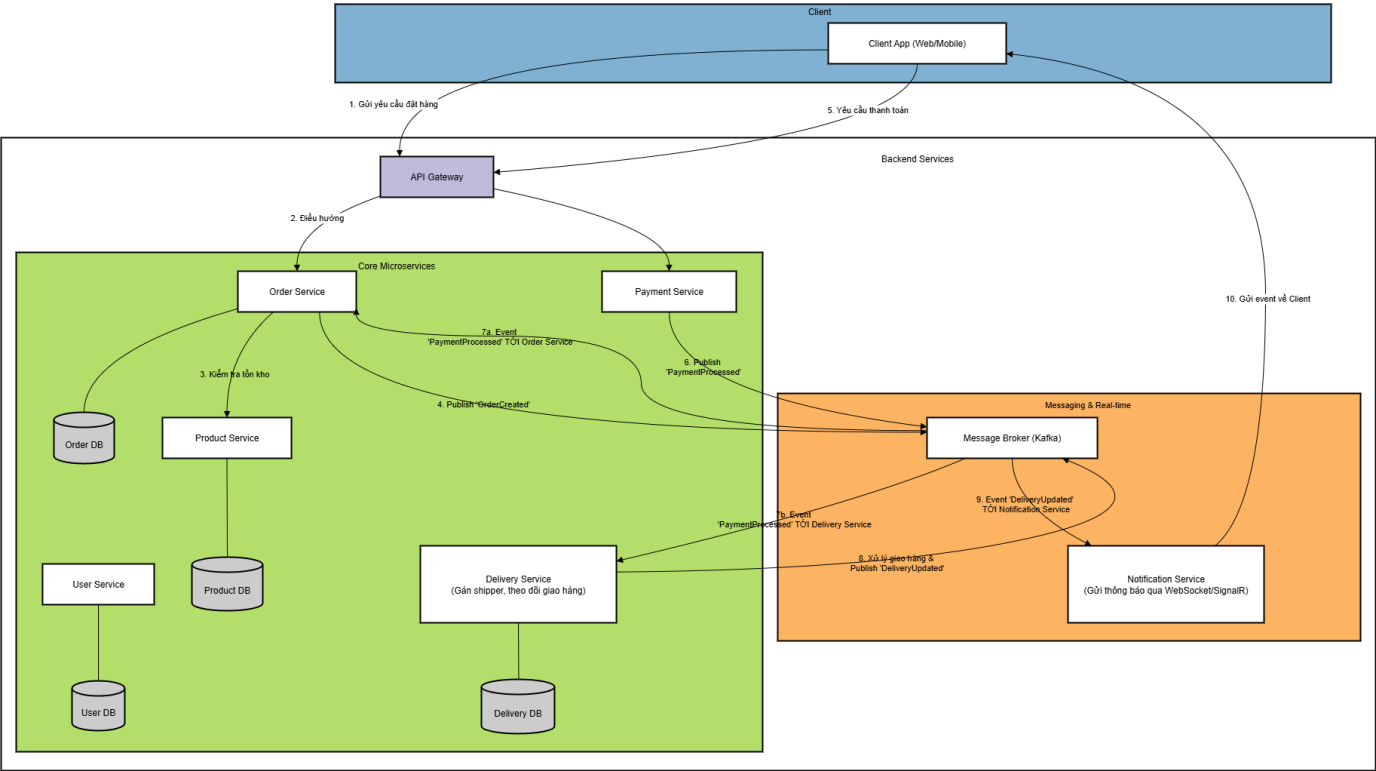
1. Người dùng mở ứng dụng (web/mobile), đăng nhập hoặc tạo tài khoản.
2. Họ duyệt menu, chọn món và thêm vào giỏ hàng.
3. Người dùng xác nhận đơn hàng, chọn phương thức thanh toán (Momo, VNPay, thẻ tín dụng).
4. Hệ thống xử lý quaPayment Service, xác nhận thành công.
5. Order Service cập nhật trạng thái đơn hàng → người dùng theo dõi tiến trình giao hàng theo thời gian thực.
6. Người dùng nhận được thông báo khi đơn hàng hoàn tất.

**1.3. Goals (Mục tiêu)**

* Cung cấp trải nghiệm đặt hàng nhanh, tiện lợi, nhất quán trên web và mobile.
* Tích hợp giỏ hàng, thanh toán, và theo dõi đơn hàng trong một hệ thống duy nhất.
* Dễ dàng mở rộng để hỗ trợ nhiều đối tác (nhà hàng, dịch vụ vận chuyển).
* Đảm bảo hệ thống luôn ổn định, có giám sát real-time và khả năng phục hồi khi gặp sự cố.

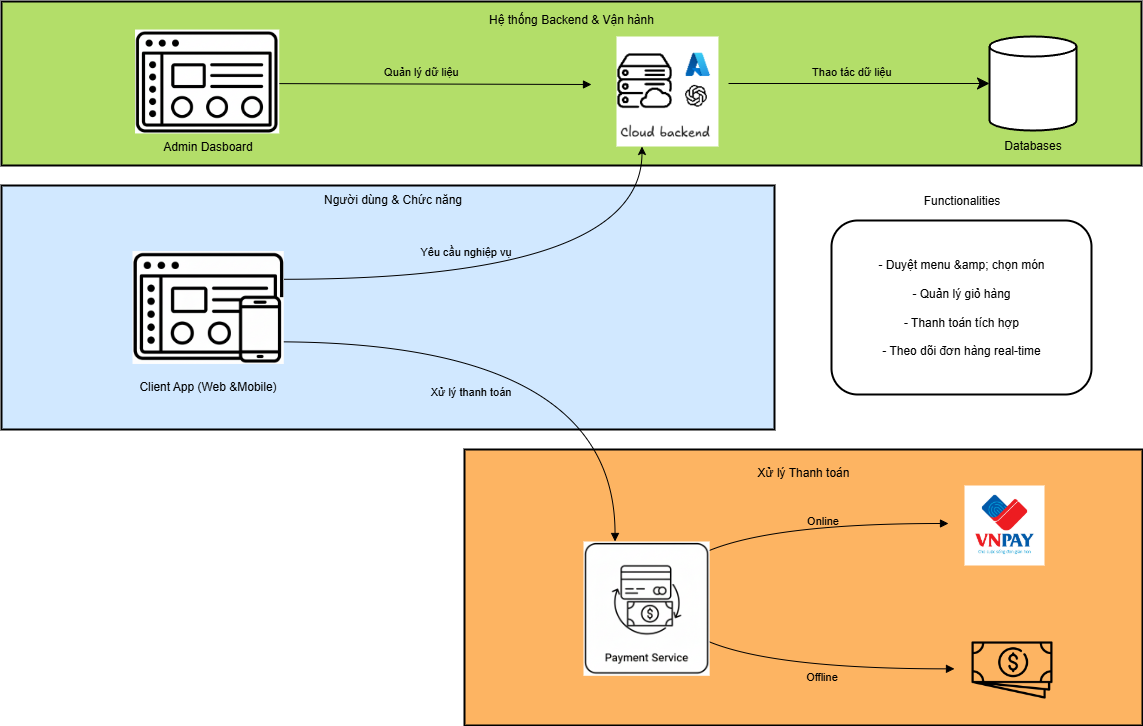
1. **Detailed System Architecture**

Hình 1. Sơ đồ kiến trúc Microservices hướng sự kiện



1. **Solution Alignment**

Hình 2. Sơ đồ tổng quan hệ sinh thái sản phẩm

****

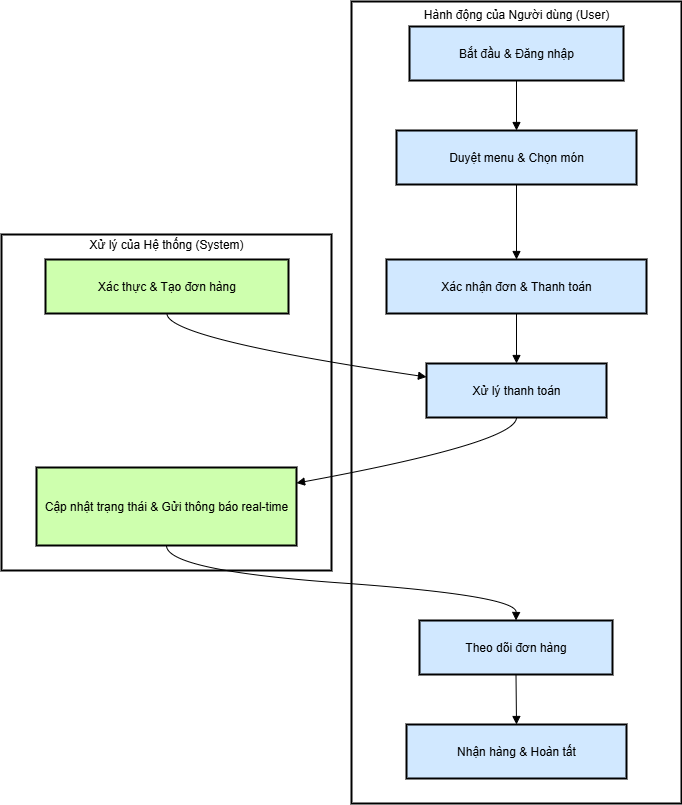
* 1. **Key Features**

1. Frontend (FE):
   * React.js (web), React Native (mobile).
   * Xử lý đặt món, thanh toán, theo dõi đơn hàng.
2. Backend (BE):
   * Kiến trúc 3 lớp: Presentation – Business – Data.
   * Kết nối với DB (MySQL/PostgreSQL).
3. 5 Microservices(Bổ sung Delivery Service):
   * User Service: quản lý tài khoản, xác thực JWT/OAuth2.
   * Product Service: quản lý món ăn, danh mục, tồn kho.
   * Order Service: xử lý giỏ hàng, trạng thái đơn.
   * Payment Service: tích hợp cổng thanh toán (Momo, VNPay, PayPal).
   * Delivery Service: một thành phần quan trọng để thực hiện mục tiêu "theo dõi tiến trình giao hàng theo thời gian thực".
4. CI/CD:
   * Pipeline tự động build, test, deploy cho cả hệ thống và từng service.
   * Dùng Docker + Kubernetes để triển khai.
5. Monitoring:
   * Giám sát real-time bằng Prometheus + Grafana.
   * Dashboard theo dõi cho từng service (User, Product, Order, Payment).

**3.2. Key Flow**

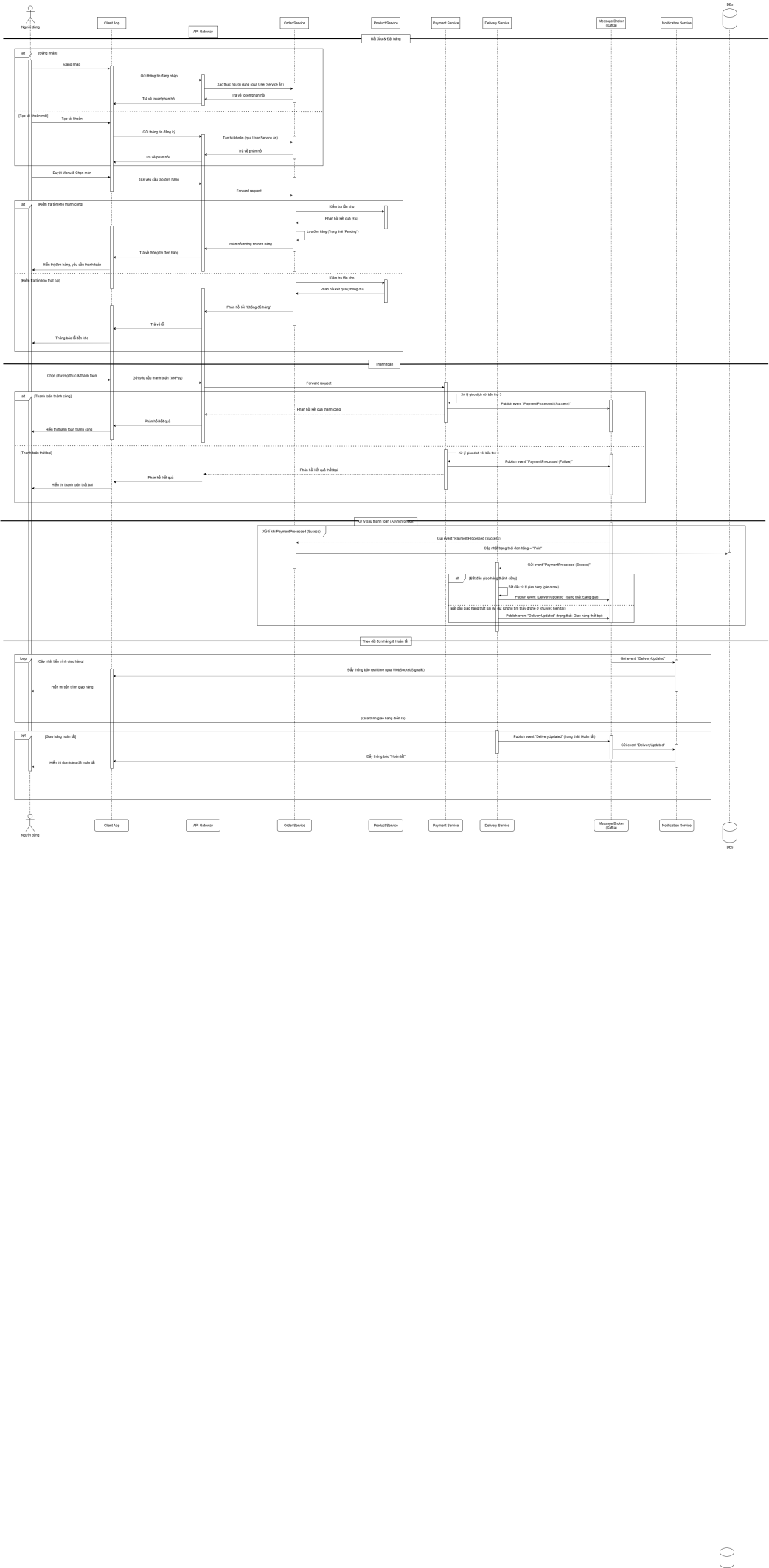
* **Operational Flow**:

Hình 3. Sơ đồ Luồng Hoạt động Nghiệp vụ (Operational Flow Diagram)



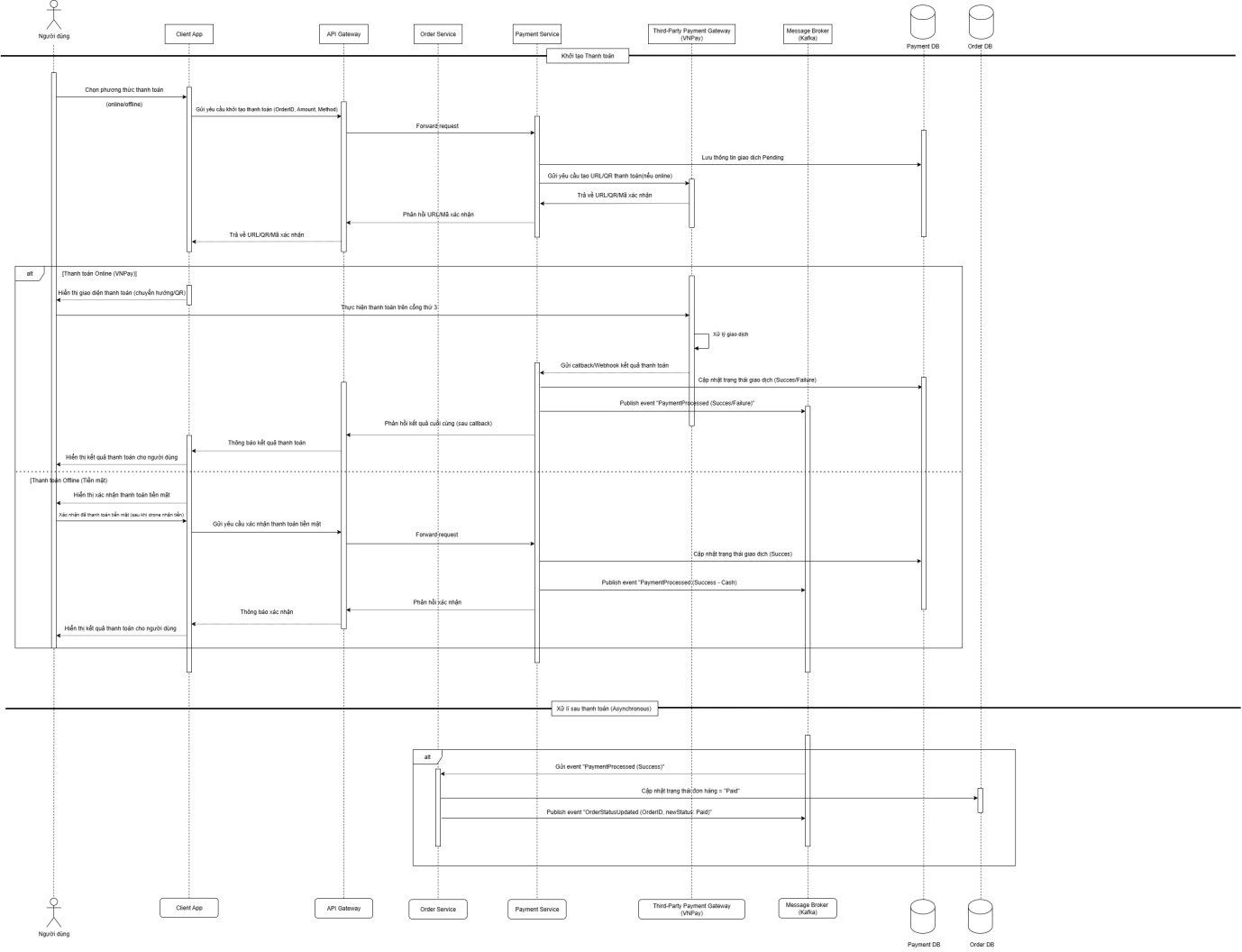
* **Detailed Interaction Flow:**

Hình 4. Luồng Tương tác Chi tiết



* **Payment Flow::**Người dùng xác nhận đơn → Payment Service xử lý → cập nhật Order Service.

Hình 5. Luồng Thanh toán



1. **Yêu cầu nghiệp vụ (Business Requirements)**

**4.1. Mục tiêu nghiệp vụ (Business Goals)**

* Cung cấp một trải nghiệm đặt hàng nhất quán, nhanh chóng và tiện lợi trên cả hai nền tảng Web và Mobile.
* Tích hợp liền mạch các chức năng từ duyệt menu, quản lý giỏ hàng, thanh toán đến theo dõi đơn hàng trong một hệ thống duy nhất để tăng sự hài lòng của khách hàng.
* Xây dựng hệ thống có khả năng mở rộng linh hoạt để dễ dàng tích hợp thêm các đối tác nhà hàng và dịch vụ vận chuyển trong tương lai.
* Đảm bảo hệ thống hoạt động ổn định, có khả năng phục hồi cao và được giám sát theo thời gian thực để giảm thiểu gián đoạn dịch vụ.

**4.2. Thực trạng & Vấn đề (Current State & Problems)**

* Người dùng hiện tại gặp khó khăn do trải nghiệm không đồng nhất khi đặt món ăn qua các nền tảng khác nhau.
* Khách hàng thiếu một công cụ theo dõi đơn hàng minh bạch và cập nhật theo thời gian thực, gây ra sự không chắc chắn.
* Quy trình đặt hàng hiện tại có nhiều bước phức tạp, dễ gây nhầm lẫn và tốn thời gian.
* Việc quản lý thanh toán và các dịch vụ khác chưa được tích hợp đồng bộ, gây khó khăn trong việc mở rộng quy mô.

**4.3. Nghiệp vụ chính (Core Business Processes)**

* Đăng ký & Đăng nhập: Người dùng có thể tạo tài khoản hoặc đăng nhập một cách dễ dàng.
* Đặt hàng & Quản lý giỏ hàng: Người dùng duyệt menu, chọn món và quản lý giỏ hàng của mình.
* Thanh toán: Người dùng xác nhận đơn hàng và thực hiện thanh toán qua các phương thức trực tuyến tích hợp (VNPay).
* Theo dõi đơn hàng: Người dùng theo dõi trạng thái và tiến trình giao hàng của đơn hàng theo thời gian thực sau khi thanh toán thành công.
* Thông báo & Hoàn tất: Người dùng nhận được thông báo tự động khi trạng thái đơn hàng thay đổi và khi đơn hàng được giao thành công.

**5. Yêu cầu chức năng (Functional Requirements)**

**5.1. User Service**

* Cho phép người dùng tạo tài khoản mới và đăng nhập bằng email/mật khẩu.
* Hỗ trợ đăng nhập thông qua các nhà cung cấp OAuth2 như Google /Facebook.
* Quản lý thông tin hồ sơ và địa chỉ giao hàng của người dùng.
* Tạo và xác thực token JWT cho các phiên làm việc an toàn.

**5.2. Product Service**

* Cung cấp API để lấy danh sách món ăn và thông tin chi tiết của từng món.
* Cho phép quản trị viên thêm, xóa, sửa thông tin món ăn (giá, mô tả, hình ảnh).
* Quản lý số lượng tồn kho của từng sản phẩm và cập nhật khi có đơn hàng được tạo.

**5.3. Order Service**

* Xử lý logic giỏ hàng: thêm, xóa, cập nhật số lượng sản phẩm.
* Tạo đơn hàng mới với trạng thái ban đầu là "Pending" sau khi người dùng xác nhận.
* Cung cấp API để người dùng có thể xem trạng thái và lịch sử các đơn hàng đã đặt.
* Cập nhật trạng thái đơn hàng dựa trên các sự kiện từ Payment Service và Delivery Service.

**5.4. Payment Service**

* Tích hợp với các cổng thanh toán bên thứ ba như Momo, VNPay để xử lý giao dịch trực tuyến.
* Xử lý các callback/webhook từ cổng thanh toán để xác nhận giao dịch thành công hay thất bại.
* Publish sự kiện PaymentProcessed lên Message Broker (Kafka) sau khi xử lý xong một giao dịch.

**5.5. Delivery Service**

* Tiếp nhận thông tin đơn hàng đã được thanh toán thành công để bắt đầu quy trình giao hàng.
* Quản lý và cập nhật các trạng thái giao hàng (ví dụ: đang tìm tài xế, đang giao, đã giao thành công).
* Cung cấp dữ liệu để người dùng có thể theo dõi tiến trình giao hàng theo thời gian thực.

**5.6. Notification Service**

* Lắng nghe các sự kiện thay đổi trạng thái đơn hàng từ Message Broker (Kafka).
* Gửi thông báo đẩy (push notification) hoặc cập nhật real-time qua WebSocket/SignalR đến Client App khi có cập nhật mới về đơn hàng.

**6. Yêu cầu phi chức năng (Non-Functional Requirements)**

**Bảo mật (Security):**

* Tất cả các API yêu cầu nghiệp vụ phải được xác thực bằng token JWT/OAuth2 để đảm bảo chỉ người dùng đã đăng nhập mới có thể truy cập.
* Giao tiếp giữa client và server phải được mã hóa bằng HTTPS.

**Hiệu năng (Performance):**

* Thời gian phản hồi của các API chính (lấy danh sách sản phẩm, tạo đơn hàng) phải nhanh chóng (ví dụ: ≤ 500ms) để đảm bảo trải nghiệm người dùng mượt mà.
* Hệ thống phải có khả năng xử lý đồng thời một lượng lớn người dùng và đơn hàng trong giờ cao điểm.

**Tính sẵn sàng & Độ tin cậy (Availability & Reliability):**

* Hệ thống phải được thiết kế để có tính sẵn sàng cao, đảm bảo hoạt động ổn định và có khả năng phục hồi khi một trong các microservice gặp sự cố.
* Sử dụng cơ sở dữ liệu có cơ chế sao lưu (backup) và phục hồi (recovery) định kỳ.

**Khả năng mở rộng (Scalability):**

* Kiến trúc microservices phải cho phép mỗi dịch vụ (User, Product, Order, v.v.) có thể được mở rộng quy mô một cách độc lập tùy theo tải.
* Sử dụng Kubernetes để tự động hóa việc triển khai và mở rộng các container.

**Khả năng giám sát (Monitoring):**

* Hệ thống phải được giám sát hiệu năng theo thời gian thực bằng các công cụ như Prometheus và Grafana.
* Phải có các dashboard giám sát riêng cho từng microservice để dễ dàng theo dõi và phát hiện sự cố.

**Khả năng triển khai (Deployability):** Toàn bộ quy trình từ build, test đến deploy phải được tự động hóa thông qua CI/CD pipeline.

**7. Backend Design**

**7.1. Sub-system (Hệ thống phụ & Dịch vụ bên ngoài)**

* Payment Service: Tích hợp với cổng thanh toán của bên thứ ba là VNPay để xử lý các giao dịch trực tuyến. Hệ thống cũng được thiết kế để có thể mở rộng tích hợp với các cổng khác như Momo.
* Authentication: Hỗ trợ đăng nhập và xác thực người dùng thông qua các nhà cung cấp OAuth2 bên ngoài như Google và Facebook.

**7.2. App (Công nghệ sử dụng)**

* Frontend:

Web App: React.js.

Mobile App: React Native.

* Backend:

Framework: Spring Boot (Gợi ý phù hợp với kiến trúc Microservices dựa trên Java, một lựa chọn phổ biến và mạnh mẽ).

Kiến trúc: Microservices.

* Database: PostgreSQL (Hỗ trợ cả MySQL/PostgreSQL theo PRD).

**7.3. Real-time Delivery Tracking (Theo dõi giao hàng thời gian thực)**

Hệ thống không sử dụng GPS của drone nhưng thực hiện theo dõi đơn hàng theo thời gian thực thông qua cơ chế sự kiện.

* Cung cấp dữ liệu (Data Feed):

Delivery Service chịu trách nhiệm quản lý và cập nhật các trạng thái giao hàng (ví dụ: đang tìm tài xế, đang giao, đã giao thành công).

Mỗi khi có thay đổi, Delivery Service sẽ publish một sự kiện (ví dụ: DeliveryUpdated) lên Message Broker (Kafka).

* Hiển thị dữ liệu (Data Display):

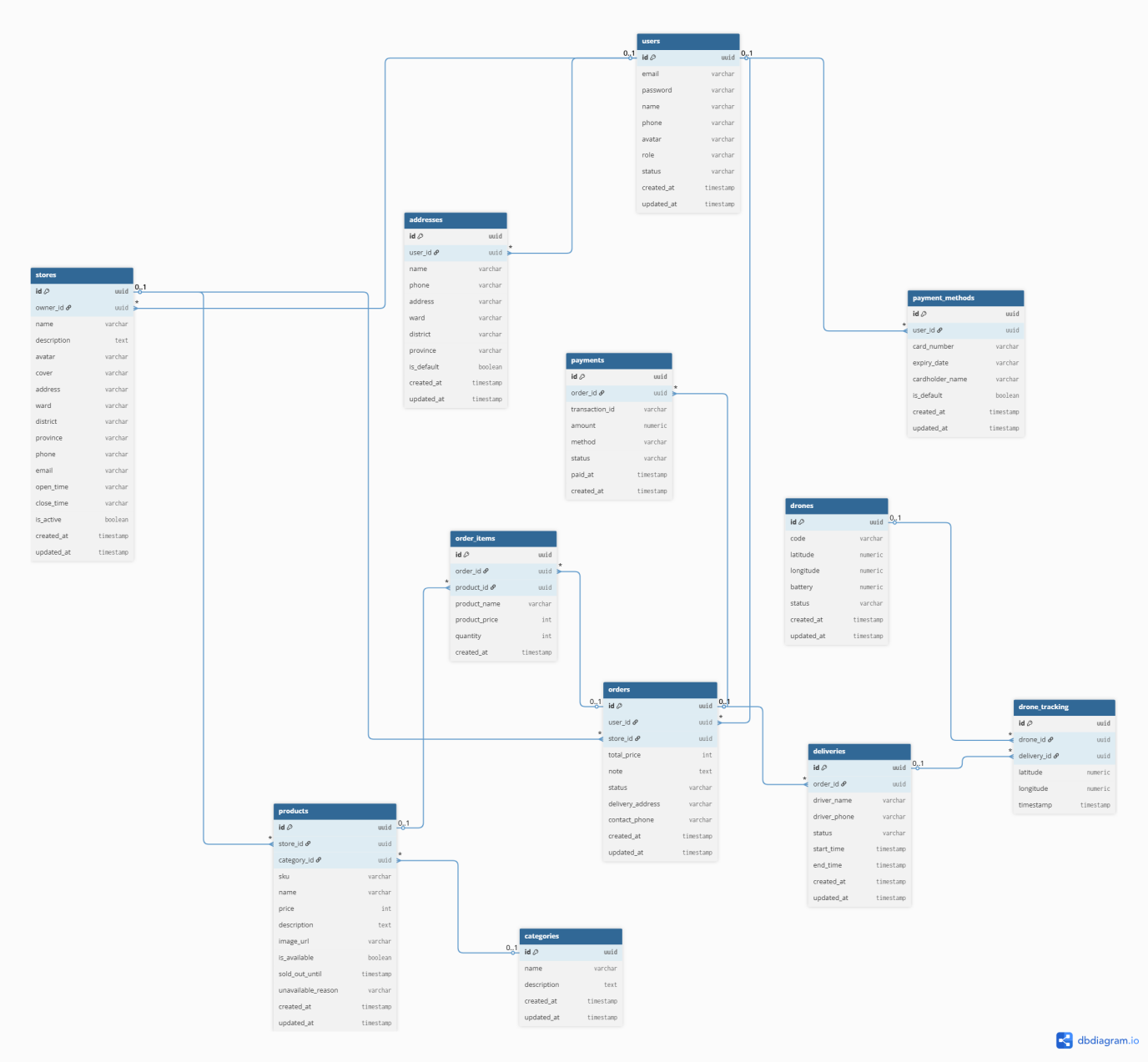
Notification Service sẽ lắng nghe các sự kiện DeliveryUpdated từ Kafka.

Ngay khi nhận được sự kiện, Notification Service sẽ gửi thông báo đẩy (push notification) hoặc cập nhật real-time qua WebSocket/SignalR đến ứng dụng của người dùng (Web/Mobile).

Người dùng từ đó có thể theo dõi trạng thái và tiến trình giao hàng của đơn hàng một cách minh bạch và tức thì.

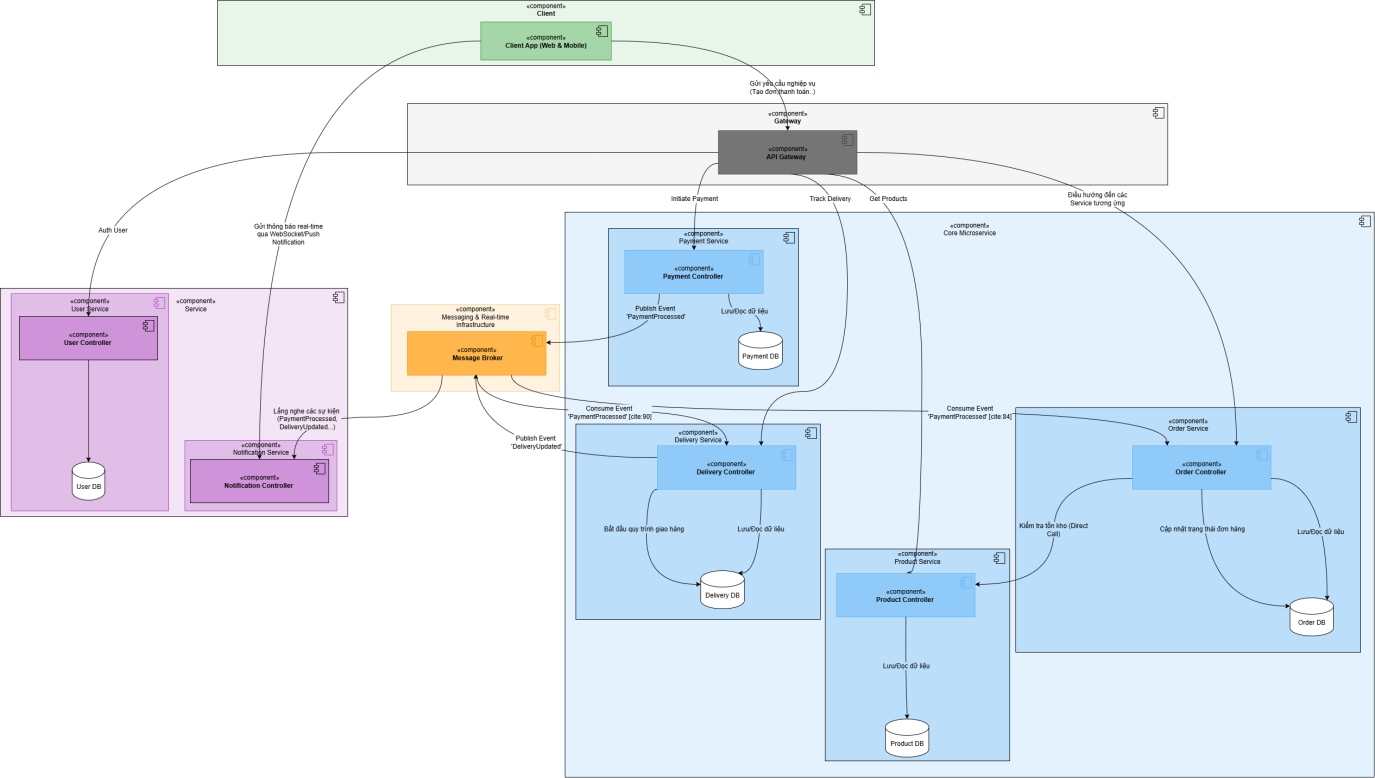
**7.4. Database**

Hình 6. Database

****

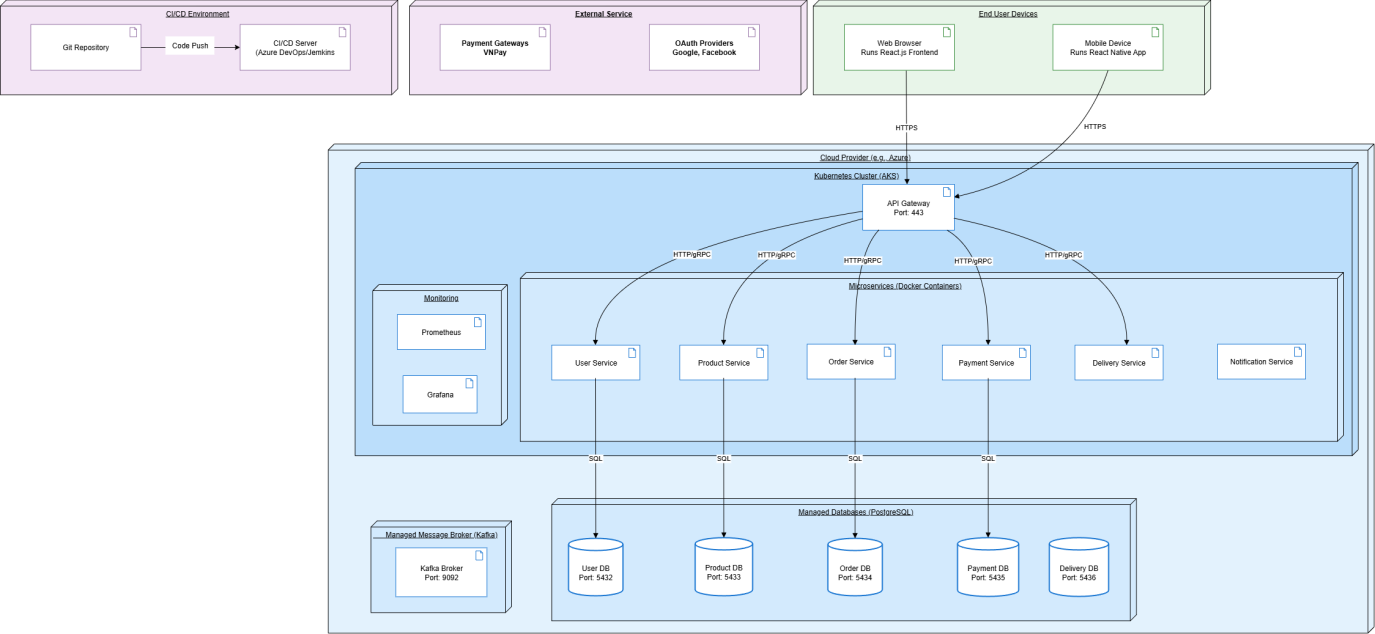
**7.5. Component**

Hình 7. Sơ đồ Component

****

**7.6. Deployment**

Hình 8. Sơ đồ Triển khai

****