# **Product Requirement Document (PRD)**

**Sản phẩm:** FoodFast Delivery   
**Kiến trúc:** Microservices + React UI  
**Ngày:** 2025-09-24

### **1. Problem Alignment (Vấn đề và Mục tiêu)**

**Vấn đề:** Các nền tảng đặt đồ ăn hiện tại có giao diện phức tạp, tốc độ xử lý chậm và trải nghiệm người dùng không mượt mà. Trong khi đó, các nhà hàng nhỏ thiếu một hệ thống dễ dàng quản lý đơn hàng. Thêm vào đó, dịch vụ giao hàng truyền thống gặp phải các vấn đề về tắc nghẽn giao thông và thời gian giao hàng không ổn định.

**Mục tiêu (KPI):**

* Giảm thời gian xử lý và xác nhận đơn hàng xuống dưới 30 giây.
* Tối thiểu 50 nhà hàng đối tác tham gia trong năm đầu.
* **Giảm thời gian giao hàng trung bình xuống dưới 20 phút bằng việc sử dụng drone.** 🚁

### **2. Solution Alignment (Giải pháp)**

### **I. Client Apps (Ứng dụng phía người dùng)**

1. **Customer Web App (React UI):**
   * **Mô tả:** Đây là giao diện người dùng chính mà khách hàng tương tác để đặt đồ ăn. Được xây dựng bằng React (SPA - Single Page Application) để mang lại trải nghiệm mượt mà, nhanh chóng.
   * **Chức năng:** Duyệt menu, tìm kiếm món ăn, thêm vào giỏ hàng, tạo đơn hàng, thực hiện thanh toán, theo dõi trạng thái đơn hàng và vị trí drone.
   * **Tương tác:** Giao tiếp với Backend Microservices thông qua **API Gateway**.
2. **Web app for mobile devices (React UI):**
   * **Mô tả:** Tương tự như Customer Web App nhưng được tối ưu hóa cho thiết bị di động. Về bản chất có thể là Responsive Web Design của cùng một ứng dụng web React.
   * **Chức năng:** Cung cấp trải nghiệm tương tự Customer Web App nhưng trên môi trường di động.

### **II. Backend Microservices (Các dịch vụ Backend)**

Đây là khối cốt lõi của hệ thống, bao gồm nhiều dịch vụ nhỏ, độc lập, giao tiếp với nhau.

1. **API Gateway:**
   * **Mô tả:** Là điểm truy cập duy nhất cho tất cả các yêu cầu từ Client Apps. Nó đóng vai trò như một "người gác cổng", định tuyến các yêu cầu đến đúng microservice và xử lý các tác vụ chung như xác thực, ủy quyền, giới hạn tốc độ yêu cầu (rate limiting).
   * **Chức năng:**
     + **Routing:** Chuyển tiếp các yêu cầu đến microservice phù hợp.
     + **Authentication/Authorization:** Xác thực JWT token của người dùng trước khi chuyển tiếp yêu cầu.
     + **Load Balancing:** Phân phối tải giữa các instance của một service.
     + **Logging:** Ghi lại các yêu cầu đến.
   * **Tương tác:** Nhận yêu cầu từ Client Apps, gửi yêu cầu đến các Microservices nội bộ, gửi log/metrics đến Monitoring & Logging Service.
2. **User Service:**
   * **Mô tả:** Quản lý tất cả thông tin liên quan đến người dùng.
   * **Chức năng:** Đăng ký, đăng nhập, quản lý hồ sơ người dùng (địa chỉ, thông tin liên hệ), xác thực, cấp/thu hồi JWT token.
   * **Data Store:** **Users DB** (MongoDB) - lưu trữ thông tin người dùng.
   * **Tương tác:** Giao tiếp với API Gateway, có thể nhận yêu cầu từ Order Service (để lấy thông tin người dùng cho đơn hàng).
3. **Product Service:**
   * **Mô tả:** Quản lý thông tin về các món ăn, menu, danh mục sản phẩm từ tất cả các nhà hàng đối tác.
   * **Chức năng:** Thêm/sửa/xóa món ăn, quản lý danh mục, tìm kiếm, lọc sản phẩm.
   * **Data Store:** **Products DB** (MongoDB) - lưu trữ thông tin sản phẩm và nhà hàng.
   * **Tương tác:** Giao tiếp với API Gateway.
4. **Order Service:**
   * **Mô tả:** Quản lý toàn bộ vòng đời của một đơn hàng, từ khi người dùng tạo, xác nhận, đến khi hoàn tất giao hàng.
   * **Chức năng:** Tạo đơn hàng, cập nhật trạng thái đơn hàng (Pending, Confirmed, Preparing, Delivered), xem chi tiết đơn hàng, quản lý danh sách đơn hàng.
   * **Data Store:** **Orders DB** (MongoDB) - lưu trữ thông tin chi tiết của từng đơn hàng.
   * **Tương tác:** Giao tiếp với API Gateway, nhận thông tin từ Product Service (về sản phẩm trong đơn hàng), gửi yêu cầu đến Payment Service để xử lý thanh toán, gửi yêu cầu đến Delivery & GPS Service để bắt đầu giao hàng. Cũng nhận thông báo cập nhật trạng thái từ Message Broker (ví dụ: thanh toán thành công, drone đã giao hàng).
5. **Payment Service:**
   * **Mô tả:** Xử lý tất cả các nghiệp vụ liên quan đến thanh toán.
   * **Chức năng:** Xử lý thanh toán mô phỏng (trong phiên bản đầu), chuẩn bị tích hợp với các cổng thanh toán thực tế.
   * **Data Store:** **Payments DB** (MongoDB) - lưu trữ lịch sử giao dịch thanh toán.
   * **Tương tác:** Nhận yêu cầu xử lý thanh toán từ Order Service, giao tiếp với External Payment Gateway (cho tương lai), gửi thông báo kết quả thanh toán về Order Service (có thể qua Message Broker).
6. **Delivery & GPS Service:**
   * **Mô tả:** Đây là service cốt lõi mới, quản lý toàn bộ quy trình giao hàng bằng drone, tận dụng công nghệ GPS.
   * **Chức năng:**
     + **Lập kế hoạch lộ trình:** Nhận tọa độ địa điểm nhận hàng và giao hàng, tính toán tuyến đường bay tối ưu cho drone.
     + **Điều khiển Drone:** Gửi các lệnh điều khiển (ví dụ: cất cánh, bay đến tọa độ, hạ cánh) đến drone.
     + **Theo dõi GPS:** Nhận dữ liệu vị trí GPS theo thời gian thực từ drone.
     + **Tính toán ETA (Estimated Time of Arrival):** Dựa trên vị trí hiện tại và lộ trình, tính toán thời gian dự kiến drone sẽ đến nơi.
     + **Cập nhật trạng thái giao hàng:** Báo cáo trạng thái giao hàng (ví dụ: đang bay, đã đến nơi) cho Order Service.
   * **Data Store:** **GPS Data DB** (MongoDB) - lưu trữ dữ liệu lộ trình, vị trí drone, lịch sử giao hàng.
   * **Tương tác:** Nhận yêu cầu giao hàng từ Order Service, giao tiếp với External Maps & GPS API (để tính toán lộ trình), gửi thông tin vị trí/ETA cho Client Apps (qua API Gateway), gửi thông báo trạng thái giao hàng đến Message Broker (để Order Service nhận).
7. **Monitoring & Logging Service:**
   * **Mô tả:** Một dịch vụ tập trung để thu thập, lưu trữ và phân tích log (nhật ký) và metrics (số liệu) từ tất cả các microservice khác.
   * **Chức năng:** Giúp phát hiện lỗi, giám sát hiệu suất hệ thống, phân tích hành vi người dùng và khắc phục sự cố.
   * **Tương tác:** Nhận log/metrics từ API Gateway và tất cả các Microservices khác.
8. **Message Broker (RabbitMQ):**
   * **Mô tả:** Một hệ thống hàng đợi tin nhắn (message queue) được sử dụng để giao tiếp bất đồng bộ giữa các microservice.
   * **Chức năng:**
     + **Đảm bảo tính nhất quán:** Khi một service cần thông báo cho các service khác về một sự kiện (ví dụ: thanh toán thành công, trạng thái đơn hàng thay đổi), nó sẽ gửi một message vào broker. Các service quan tâm sẽ lắng nghe và xử lý message đó.
     + **Khử khớp nối (Decoupling):** Các service không cần biết trực tiếp về nhau, chỉ cần gửi/nhận message qua broker.
     + **Khả năng mở rộng:** Dễ dàng thêm các service mới lắng nghe các sự kiện.
   * **Tương tác:** Order Service, Payment Service, Delivery & GPS Service có thể gửi và nhận message qua đây để cập nhật trạng thái và phối hợp nghiệp vụ.

### **III. Data Stores (Cơ sở dữ liệu)**

Mỗi microservice có cơ sở dữ liệu riêng (trong trường hợp này là MongoDB) để đảm bảo tính độc lập.

1. **Users DB:** Dành cho User Service.
2. **Products DB:** Dành cho Product Service.
3. **Orders DB:** Dành cho Order Service.
4. **Payments DB:** Dành cho Payment Service.
5. **GPS Data DB:** Dành cho Delivery & GPS Service.

### **IV. External Systems (Các hệ thống bên ngoài)**

Các dịch vụ không phải là một phần của hệ thống FoodFast Delivery nhưng được tích hợp để cung cấp chức năng.

1. **Payment Gateway (Ex: MoMo, ZaloPay):**
   * **Mô tả:** Cổng thanh toán bên thứ ba để xử lý các giao dịch tài chính thực tế (cho tương lai).
   * **Tương tác:** Payment Service sẽ giao tiếp với cổng này.
2. **Maps & GPS API (Ex: Google Maps, OpenStreetMap):**
   * **Mô tả:** Các dịch vụ bản đồ và GPS bên thứ ba cung cấp dữ liệu bản đồ, tính toán lộ trình, thông tin giao thông.
   * **Tương tác:** Delivery & GPS Service sẽ gọi API của các dịch vụ này để có được dữ liệu cần thiết cho việc điều khiển và theo dõi drone.

### **V. Other Components**

1. **Cloud Backend:**
   * **Mô tả:** Nền tảng điện toán đám mây (ví dụ: Azure, AWS, GCP) nơi tất cả các microservice và cơ sở dữ liệu được triển khai.
   * **Chức năng:** Cung cấp tài nguyên tính toán, lưu trữ, mạng và các dịch vụ quản lý khác.

FoodFast Delivery là một ứng dụng web cho phép người dùng đặt đồ ăn trực tuyến.

* **Frontend:** Ứng dụng web đơn trang **React (SPA)** cung cấp trải nghiệm mượt mà, trực quan.
* **Backend:** Kiến trúc **microservices** linh hoạt và dễ mở rộng, quản lý các nghiệp vụ độc lập như người dùng, sản phẩm, đơn hàng, thanh toán và **giao hàng bằng drone**.
* **Deployment:** Triển khai bằng **Docker Compose** để dễ dàng cài đặt và quản lý, có thể nâng cấp lên **Kubernetes** khi hệ thống mở rộng.

### **3. High-Level Goals (Mục tiêu tổng quan)**

* Xây dựng giao diện web thân thiện, cho phép người dùng duyệt món ăn, đặt hàng và thanh toán nhanh chóng.
* Phát triển các microservices độc lập cho từng nghiệp vụ:
  + **User Service:** Quản lý thông tin người dùng, đăng ký, đăng nhập và xác thực.
  + **Product Service:** Quản lý menu, danh mục sản phẩm của các nhà hàng.
  + **Order Service:** Quản lý toàn bộ vòng đời của một đơn hàng, từ khi tạo đến khi hoàn tất.
  + **Payment Service:** Xử lý các giao dịch thanh toán.
  + **Delivery & GPS Service:** Quản lý toàn bộ quy trình giao hàng bằng drone, bao gồm **định vị GPS**, **tìm đường tối ưu** và theo dõi trạng thái.
* Đảm bảo hệ thống có khả năng mở rộng dễ dàng bằng cách thêm các services mới (ví dụ: Review Service).
* Sử dụng **MongoDB** cho cơ sở dữ liệu để tận dụng tính linh hoạt của NoSQL, phù hợp với từng service.

### **4. Scope (Phạm vi dự án)**

**In-Scope (Trong phạm vi):**

* **Người dùng:** Đăng ký, đăng nhập, chỉnh sửa thông tin cá nhân.
* **Sản phẩm:** Xem danh sách, tìm kiếm, xem chi tiết sản phẩm.
* **Giỏ hàng & Đơn hàng:** Thêm/bớt sản phẩm vào giỏ hàng, tạo và xem chi tiết đơn hàng.
* **Thanh toán:** Xử lý thanh toán mô phỏng.
* **Giao hàng bằng Drone:** Lập trình điều khiển drone tự động dựa trên tọa độ GPS, tìm đường tối ưu và tự động giao hàng.
* **Triển khai:** Sử dụng Docker Compose.
* **Backend:** API Gateway để định tuyến request. **Out-of-Scope (Ngoài phạm vi):**
* Ứng dụng di động Native (sẽ phát triển sau).
* Tích hợp cổng thanh toán thực tế (Stripe, MoMo).
* Quản lý tài khoản cho nhà hàng đối tác (chỉ có admin quản lý sản phẩm).
* Chức năng đánh giá và bình luận sản phẩm.

**Giả định & Ràng buộc:**

* Giả định người dùng và drone có kết nối Internet/GPS ổn định.
* Phiên bản đầu tiên chỉ hỗ trợ một khu vực địa lý nhất định (ví dụ: một quận/thành phố).
* Việc sử dụng drone tuân thủ các quy định về không lưu tầm thấp của khu vực.

### **5. Features (Tính năng chi tiết)**

* **API Gateway:**
  + POST /api/auth/login: Xác thực người dùng và trả về JWT token.
  + POST /api/auth/register: Đăng ký tài khoản mới.
* **User Service:**
  + GET /users/:id: Lấy thông tin người dùng.
  + PATCH /users/:id: Cập nhật thông tin người dùng.
  + GET /users/:id/orders: Lấy danh sách đơn hàng đã đặt của người dùng.
* **Product Service:**
  + GET /products: Lấy danh sách sản phẩm, hỗ trợ tìm kiếm và lọc.
  + GET /products/:id: Lấy chi tiết sản phẩm.
  + POST /products: Thêm sản phẩm mới (chỉ admin).
* **Order Service:**
  + POST /orders: Tạo đơn hàng mới từ giỏ hàng.
  + GET /orders/:id: Xem chi tiết đơn hàng.
  + PATCH /orders/:id/status: Cập nhật trạng thái đơn hàng (PENDING, CONFIRMED, DELIVERED).
* **Payment Service:**
  + POST /payments/process: Xử lý thanh toán giả lập, trả về kết quả thành công/thất bại.
* **Delivery & GPS Service:**
  + POST /delivery/start-trip: Bắt đầu một chuyến giao hàng mới.
  + GET /delivery/:orderId/location: Cung cấp vị trí GPS hiện tại của drone.
  + GET /delivery/:orderId/eta: Cung cấp thời gian dự kiến (ETA) của drone.

### **6. User Flow (Kịch bản người dùng chi tiết)**

1. Người dùng mở FoodFast web app.
2. Đăng nhập (nếu đã có tài khoản) hoặc đăng ký tài khoản mới.
3. Duyệt sản phẩm, thêm vào giỏ hàng và đặt đơn.
4. Khi đơn hàng được xác nhận, **Order Service** sẽ gửi request đến **Payment Service** để xử lý thanh toán giả lập.
5. Sau khi thanh toán thành công, **Order Service** sẽ gửi yêu cầu đến **Delivery & GPS Service** để khởi động drone và bắt đầu hành trình giao hàng.
6. Drone nhận tọa độ GPS của người dùng và nhà hàng. **Delivery & GPS Service** sử dụng thuật toán tìm đường tối ưu để xác định tuyến đường bay.
7. Trong quá trình giao hàng, người dùng có thể theo dõi vị trí của drone và thời gian nhận hàng dự kiến thông qua **Delivery & GPS Service** (GET /delivery/:orderId/location và GET /delivery/:orderId/eta).
8. Sau khi drone đến nơi, đơn hàng được hoàn tất.

### **7. Architecture (Kiến trúc)**

**Frontend:** React + Vite (SPA) với Redux Toolkit để quản lý trạng thái. **Backend:**

* **API Gateway:** Định tuyến và bảo mật request.
* **Microservices:**
  + User Service (Express.js, MongoDB)
  + Product Service (Express.js, MongoDB)
  + Order Service (Express.js, MongoDB)
  + Payment Service (Express.js, MongoDB)
  + **Delivery & GPS Service (Express.js, MongoDB)**
* **Giao tiếp:** Các services giao tiếp với nhau bằng REST API. Có thể sử dụng **message queue (RabbitMQ)** cho các luồng bất đồng bộ như cập nhật trạng thái đơn hàng, đảm bảo tính nhất quán.
* **Triển khai:**
  + Development/Staging: Docker Compose.
  + Production: Nâng cấp lên Kubernetes.

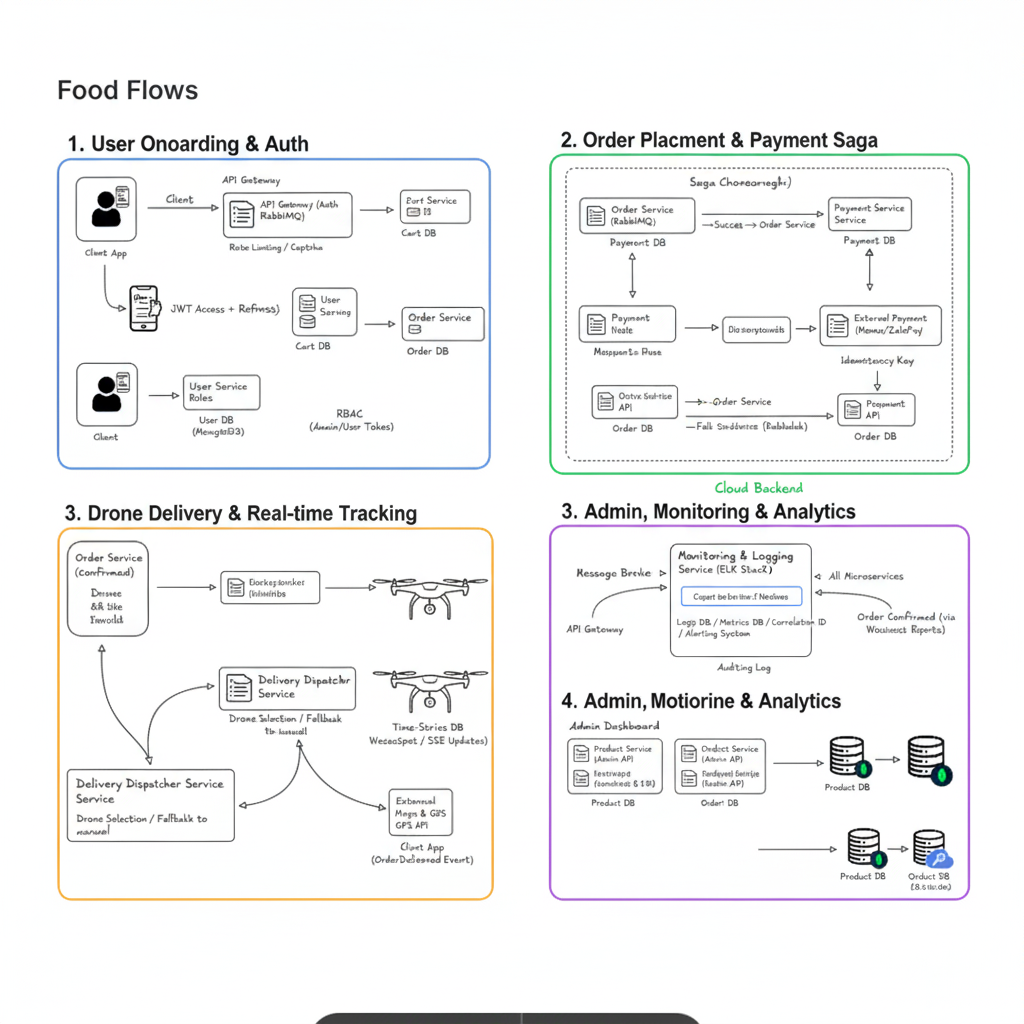
### **8. Non-Functional Requirements (Yêu cầu phi chức năng)**

* **Hiệu năng:** Thời gian phản hồi API trung bình dưới 500ms. Riêng các API liên quan đến đơn hàng và giao hàng phải dưới 200ms.
* **Bảo mật:** Mã hoá mật khẩu người dùng (bcrypt), xác thực JWT token.
* **Khả năng mở rộng:** Kiến trúc microservices cho phép mở rộng độc lập từng service.
* **Khả dụng:** Uptime hệ thống tối thiểu 99%.
* **Khả năng quan sát:** Thiết lập hệ thống logging tập trung (ELK Stack) và giám sát hiệu năng (Prometheus/Grafana).

### **9. Future Considerations (Mở rộng tương lai)**

* Tích hợp cổng thanh toán thực tế (MoMo, ZaloPay).
* Phát triển hệ thống quản lý tài khoản nhà hàng.
* Xây dựng hệ thống gợi ý sản phẩm dựa trên AI.
* Triển khai hệ thống đánh giá và xếp hạng.
* Phát triển ứng dụng di động Native cho người dùng và hệ thống quản lý drone

### **10. Key Flow**



I. Đăng ký & Xác thực người dùng (User Onboarding & Authentication)

Luồng này đã được nâng cấp để đảm bảo tính an toàn và quản lý phiên làm việc hiệu quả hơn.

1. **Đăng ký người dùng mới:** Khi người dùng gửi thông tin đăng ký, ứng dụng web (Client App) sẽ gửi yêu cầu đến **API Gateway**. Tại đây, một cơ chế **rate limiting** (giới hạn tốc độ) và **Captcha** được áp dụng để chống lại các hành vi spam. Yêu cầu sau đó được chuyển đến **User Service**. Service này sẽ xác thực dữ liệu, mã hóa mật khẩu bằng thuật toán **bcrypt** rồi lưu thông tin vào cơ sở dữ liệu **User DB** của nó.
2. **Đăng nhập & Cấp Token:** Người dùng đăng nhập bằng tên và mật khẩu. **User Service** xác minh thông tin và, nếu thành công, sẽ tạo và cấp hai loại token: một **Access Token** có thời gian sống ngắn (ví dụ: 15-30 phút) và một **Refresh Token** có thời gian sống dài hơn (ví dụ: 7 ngày). Access Token được sử dụng để truy cập các API cần xác thực, trong khi Refresh Token được lưu trữ an toàn trên máy người dùng để lấy Access Token mới khi cái cũ hết hạn mà không cần phải đăng nhập lại.
3. **Kiểm soát truy cập dựa trên vai trò (RBAC):** Access Token còn chứa thông tin về vai trò của người dùng (như user hoặc admin). **API Gateway** sẽ kiểm tra vai trò này để đảm bảo rằng các API nhạy cảm (ví dụ: API thêm sản phẩm) chỉ có thể được truy cập bởi người dùng có quyền admin.

II. Đặt hàng & Thanh toán (Order Placement & Payment)

Luồng này được tối ưu hóa để đảm bảo tính nhất quán của dữ liệu giữa các dịch vụ.

1. **Quản lý giỏ hàng:** Người dùng thêm sản phẩm vào giỏ hàng. Ứng dụng web sẽ gửi yêu cầu đến một **Cart Service** mới, dịch vụ này sẽ quản lý và lưu trữ dữ liệu giỏ hàng của người dùng trong cơ sở dữ liệu riêng, đảm bảo giỏ hàng không bị mất khi người dùng tải lại trang.
2. **Tạo đơn hàng & Mô hình Saga:** Khi người dùng xác nhận đơn hàng, **Order Service** sẽ khởi tạo một giao dịch **Saga**. Đây là một chuỗi các giao dịch nhỏ được phối hợp để đảm bảo tính toàn vẹn của dữ liệu giữa các dịch vụ độc lập.
   * **Bước 1:** **Order Service** tạo một đơn hàng với trạng thái PENDING và xuất bản một sự kiện OrderCreated lên **Message Broker**.
   * **Bước 2:** **Payment Service** lắng nghe và nhận sự kiện OrderCreated. Nó thực hiện xử lý thanh toán, sử dụng một **Idempotency Key** duy nhất để ngăn chặn việc bị trừ tiền hai lần nếu yêu cầu được gửi lại.
   * **Bước 3:** Nếu thanh toán thành công, **Payment Service** sẽ xuất bản sự kiện PaymentSucceeded. **Order Service** lắng nghe sự kiện này và cập nhật trạng thái đơn hàng thành CONFIRMED.
   * **Rollback:** Nếu thanh toán thất bại, **Payment Service** sẽ xuất bản sự kiện PaymentFailed. **Order Service** sẽ nhận và cập nhật trạng thái đơn hàng thành FAILED, đảm bảo rằng không có hành động nào khác được thực hiện trên đơn hàng thất bại.

III. Giao hàng bằng Drone & Theo dõi (Drone Delivery & Tracking)

Luồng này được cải tiến để có khả năng chịu lỗi, theo dõi thời gian thực và xử lý dữ liệu hiệu quả.

1. **Phân công giao hàng:** Khi đơn hàng chuyển sang trạng thái CONFIRMED, **Order Service** xuất bản sự kiện OrderConfirmed lên **Message Broker**. Một dịch vụ mới là **Delivery Dispatcher Service** sẽ lắng nghe sự kiện này, tìm kiếm drone khả dụng và gửi yêu cầu giao hàng đến **Delivery & GPS Service**. Cơ chế này cũng bao gồm một **cơ chế dự phòng (fallback)**: nếu không có drone nào sẵn sàng, hệ thống có thể chuyển sang chế độ giao hàng thủ công.
2. **Theo dõi thời gian thực:**
   * **Delivery & GPS Service** nhận dữ liệu GPS liên tục từ drone vật lý.
   * Thay vì lưu trữ trong MongoDB thông thường, dữ liệu GPS được lưu vào một **Time-Series DB** (như InfluxDB) được tối ưu hóa cho dữ liệu theo chuỗi thời gian, giúp ngăn chặn cơ sở dữ liệu bị phình to nhanh chóng.
   * Dịch vụ sử dụng **WebSocket** hoặc **Server-Sent Events (SSE)** để đẩy thông tin vị trí và thời gian đến dự kiến (ETA) theo thời gian thực đến **Client App**, mang lại trải nghiệm theo dõi mượt mà và hiệu quả hơn so với việc liên tục "hỏi" server (polling).
3. **Hoàn thành & dự phòng:** Sau khi drone hoàn tất giao hàng, **Delivery & GPS Service** xuất bản một sự kiện OrderDelivered để các dịch vụ khác có thể cập nhật trạng thái. Nếu drone gặp sự cố, **Delivery Dispatcher Service** có thể tự động thử lại hoặc chuyển sang cơ chế dự phòng.

IV. Quản trị & Giám sát (Admin & Monitoring)

Luồng này được bổ sung các tính năng giám sát chủ động và truy vết dữ liệu để tăng cường khả năng quản lý và khắc phục sự cố.

1. **Logging tập trung & Correlation ID:** Tất cả các dịch vụ đều gửi nhật ký (log) đến một hệ thống tập trung. Mỗi yêu cầu sẽ được gán một **Correlation ID** duy nhất, cho phép các kỹ sư dễ dàng theo dõi toàn bộ hành trình của một yêu cầu xuyên suốt qua nhiều microservices.
2. **Cảnh báo & Phân tích:**
   * Hệ thống giám sát liên tục các chỉ số quan trọng (hiệu năng API, thời gian hoạt động). Khi một chỉ số vượt ngưỡng, một **hệ thống cảnh báo (alerting system)** sẽ tự động gửi thông báo đến đội ngũ kỹ thuật.
   * Một cơ chế **ghi lại hành vi (auditing)** được thêm vào để lưu trữ các hành động quan trọng của người quản trị (admin), giúp tăng cường tính trách nhiệm.
   * Dữ liệu từ các dịch vụ được thu thập và phân tích để tạo ra các báo cáo thời gian thực, cung cấp thông tin kinh doanh sâu sắc cho quản trị viên mà không ảnh hưởng đến hiệu năng của hệ thống chính.