**CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI VÀ CƠ SỞ KIỂM THỬ**

**1.1. Giới thiệu chung**

**1.1.1. Bối cảnh và lý do chọn đề tài**

Trong bối cảnh thị trường giao đồ ăn trực tuyến (Food Delivery) đang bùng nổ, các hệ thống hiện tại vẫn gặp nhiều bất cập như trải nghiệm người dùng không đồng nhất giữa các nền tảng, quy trình đặt hàng phức tạp và thiếu sự minh bạch trong việc theo dõi đơn hàng.

Dự án FoodFast Delivery (tiền thân là DRONE - FAST FOOD DELIVERY) được xây dựng nhằm giải quyết các vấn đề trên. Hệ thống tập trung phục vụ các món ăn đặc trưng Việt Nam (Cơm Tấm, Phở, Bún chả...) và đặc biệt ứng dụng công nghệ Drone vào quy trình vận chuyển. Mục tiêu cốt lõi là tạo ra một hệ thống đồng bộ, minh bạch và có khả năng mở rộng cao**.**

**1.1.2. Tính cấp thiết của việc đảm bảo chất lượng trong hệ thống giao đồ ăn bằng Drone**

Với kiến trúc Microservices hướng sự kiện (Event-Driven) và tích hợp nhiều thành phần phức tạp như thanh toán điện tử (VNPay), điều phối Drone và xử lý thời gian thực (Real-time tracking), việc đảm bảo chất lượng phần mềm là yếu tố sống còn. Một lỗi nhỏ trong xử lý tồn kho hoặc sự cố trong luồng sự kiện Kafka có thể dẫn đến sai lệch đơn hàng, mất doanh thu hoặc trải nghiệm tồi tệ cho người dùng. Do đó, việc xây dựng cơ sở kiểm thử chặt chẽ cho FoodFast Delivery là vô cùng cấp thiết.

**1.2. Phân tích bài toán và Xác định cơ sở kiểm thử (Test Basis)**

**1.2.1. Khảo sát hiện trạng và mô hình hóa quy trình nghiệp vụ (Business Process)**

Hệ thống hoạt động dựa trên sự phối hợp của 5 dịch vụ chính (User, Product, Order, Payment, Delivery). Các quy trình nghiệp vụ trọng yếu cần được kiểm thử bao gồm:

1. Luồng Đặt hàng & Kiểm tra Tồn kho:
   * User đặt món -> Order Service gọi Product Service kiểm tra tồn kho.
   * Xử lý logic: Nếu còn hàng (giữ hàng/tạo đơn Pending) và nếu hết hàng (báo lỗi).
2. Luồng Phục hồi (Compensation Transaction):
   * Xử lý khi thanh toán VNPay thất bại.
   * Cơ chế Rollback: Payment Service báo lỗi -> Product Service hoàn trả tồn kho -> Order Service hủy đơn.
3. Luồng Theo dõi Drone (Real-time Tracking):
   * Mô phỏng vị trí Drone qua Delivery Service -> Publish sự kiện lên Kafka -> Notification Service đẩy xuống Client qua WebSocket.

**1.2.2. Phạm vi công nghệ và kiến trúc kiểm thử tương ứng**

Hệ thống sử dụng kiến trúc Microservices kết hợp Event-Driven, đặt ra các yêu cầu kiểm thử đặc thù:

* Backend: Spring Boot (Java) -> Cần kiểm thử Unit Test (JUnit), Integration Test cho từng Service.
* Giao tiếp: Apache Kafka -> Cần kiểm thử tính toàn vẹn của Message, độ trễ và khả năng chịu lỗi (Consumer/Producer failure).
* Frontend: React.js & React Native -> Kiểm thử giao diện (UI/UX) và luồng dữ liệu trên đa nền tảng.
* Hạ tầng: Docker & Kubernetes -> Kiểm thử khả năng triển khai, Self-healing (tự phục hồi) và Scalability (mở rộng).

**1.2.3. Đặc tả yêu cầu kiểm thử chức năng (Functional Requirements)**

Dựa trên tính năng của các Service, các yêu cầu kiểm thử chức năng bao gồm:

* User Service: Đăng ký, đăng nhập, xác thực JWT, quản lý profile.
* Product Service: CRUD món ăn, cập nhật ảnh, quản lý số lượng tồn kho chính xác.
* Order Service: Thêm/sửa/xóa giỏ hàng, tạo đơn, cập nhật trạng thái đơn hàng (Pending -> Paid -> Delivering -> Delivered).
* Payment Service: Tích hợp cổng VNPay, xử lý callback chính xác (Thành công/Thất bại).
* Delivery Service & Notification: Điều phối trạng thái giao vận, cập nhật vị trí Drone thời gian thực trên bản đồ.

**1.2.4. Đặc tả yêu cầu kiểm thử phi chức năng (Non-functional Requirements)**

* Hiệu năng (Performance): Thời gian phản hồi API chính phải ≤ 500ms.
* Bảo mật (Security): Toàn bộ API được bảo vệ bằng JWT, giao tiếp qua HTTPS.
* Tính sẵn sàng (Availability): Hệ thống có khả năng chịu lỗi cao (Fault Tolerance), Database có cơ chế backup.
* Khả năng mở rộng (Scalability): Các service phải có khả năng scale độc lập trên Kubernetes khi tải tăng cao.
* Giám sát (Monitoring): Dữ liệu được theo dõi real-time qua Prometheus/Grafana.

**1.3. Phạm vi và Mục tiêu nghiên cứu**

**1.3.1. Phạm vi kiểm thử (In-scope & Out-scope)**

* Trong phạm vi (In-scope):
  + Kiểm thử chức năng API của 5 Core Microservices.
  + Kiểm thử luồng sự kiện Kafka (Happy case & Negative case).
  + Kiểm thử giao diện người dùng trên Web và Mobile App.
  + Kiểm thử tích hợp thanh toán Sandbox VNPay.
* Ngoài phạm vi (Out-scope):
  + Kiểm thử phần cứng vật lý của Drone (chỉ sử dụng giả lập sự kiện vị trí).
  + Kiểm thử tải (Load Test) ở quy mô cực lớn (hàng triệu người dùng đồng thời - giới hạn ở mức quy mô dự án).

**1.3.2. Mục tiêu chất lượng cần đạt được**

* Trải nghiệm nhất quán: Đảm bảo luồng đặt hàng mượt mà trên cả Web và Mobile.
* Độ chính xác dữ liệu: Không xảy ra tình trạng lệch tồn kho (Overselling) hoặc mất đơn hàng.
* Tính Real-time: Độ trễ thông báo vị trí Drone và trạng thái đơn hàng ở mức thấp nhất có thể.

**1.4. Phương pháp luận và Quy trình tiếp cận**

**1.4.1. Quy trình phát triển tích hợp kiểm thử (Agile/Scrum & CI/CD)**

Dự án áp dụng quy trình CI/CD (Continuous Integration/Continuous Deployment) tự động hóa hoàn toàn.

* Mã nguồn được quản lý trên Git.
* Sử dụng Docker để đóng gói ứng dụng.
* Quy trình kiểm thử tự động được tích hợp vào pipeline trước khi deploy lên Kubernetes.

**1.4.2. Chiến lược tiếp cận theo mô hình V (V-Model Overview)**

Áp dụng mô hình V biến thể cho Microservices:

1. Unit Testing: Tại tầng Service (Spring Boot).
2. Component Testing: Kiểm thử riêng biệt từng Microservice với Database riêng.
3. Integration Testing: Kiểm thử giao tiếp giữa các Service qua API Gateway và Kafka.
4. System Testing: Kiểm thử toàn trình (End-to-End) từ Frontend đến Backend.

**1.4.3. Điều kiện chấp thuận sản phẩm (User Acceptance Criteria - UAC)**

* Người dùng có thể đặt món, thanh toán thành công qua VNPay và nhìn thấy vị trí Drone di chuyển trên bản đồ.
* Hệ thống tự động hoàn trả tồn kho nếu thanh toán thất bại.
* Admin có thể xem thống kê doanh thu và bản đồ nhiệt (Heatmap) hoạt động của Drone.

**1.5. Kế hoạch triển khai tổng thể**

**1.5.1. Lộ trình phát triển (Development Roadmap)**

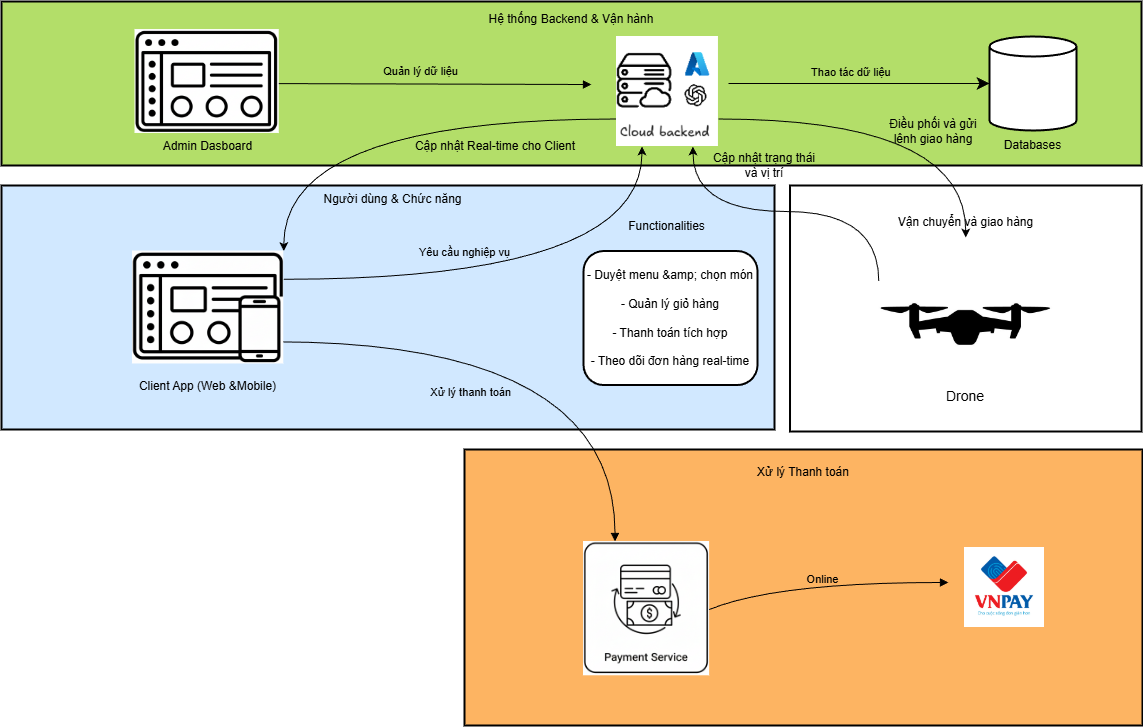
1. Thiết kế kiến trúc hệ thống và Database (PostgreSQL).
2. Phát triển Core Services (User, Product, Order).
3. Tích hợp Payment (VNPay) và Message Broker (Kafka).
4. Phát triển Delivery Service và Notification (Real-time).
5. Xây dựng Frontend (Web/Mobile) và Admin Portal.

**1.5.2. Lộ trình kiểm thử (Testing Roadmap)**

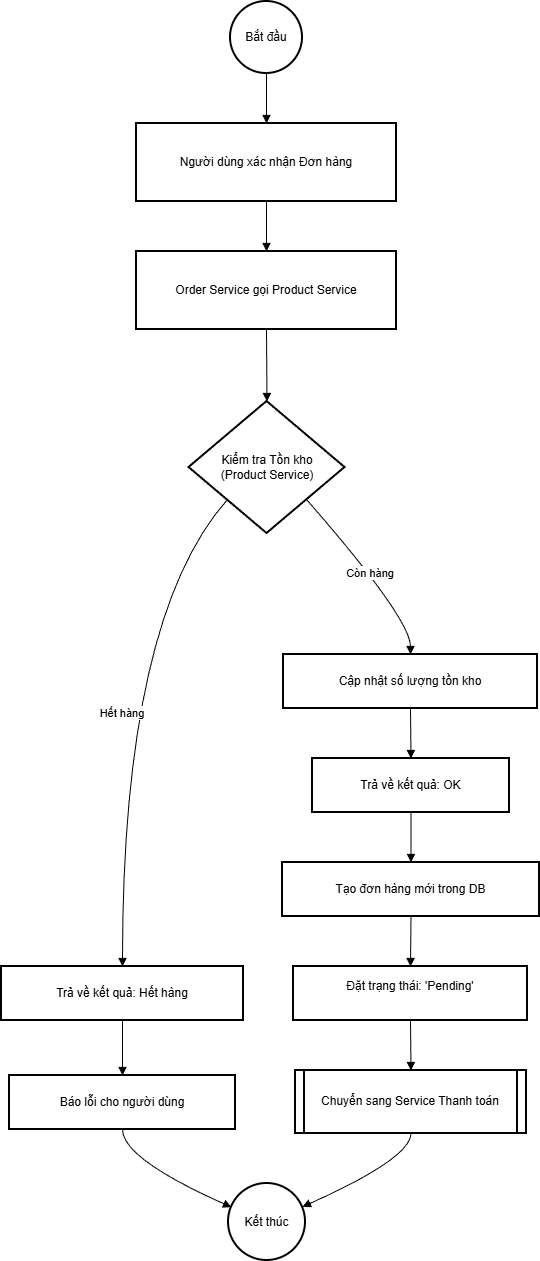
1. Giai đoạn 1: Unit Test cho các chức năng CRUD cơ bản.
2. Giai đoạn 2: Integration Test cho luồng đặt hàng và kiểm tra tồn kho (Order <-> Product).
3. Giai đoạn 3: Kiểm thử luồng sự kiện bất đồng bộ và cơ chế Rollback (Payment <-> Kafka <-> Order).
4. Giai đoạn 4: UAT và kiểm thử hiệu năng, giám sát qua Grafana.

**CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG (CƠ SỞ CHO KIỂM THỬ TÍCH HỢP)**

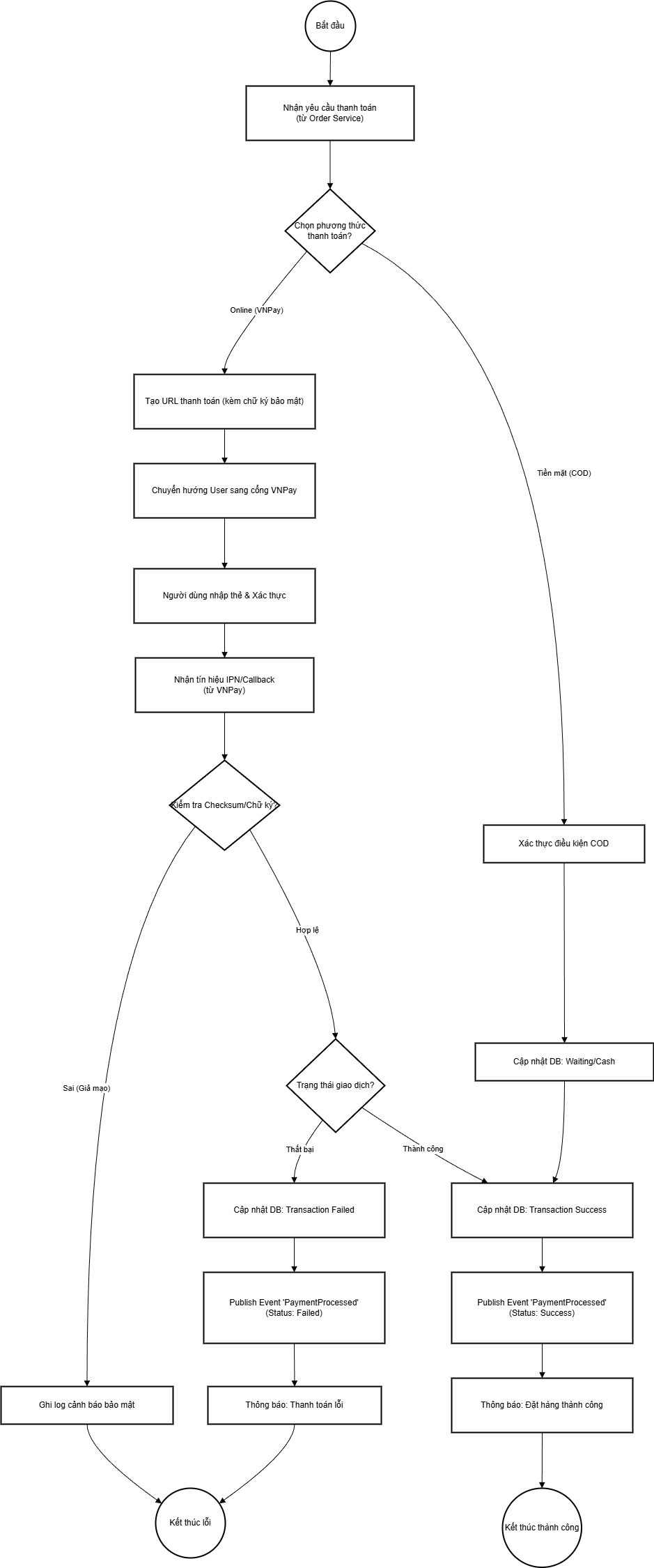
**2.1. Thiết kế hệ thống (System Design)**



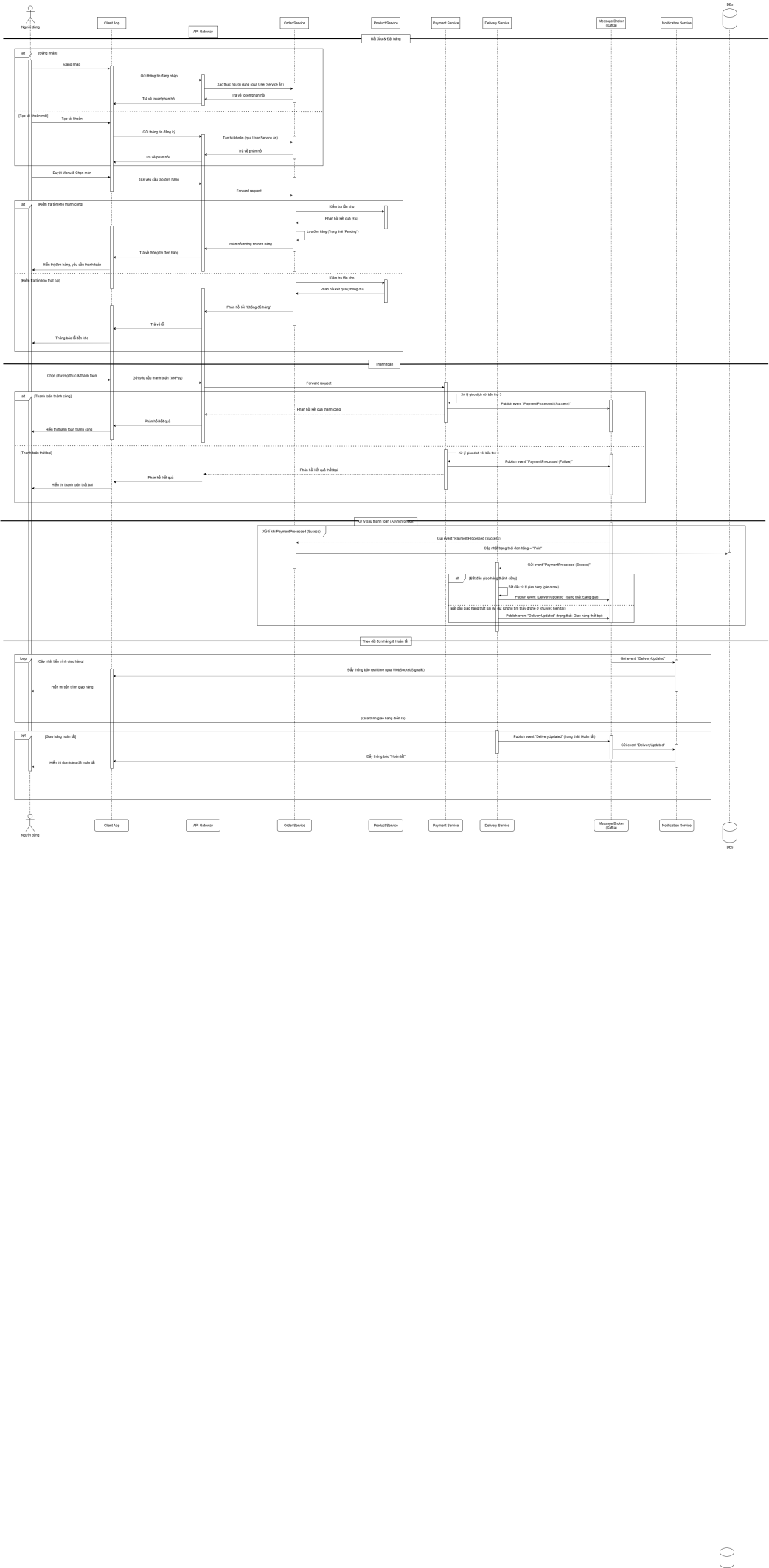
Hình 3. Sơ đồ Luồng Vận hành Hệ thống (Operational Flow Diagram)

****

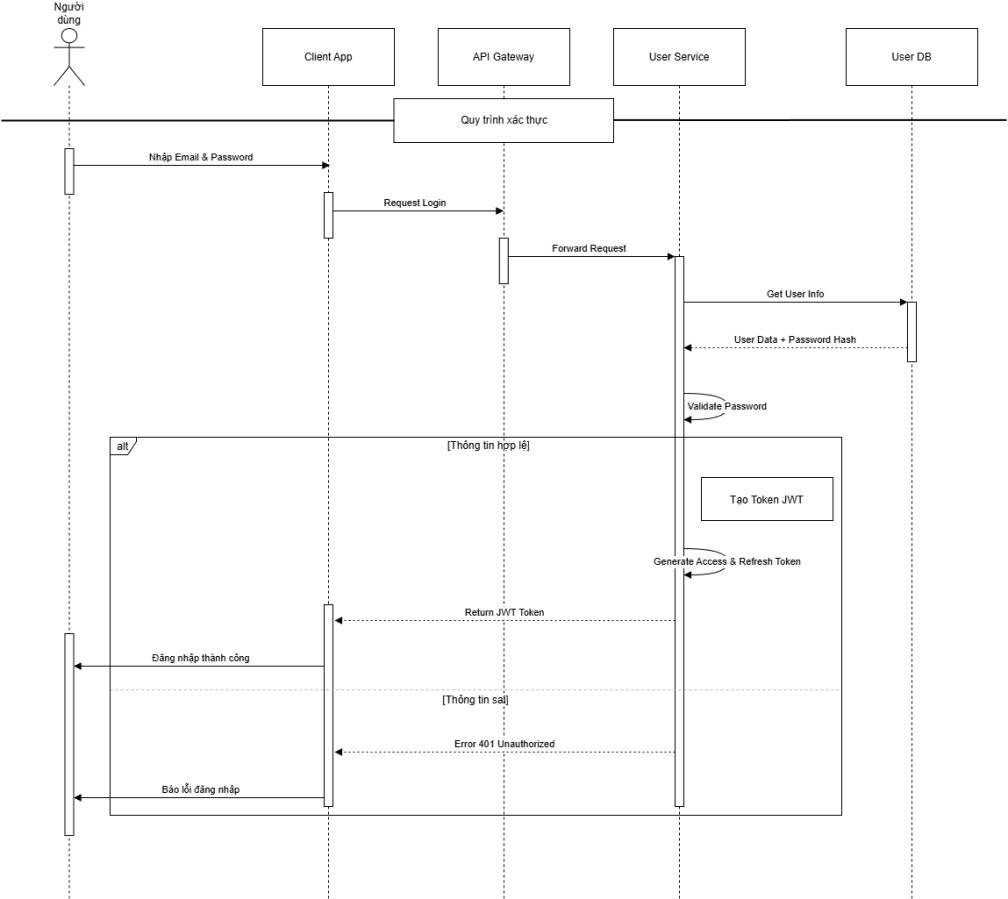
Hình 4. Quy trình Kiểm tra Tồn Kho

****

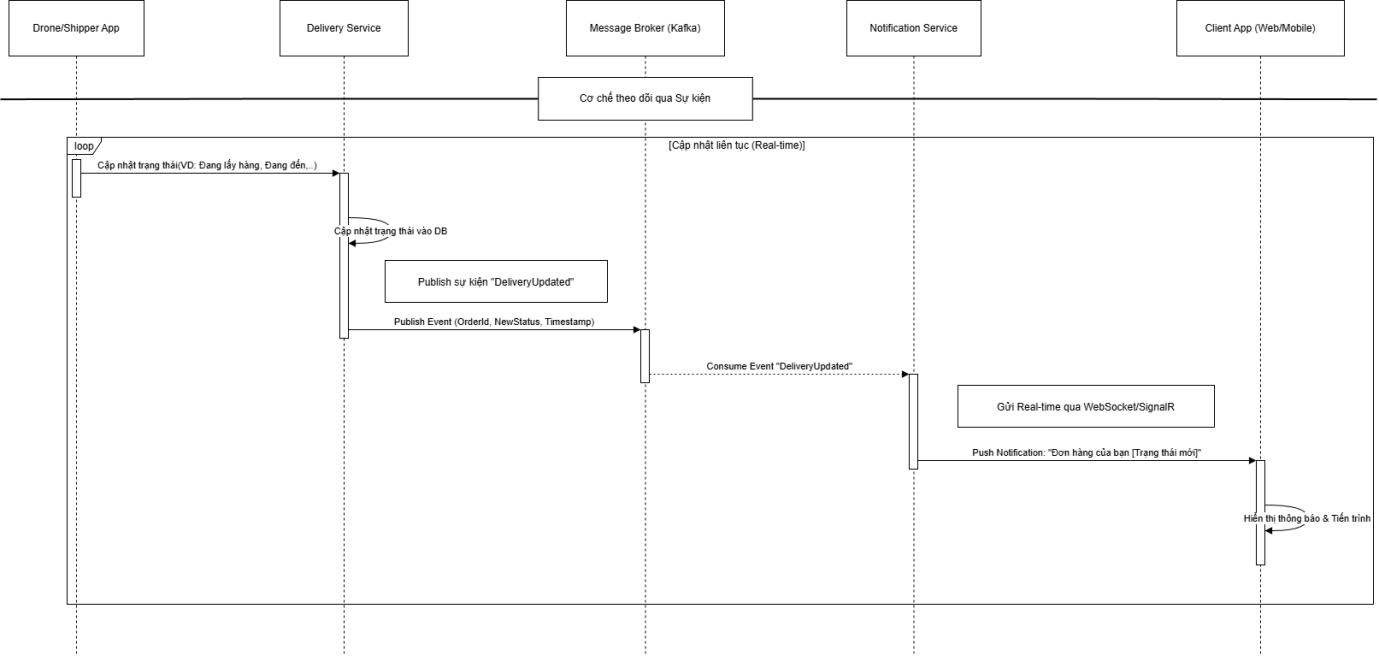
Hình 6. Quy trình Thanh Toán



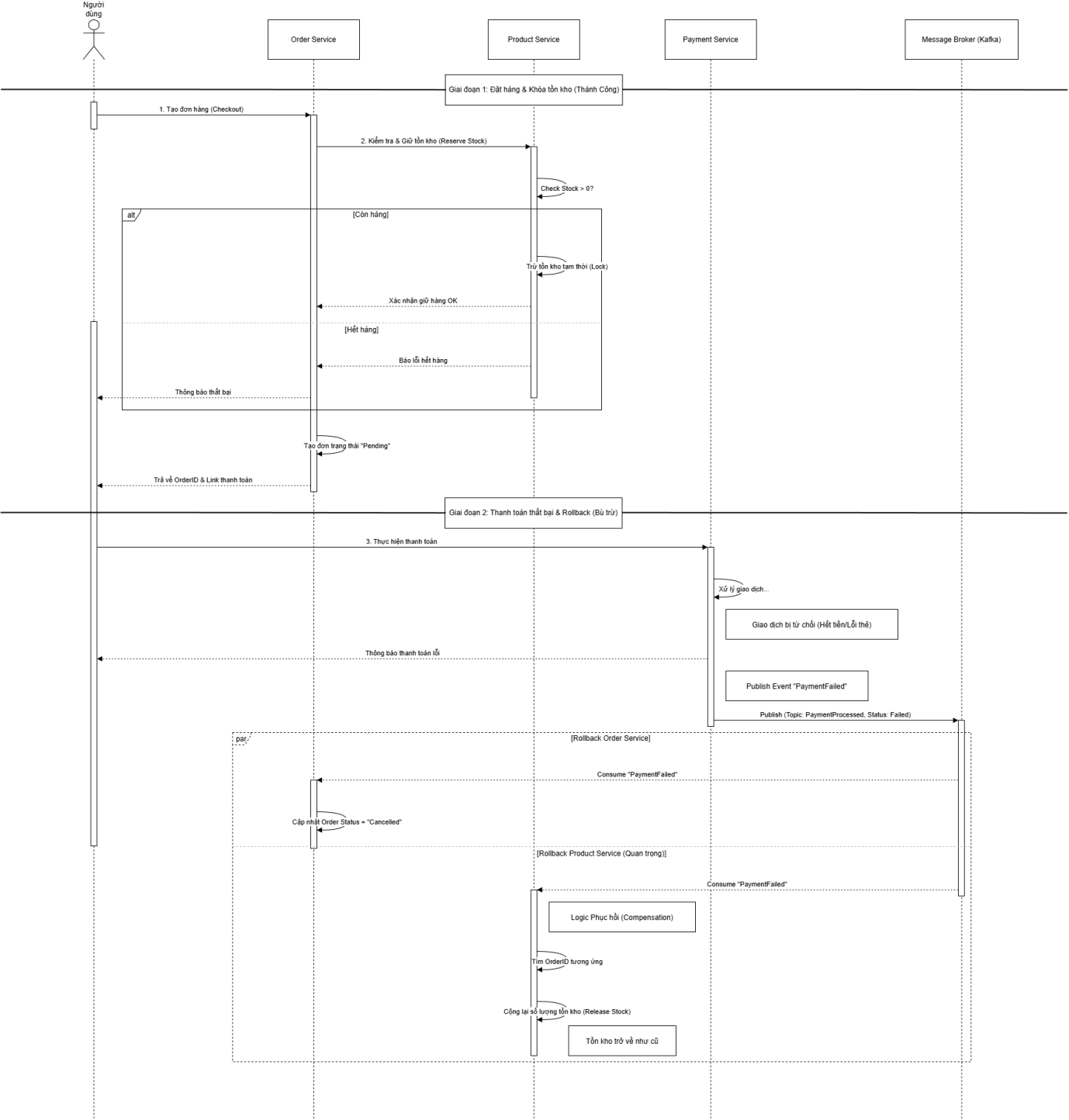
Hình 7. Luồng Tương tác Chi tiết

****

Hình 10. Luồng Xác thực & Đăng nhập

****

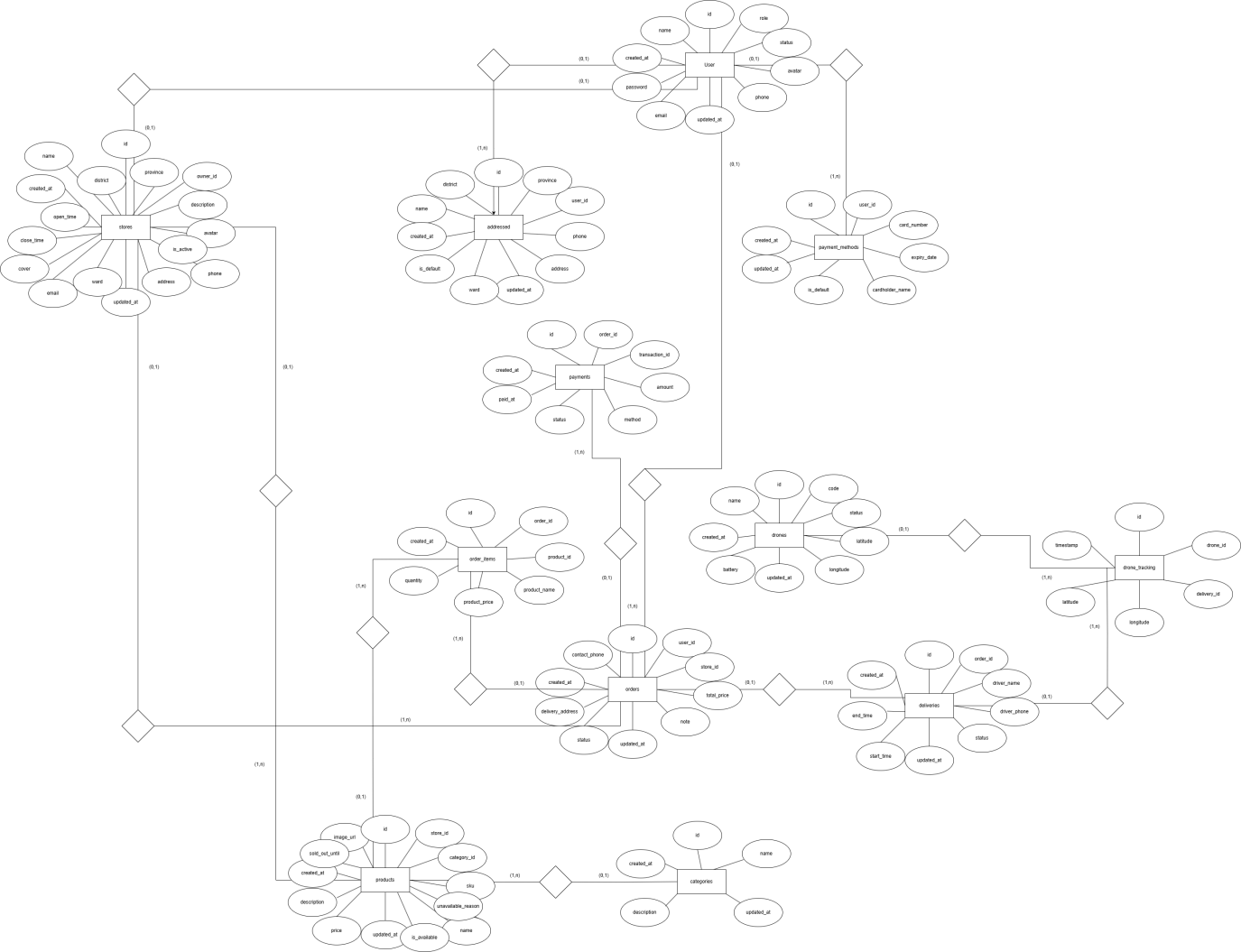
Hình 9. Luồng Theo dõi Giao hàng Thời gian thực

****

Hình 11. Luồng Phục hồi Tồn Kho

**2.1.1. Thiết kế chức năng và biểu đồ luồng dữ liệu (DFD/Activity Diagram)**

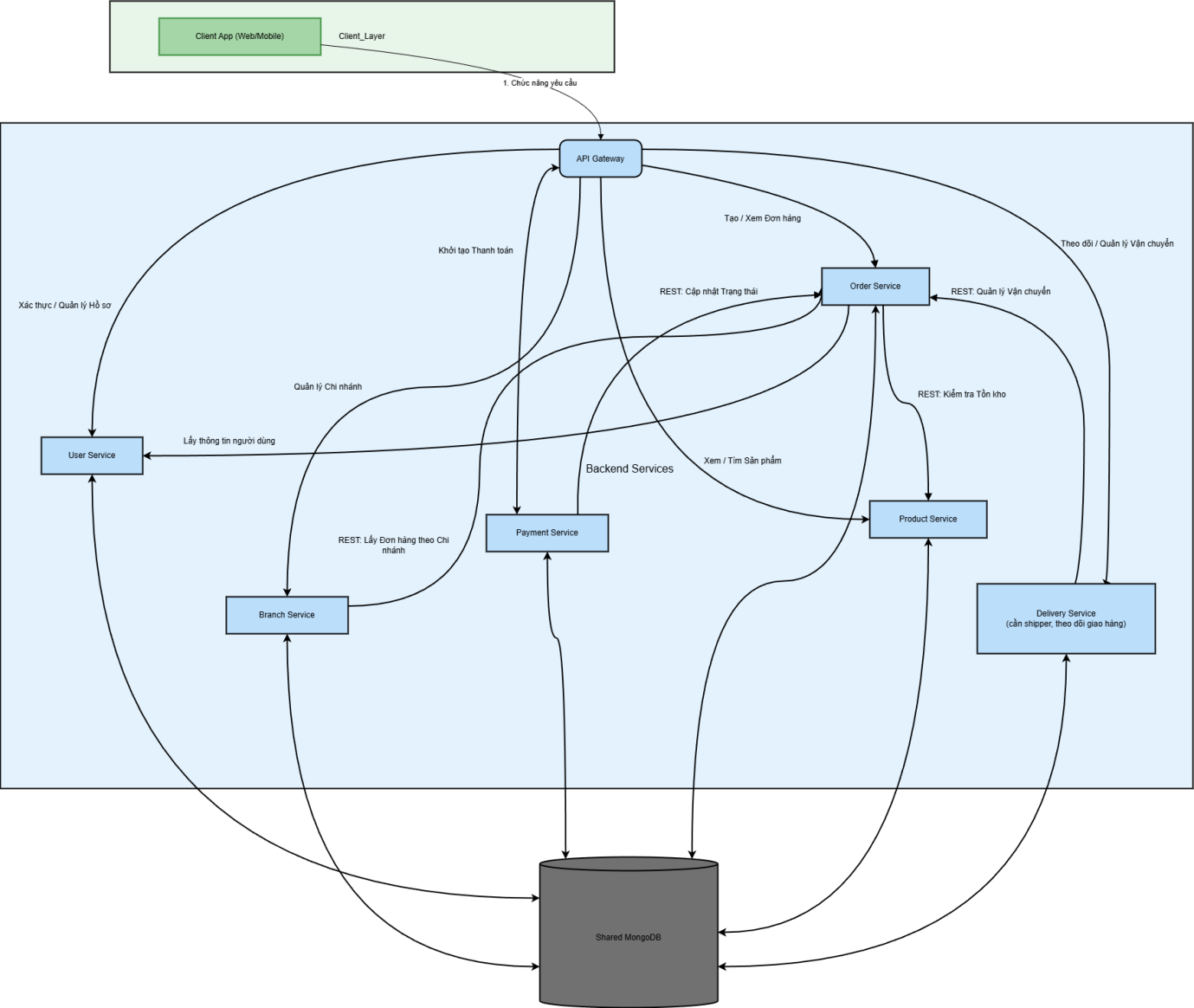
**2.1.2. Thiết kế cơ sở dữ liệu (ERD)**

****

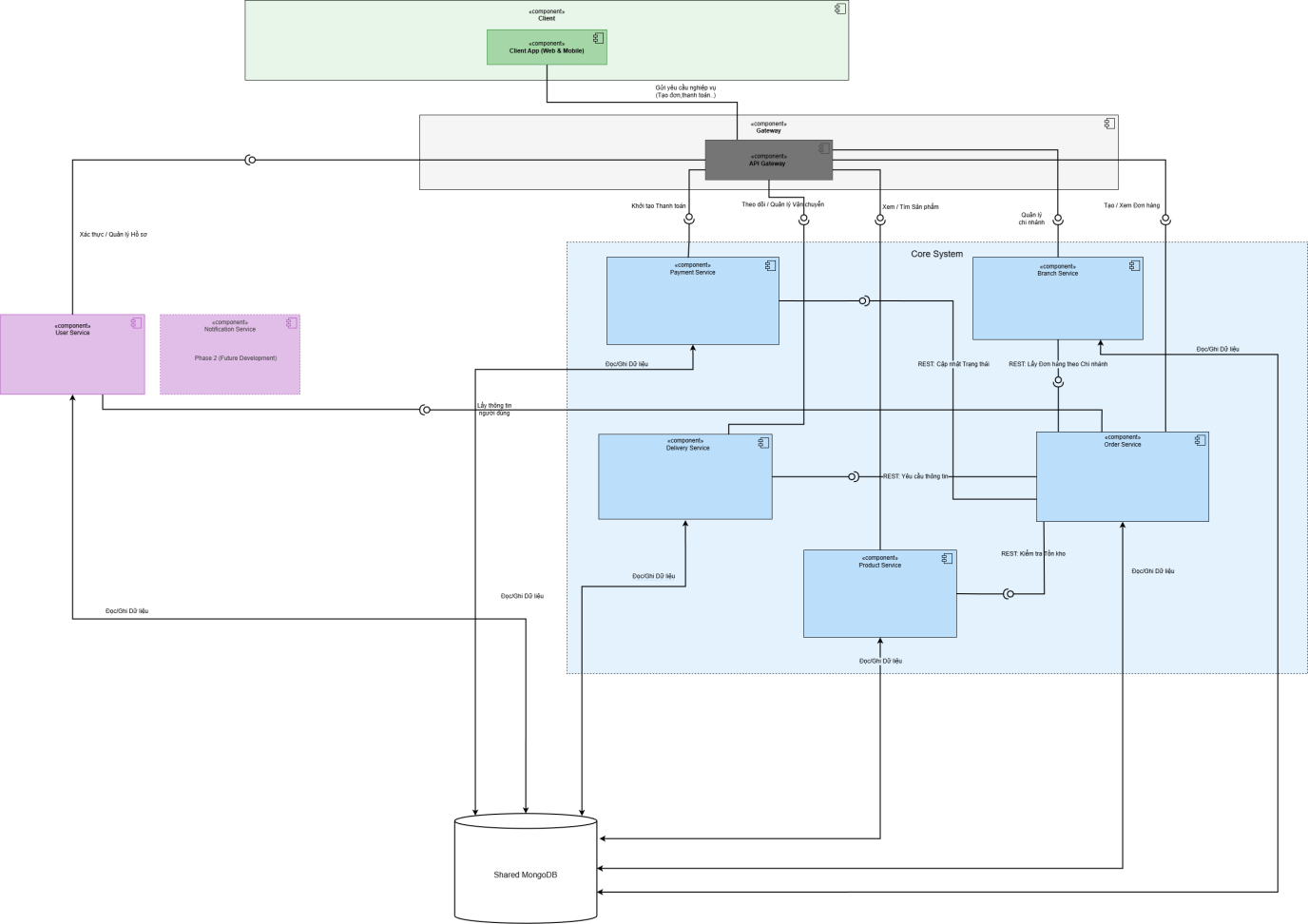
Hình 13. Sơ đồ ERD

**2.1.3. Thiết kế giao diện người dùng (UI/UX)**

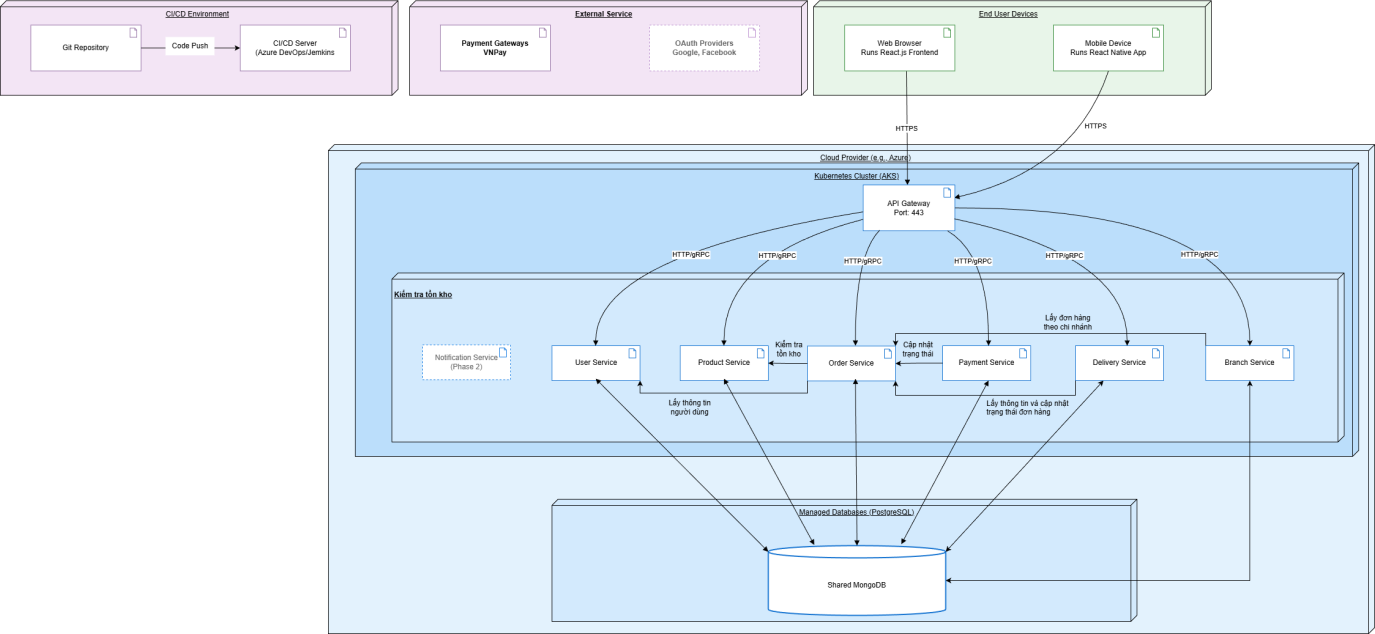
**2.2. Thiết kế kiến trúc (Architecture Design)  
2.2.1. Mô hình kiến trúc phần mềm**



Hình 1. Sơ đồ kiến trúc Microservices hướng sự kiện

****

Hình 14. Sơ đồ Component

**2.2.2. Sơ đồ triển khai và các điểm giao tiếp (Interfaces)  
**

Hình 15. Sơ đồ Triển khai

**CHƯƠNG 3: KẾ HOẠCH KIỂM THỬ**

**3.1. Tổng quan kế hoạch kiểm thử**

**3.1.1. Mục tiêu kiểm thử**

Mục tiêu của chương này là xác định quy trình và phương pháp đảm bảo chất lượng cho **Website FoodFast**. Việc kiểm thử nhằm đảm bảo khách hàng có thể tìm kiếm món ăn, đặt hàng và thanh toán mượt mà, đồng thời hệ thống quản trị vận hành chính xác các đơn hàng.

**3.1.2. Phạm vi kiểm thử (Scope of Test)**

Phạm vi kiểm thử tập trung vào các luồng nghiệp vụ cốt lõi của việc kinh doanh đồ ăn trực tuyến:

* **Các phần được kiểm thử:**
  + Giao diện người dùng (Front-end): Trang chủ, Danh sách món ăn (Menu), Chi tiết món ăn, Giỏ hàng, Trang thanh toán, trang theo dõi đơn hàng
  + Chức năng quản trị (Back-end): Quản lý danh mục (Thêm/Sửa/Xóa món, chi nhánh), Quản lý đơn hàng (Tiếp nhận/Hủy/Giao), quản lý drone, Cập nhật trạng thái đơn hàng
  + Tính đúng đắn của dữ liệu: Tính tổng tiền, phí vận chuyển
* **Các phần không kiểm thử:** Các lỗi phát sinh do đường truyền mạng của người dùng yếu, hoặc lỗi từ phía cổng thanh toán ngân hàng (chỉ kiểm thử việc gửi/nhận yêu cầu thanh toán).

**3.1.3 Quy trình kiểm thử**

*(Sơ đồ quy trình kiểm thử phần mềm)*

**[Vị trí chèn hình ảnh sơ đồ nhóm]** *Lưu ý: Bạn nên chèn sơ đồ quy trình từ lúc: Nhận yêu cầu -> Viết Test Case -> Thực thi Test -> Log Bug -> Fix Bug -> Retest -> Đóng Bug.*

**3.2. Đối tượng kiểm thử**

Hệ thống được chia thành các phân hệ chính cần kiểm thử như sau:

**3.2.1. Phân hệ Khách hàng (End-User):**

* **Đăng ký/Đăng nhập:** Đảm bảo bảo mật thông tin khách hàng.
* **Tìm kiếm & Lọc món ăn:** Kiểm tra độ chính xác khi tìm theo tên, theo loại (đồ uống, đồ ăn nhanh, cơm...), hoặc theo mức giá.
* **Giỏ hàng (Cart):** Kiểm tra chức năng thêm món, xóa món, cập nhật số lượng, tính tạm tính.
* **Đặt hàng (Checkout):** Kiểm tra việc nhập địa chỉ giao hàng, chọn phương thức thanh toán và xác nhận đơn.
* **Theo dõi đơn hàng:** Kiểm tra tiến trình giao hàng, quản đường di chuyển của drone

**3.2.2. Phân hệ Quản trị (Admin):**

* **Quản lý Thực đơn (Menu Management):** Kiểm tra chức năng CRUD (Thêm, Đọc, Sửa, Xóa) món ăn, cập nhật giá và hình ảnh món.
* **Quản lý Đơn hàng (Order Processing):** Kiểm tra quy trình cập nhật trạng thái đơn hàng (Chờ xác nhận -> Đang làm -> Đang giao -> Hoàn thành).
* **Quản lý chi nhánh:** Thêm, sửa, đóng cửa, cập nhật thông tin chi nhánh
* **Quản lý drone:** Theo dõi tình trạng pin, thay đổi trạng thái, thêm sửa xóa drone

**3.2.3. Cơ sở dữ liệu (Database):**

Kiểm tra tính toàn vẹn dữ liệu khi có nhiều người cùng đặt một món tại cùng một thời điểm.

**3.3. Chiến lược kiểm thử**

**3.3.1. Các cấp độ kiểm thử (Test Levels)**

Dự án áp dụng mô hình kiểm thử theo tháp (Testing Pyramid) bao gồm:

* Kiểm thử đơn vị (Unit Testing): Do đội ngũ phát triển thực hiện, tập trung vào các hàm xử lý logic nhỏ (ví dụ: hàm tính tổng tiền giỏ hàng, hàm validate định dạng email).
* Kiểm thử tích hợp (Integration Testing): Kiểm tra sự tương tác giữa các module (ví dụ: tích hợp giữa module Giỏ hàng và module Thanh toán; tương tác giữa Backend và Database).
* Kiểm thử hệ thống (System Testing): Kiểm thử toàn bộ ứng dụng như một hệ thống hoàn chỉnh từ đầu đến cuối (End-to-End) trên môi trường giống thực tế nhất.
* Kiểm thử chấp nhận (Acceptance Testing - UAT): Kiểm thử dựa trên kịch bản nghiệp vụ thực tế để xác nhận hệ thống đáp ứng nhu cầu kinh doanh của việc đặt đồ ăn

**3.3.2. Phương pháp tiếp cận**

* **Kiểm thử chức năng (Functional Testing):** Tập trung kiểm tra xem các tính năng có hoạt động đúng như mô tả không
* **Kiểm thử giao diện (UI/UX Testing):** Đảm bảo hình ảnh món ăn hiển thị đẹp, không bị vỡ khung, menu hiển thị tốt trên cả máy tính và điện thoại.
* **Kiểm thử biên (Boundary Testing):** Kiểm tra các trường hợp nhập số lượng món ăn cực lớn, hoặc nhập số lượng âm, nhập giá tiền 0 đồng.

**3.3.3. Điều kiện chấp nhận (Acceptance Criteria)**

Hệ thống được chấp nhận khi:

* Luồng đặt hàng thành công từ A-Z mà không gặp lỗi crash.
* Tổng tiền thanh toán (bao gồm giá món + ship - giảm giá) phải chính xác tuyệt đối.
* Thông báo đơn hàng mới phải hiển thị ngay lập tức bên trang Admin.
* Không còn lỗi mức độ Nghiêm trọng (Critical) hoặc Cao (High).

**3.4. Triển khai kiểm thử**

**3.4.1.** Thiết lập môi trường kiểm thử (Test Environment)

* Phần cứng (Hardware):
  + Máy tính cá nhân (Laptop): Windows 10/11, RAM 8GB+.
  + Thiết bị di động: Smartphone Android và iOS (để test giao diện Responsive).
* Phần mềm (Software):
  + Trình duyệt: Google Chrome (bản mới nhất), Firefox, Edge.
  + Hệ quản trị CSDL: MongoDB
  + Web Server: Node.js / Apache / IIS.
* **Mạng (Network):** Kết nối Internet ổn định, giả lập mạng chậm (3G) để test hiệu năng loading.

**3.4.2.** Công cụ kiểm thử và quản lý lỗi (Test Tools)

* **Quản lý Test Case & Bug:** Sử dụng **Excel** (hoặc Jira/Trello) để viết test case và theo dõi trạng thái lỗi.
* **Kiểm thử API:** Sử dụng **Postman** để gửi request và kiểm tra response JSON.
* **Kiểm thử Hiệu năng:** Sử dụng **JMeter** để giả lập người dùng ảo.
* **Kiểm thử Tự động (nếu có):** Sử dụng **Selenium WebDriver** cho các kịch bản hồi quy cơ bản.
* **Chụp ảnh/Quay màn hình lỗi:** Snipping Tool, OBS Studio.

**3.4.3. Các mốc thời gian (Milestones)**

* **Giai đoạn 1:** Kiểm thử Unit Test các hàm xử lý giá tiền (Ngày: ...).
* **Giai đoạn 2:** Kiểm thử tích hợp chức năng Giỏ hàng và Đặt hàng (Ngày: ...).
* **Giai đoạn 3:** Kiểm thử toàn hệ thống và sửa lỗi (Ngày: ...).
* **Kết quả:** Bảng thống kê các lỗi đã phát hiện và trạng thái (Đã sửa/Chưa sửa).

**3.5. Các kế hoạch mở rộng**

Để nâng cao chất lượng dự án trong tương lai (nếu có thêm thời gian):

* **Kiểm thử đơn vị (Unit Testing):** Mở rộng độ bao phủ code (code coverage) lên trên 80% cho các module xử lý logic nghiệp vụ.
* **Kiểm thử tích hợp (Integration Testing):** Tập trung sâu vào API kết nối giữa Front-end và Back-end, đảm bảo dữ liệu JSON trả về đúng định dạng.
* **Kiểm thử hiệu năng (Performance Testing):** Sử dụng công cụ (như JMeter) để giả lập 100-1000 người cùng đặt món một lúc xem website có bị sập không.
* **Kiểm thử bảo mật (Security Testing):** Chống SQL Injection tại các ô tìm kiếm và bảo mật API.

**CHƯƠNG 4: THIẾT KẾ KIỂM THỬ (TEST DESIGN)**

**4.1.** Tổng quan và Pipeline tiếp cận (Mô hình V)

**4.1.1.** Ánh xạ giữa giai đoạn phát triển và cấp độ thiết kế kiểm thử

Trong mô hình V áp dụng cho dự án website kinh doanh cây cảnh, quá trình thiết kế kiểm thử được thực hiện song song với quá trình phát triển phần mềm để đảm bảo tính bao quát và phát hiện lỗi sớm:

* **Giai đoạn Phân tích yêu cầu (Requirement Analysis):** Đội kiểm thử xây dựng kế hoạch kiểm thử chấp nhận (UAT) để đảm bảo các nghiệp vụ như "Đặt đồ ăn", "Thanh toán", "Giao hàng" đúng với nhu cầu người dùng.
* **Giai đoạn Thiết kế hệ thống (System Design):** Thiết kế các ca kiểm thử hệ thống (System Test Cases) dựa trên biểu đồ luồng dữ liệu và thiết kế giao diện UI/UX.
* **Giai đoạn Thiết kế chi tiết (Module Design):** Thiết kế kiểm thử tích hợp (Integration Test) cho các API giao tiếp giữa Frontend và Backend.

**4.1.2.** Quy trình thiết kế Test Case

Quy trình thiết kế kịch bản kiểm thử tuân theo các bước chuẩn hóa:

1. **Xác định điều kiện kiểm thử (Test Condition):** Dựa trên SRS (Đặc tả yêu cầu).
2. **Thiết kế Test Case:** Xác định dữ liệu đầu vào (Input), các bước thực hiện (Steps) và kết quả mong đợi (Expected Output).
3. **Review Test Case:** Nhóm trưởng kiểm tra độ bao phủ của Test Case.
4. **Cập nhật RTM:** Ánh xạ Test Case vào Ma trận truy vết yêu cầu.

**4.2.** Ma trận truy vết yêu cầu (Requirements Traceability Matrix - RTM)

Ma trận RTM giúp đảm bảo không có yêu cầu chức năng nào bị bỏ sót trong quá trình kiểm thử.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Yêu cầu (Req ID)** | **Tên chức năng / Mô tả yêu cầu** | **ID Test Case (TC ID)** | **Mức độ ưu tiên (Priority)** | **Trạng thái (Status)** |
| **I. MODULE XÁC THỰC & TÀI KHOẢN (AUTHENTICATION)** | | | | |
| **REQ\_AUTH\_01** | Đăng ký tài khoản mới (Khách hàng) | TC\_AUTH\_01, TC\_AUTH\_02 | High | Pass |
| **REQ\_AUTH\_02** | Đăng nhập hệ thống (Email/SĐT & Mật khẩu) | TC\_AUTH\_03, TC\_AUTH\_04 | Critical | Pass |
| **REQ\_AUTH\_03** | Đăng nhập qua mạng xã hội (Google/Facebook) | TC\_AUTH\_05 | Medium | Fail |
| **REQ\_AUTH\_04** | Quên mật khẩu & Khôi phục qua Email | TC\_AUTH\_06, TC\_AUTH\_07 | High | Fail |
| **REQ\_AUTH\_05** | Đổi mật khẩu & Cập nhật hồ sơ cá nhân | TC\_AUTH\_08 | Low | Pass |
| **II. MODULE TÌM KIẾM & ĐẶT MÓN (CUSTOMER)** | | | | |
| **REQ\_FOOD\_01** | Tìm kiếm món ăn theo từ khóa | TC\_SEARCH\_01, TC\_SEARCH\_02 | High | Pass |
| **REQ\_FOOD\_02** | Lọc món ăn theo danh mục, giá, đánh giá | TC\_SEARCH\_03, TC\_SEARCH\_04 | Medium | Fail |
| **REQ\_FOOD\_03** | Xem chi tiết món ăn (Hình ảnh, Mô tả, Topping) | TC\_VIEW\_01 | Medium | Pass |
| **REQ\_CART\_01** | Thêm món vào giỏ hàng (Check số lượng, Option) | TC\_CART\_01, TC\_CART\_02 | Critical | Pass |
| **REQ\_CART\_02** | Cập nhật giỏ hàng (Tăng/giảm món, Xóa món) | TC\_CART\_03, TC\_CART\_04 | High | Pass |
| **REQ\_CART\_03** | Nhập mã giảm giá (Voucher) & Tính lại tổng tiền | TC\_PROMO\_01, TC\_PROMO\_02 | Medium | **Fail** |
| **III. MODULE THANH TOÁN & ĐƠN HÀNG (ORDER & PAYMENT)** | | | | |
| **REQ\_ORD\_01** | Tạo đơn hàng mới & Chọn địa chỉ giao hàng | TC\_ORD\_01, TC\_ORD\_02 | Critical | Pass |
| **REQ\_PAY\_01** | Thanh toán tiền mặt (COD) | TC\_PAY\_01 | High | Pass |
| **REQ\_PAY\_02** | Thanh toán Online (Ví điện tử/Ngân hàng) | TC\_PAY\_02, TC\_PAY\_03 | High | Pass |
| **REQ\_ORD\_02** | Hủy đơn hàng (Trước khi nhà hàng xác nhận) | TC\_ORD\_03 | Medium | Pass |
| **REQ\_ORD\_03** | Xem lịch sử đơn hàng đã đặt | TC\_ORD\_04 | Low | Pass |
| **IV. MODULE GIAO HÀNG & THEO DÕI (DELIVERY/DRONE)** | | | | |
| **REQ\_SHIP\_01** | Tính phí vận chuyển dựa trên khoảng cách (Km) | TC\_SHIP\_01, TC\_SHIP\_02 | Critical | Fail |
| **REQ\_SHIP\_02** | Theo dõi vị trí tài xế/Drone theo thời gian thực | TC\_TRACK\_01 | High | Pass |
| **REQ\_SHIP\_03** | Cập nhật trạng thái đơn (Shipping -> Delivered) | TC\_SHIP\_03 | Critical | Pass |
| **REQ\_SHIP\_04** | Gửi thông báo khi đơn hàng gần đến nơi | TC\_NOTIFY\_01 | Medium | **Pending** |
| **V. YÊU CẦU PHI CHỨC NĂNG (NON-FUNCTIONAL)** | | | | |
| **REQ\_PERF\_01** | Hệ thống chịu tải 1000 user truy cập đồng thời | TC\_LOAD\_01 | High | Pending |
| **REQ\_PERF\_02** | Thời gian phản hồi API tìm kiếm < 2 giây | TC\_PERF\_01 | High | Pass |
| **REQ\_SEC\_01** | Chống tấn công SQL Injection tại ô tìm kiếm | TC\_SEC\_01 | Critical | Pass |
| **REQ\_UI\_01** | Hiển thị tốt trên Mobile (Responsive Design) | TC\_UI\_01, TC\_UI\_02 | Medium | Pass |

**4.3. Kỹ thuật và Thiết kế kịch bản kiểm thử**

4.3.1. Các kỹ thuật thiết kế áp dụng

* **Phân vùng tương đương (Equivalence Partitioning):**

*Ví dụ:* Khoảng cách giao hàng để tính phí ship.

Vùng 1 (Free ship): 0 - 2km.

Vùng 2 (Phí tiêu chuẩn): > 2km đến 10km.

Vùng 3 (Không hỗ trợ): > 10km.

* **Phân tích giá trị biên (Boundary Value Analysis):**

Kiểm tra các mốc đơn hàng tối thiểu (Min order) để được giảm giá: Ví dụ đơn 100k. Test tại các giá trị: 99k (Không giảm), 100k (Được giảm), 101k (Được giảm).

* **Bảng quyết định (Decision Table):**

Sử dụng cho logic **Áp dụng Voucher Freeship**:

Điều kiện 1: Là thành viên Gold (Yes/No).

Điều kiện 2: Đơn hàng > 200k (Yes/No).

Điều kiện 3: Khoảng cách < 5km (Yes/No).

Kết quả: Được Freeship hay không.

4.3.2. Thiết kế kịch bản kiểm thử chức năng (Functional Test Cases)

\* Kịch bản kiểm thử Đăng ký/Đăng nhập

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Tên kịch bản** | **Các bước thực hiện** | **Dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế (Actual Result)** | **Trạng thái** |
| **TC\_AUTH\_01** | Đăng ký thành công | 1. Nhập thông tin hợp lệ.  2. Nhấn Đăng ký. | Email: newuser@gmail.com    Pass: Abc@123 | Thông báo thành công, chuyển sang trang Login. | Hệ thống hiển thị thông báo và chuyển trang đúng luồng. | **Pass** |
| **TC\_AUTH\_02** | Đăng ký trùng Email | 1. Nhập Email đã tồn tại.  2. Nhấn Đăng ký. | Email: existed@gmail.com | Báo lỗi: "Email đã tồn tại". | Hệ thống hiển thị lỗi "Email đã tồn tại" màu đỏ. | **Pass** |
| **TC\_AUTH\_03** | Đăng nhập thành công | 1. Nhập đúng Email/Pass.  2. Nhấn Đăng nhập. | Email: valid@gmail.com    Pass: Abc@123 | Vào trang chủ, hiển thị tên user trên Header. | Header hiển thị "Xin chào, Nguyen Van A". | **Pass** |
| **TC\_AUTH\_04** | Đăng nhập sai Pass | 1. Nhập sai Pass.  2. Nhấn Đăng nhập. | Pass: WrongPass | Báo lỗi: "Sai thông tin đăng nhập". | Hệ thống báo lỗi chính xác, không cho login. | **Pass** |

\* Kịch bản kiểm thử Tìm kiếm món ăn

| **ID** | **Tên kịch bản** | **Các bước thực hiện** | **Dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế (Actual Result)** | **Trạng thái** |
| --- | --- | --- | --- | --- | --- | --- |
| **TC\_SRCH\_01** | Tìm đúng tên món | 1. Nhập từ khóa.  2. Nhấn Enter. | Keyword: "Cơm tấm" | List món ăn chứa từ "Cơm tấm" hiện ra. | Hiển thị các món ăn liên quan đúng từ khóa. | **Pass** |
| **TC\_SRCH\_02** | Tìm không thấy | 1. Nhập từ khóa lạ.  2. Nhấn Enter. | Keyword: "xyz123" | Thông báo "Không tìm thấy kết quả". | Hiển thị icon rỗng và thông báo như mong đợi. | **Pass** |
| **TC\_SRCH\_03** | Lọc theo danh mục | 1. Click icon "Trà sữa". | Category: "Milk Tea" | Chỉ hiện các món thuộc loại Trà sữa. | Danh sách lọc đúng, không lẫn món ăn mặn. | **Pass** |

\* Kịch bản kiểm thử Giỏ hàng và Đặt hàng

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Tên kịch bản** | **Các bước thực hiện** | **Dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế (Actual Result)** | **Trạng thái** |
| **TC\_CART\_01** | Thêm món + Topping | 1. Chọn món + Size L.    2. Thêm vào giỏ. | Món: 30k    Size: +10k | Giỏ hàng hiện món với giá 40k. | Món vào giỏ, tổng tiền hiển thị 40.000đ. | **Pass** |
| **TC\_CART\_02** | Tăng số lượng | 1. Bấm nút (+) trong giỏ. | SL: 1 -> 2 | Tổng tiền nhân đôi. | Số lượng tăng lên 2, tổng tiền cập nhật ngay lập tức. | **Pass** |
| **TC\_ORD\_01** | Áp Voucher hợp lệ | 1. Nhập mã giảm giá.    2. Bấm Áp dụng. | Code: GIAM10K | Tổng tiền giảm 10k. | Hệ thống trừ đúng 10k vào tổng thanh toán. | Fail |
| **TC\_ORD\_02** | Voucher không đạt | 1. Nhập mã yêu cầu đơn 200k cho đơn 50k. | Code: GIAM50K | Báo lỗi: "Đơn hàng chưa đủ điều kiện". | Hệ thống chặn voucher và hiển thị lý do đúng. | Fail |

\* Kịch bản kiểm thử Thanh toán và Vận chuyển

| **ID** | **Tên kịch bản** | **Các bước thực hiện** | **Dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế (Actual Result)** | **Trạng thái** |
| --- | --- | --- | --- | --- | --- | --- |
| **TC\_PAY\_01** | Tính phí Ship | 1. Chọn địa chỉ giao.  2. Xem phí ship. | Khoảng cách: 5km | Phí ship hiển thị đúng công thức (ví dụ 25k). | Phí ship hiển thị 25.000đ (Đúng theo API Google Maps). | **Fail** |
| **TC\_PAY\_02** | Đặt hàng COD | 1. Chọn Tiền mặt.  2. Bấm Đặt đơn. | - | Tạo đơn thành công, trạng thái "Chờ xác nhận". | Đơn hàng được tạo, Admin nhận được đơn mới. | **Pass** |
| **TC\_PAY\_03** | Thanh toán Online | 1. Chọn thanh toán online  2. Thanh toán thẻ test. |  | Trả về trang "Giao dịch thành công". | Chuyển hướng đúng, đơn cập nhật trạng thái "Đã thanh toán". | **Pass** |
| **TC\_SHIP\_01** | Tracking Drone/Xe | 1. Xem chi tiết đơn.  2. Nhìn bản đồ. | - | Icon drone di chuyển theo tọa độ thực. | Icon cập nhật vị trí mỗi 5 giây đúng thiết kế. | **Pass** |

\* Kịch bản kiểm thử Drone Giao hàng (Drone Delivery Edge Cases)

| **ID** | **Tên kịch bản** | **Các bước thực hiện** | **Dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế** | **Trạng thái** |
| --- | --- | --- | --- | --- | --- | --- |
| **TC\_DRONE\_01** | Điều phối Drone tự động | 1. Đơn hàng chuyển sang "Ready  2. Hệ thống tìm Drone rảnh gần nhất.  3. Gán đơn cho Drone. | Drone A (Pin 80%, Status: Idle) | Drone A chuyển trạng thái "Delivering". App hiện thông báo "Drone đang bay đến". | Hệ thống gán đúng Drone A, không gán Drone đang bận. | **Pass** |
| **TC\_DRONE\_02** | Drone hết pin/Pin yếu | 1. Giả lập Drone có pin < 20%.  2. Có đơn hàng mới cần giao. | Drone B (Pin 15%) | Hệ thống KHÔNG gán đơn cho Drone B. Tìm Drone khác hoặc báo "Chờ tài xế". | Hệ thống bỏ qua Drone B, gán cho Drone C (Pin 90%). | **Pass** |
| **TC\_DRONE\_03** | Drone cập nhật tọa độ Realtime | 1. Drone bay từ điểm A đến B.  2. Quan sát trên bản đồ khách hàng. | Tọa độ thay đổi mỗi 2s | Icon Drone di chuyển mượt mà, không bị giật lùi hoặc đứng yên. | Socket trả về tọa độ liên tục, hiển thị đúng lộ trình. | **Pass** |
| **TC\_DRONE\_04** | Drone hoàn thành nhiệm vụ | 1. Drone đến tọa độ khách hàng.  2. Xác nhận giao hàng. | - | Đơn hàng thành "Completed". Drone tự động bay về trạm sạc (Status: Returning). | Drone cập nhật trạng thái "Returning" và bay về kho. | **Pass** |

\*Kịch bản kiểm thử Quản lý đơn hàng (Dành cho Admin/Nhà hàng)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Tên kịch bản** | **Các bước thực hiện** | **Dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế** | **Trạng thái** |
| **TC\_ADM\_01** | Nhà hàng xác nhận đơn mới | 1. Đăng nhập Admin Nhà hàng.  2. Thấy đơn mới trạng thái "Pending".    3. Nhấn "Xác nhận đơn". | Order ID: #1001 | Trạng thái đơn chuyển sang "Preparing" (Đang chuẩn bị). Khách hàng nhận được thông báo. | Dashboard cập nhật trạng thái tức thời qua Socket. | **Pass** |
| **TC\_ADM\_02** | Hủy đơn do hết món | 1. Nhà hàng nhấn "Hủy đơn".  2. Chọn lý do "Hết nguyên liệu". | Order ID: #1002 | Đơn bị hủy. Tiền được hoàn lại ví khách (nếu đã thanh toán Online). | Trạng thái đơn đổi thành "Cancelled", hoàn tiền thành công. | **Fail** |
| **TC\_ADM\_03** | Thêm món ăn mới vào menu | 1. Admin vào "Quản lý món".  2. Nhập tên, giá, ảnh.  3. Nhấn "Lưu". | Món: "Cơm tấm"  Giá: 45k | Món mới xuất hiện trên App Khách hàng ngay lập tức. | Món ăn hiển thị đầy đủ hình ảnh và giá. | **Pass** |

**4.3.3. Thiết kế kịch bản kiểm thử phi chức năng (Non-functional Test Cases)**

**a. Kịch bản kiểm thử Hiệu năng (Performance - Load/Stress Testing)**

*Mục tiêu:* Đảm bảo hệ thống chịu được tải cao vào giờ cao điểm (bữa trưa/bữa tối) và tính năng theo dõi Drone (Realtime) hoạt động mượt mà.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Tên kịch bản** | **Công cụ / Môi trường** | **Các bước thực hiện (Steps)** | **Thông số kỳ vọng (Metrics)** | **Kết quả thực tế** | **Trạng thái** |
| **TC\_PERF\_01** | Kiểm thử chịu tải (Load Test) - Đặt hàng đồng thời | **JMeter** / **K6** | 1. Giả lập 500 người dùng truy cập cùng lúc (Concurrent users).    2. Thực hiện hành động: Tìm món -> Thêm giỏ -> Đặt hàng.    3. Duy trì trong 15 phút. | - Thời gian phản hồi (Response Time) < 2s.    - Tỷ lệ lỗi (Error Rate) < 1%. | Avg Response Time: 1.2s. Không có request bị timeout. | **Pass** |
| **TC\_PERF\_02** | Kiểm thử chịu tải - Tracking Drone Realtime | **Socket.io Client Tool** | 1. Giả lập 200 kết nối Socket cùng lúc vào kênh theo dõi đơn hàng.    2. Gửi tọa độ cập nhật liên tục mỗi 1s từ phía Server. | - Độ trễ (Latency) < 200ms.    - Không bị mất gói tin tọa độ. | Socket kết nối ổn định, icon Drone di chuyển mượt. | **Pass** |
| **TC\_PERF\_03** | Kiểm thử áp lực (Stress Test) - Tìm điểm gãy | **JMeter** | 1. Tăng dần lượng user từ 100 lên 5.000 trong vòng 5 phút.    2. Xác định điểm server bắt đầu quá tải (CPU > 90% hoặc Crash). | Xác định được ngưỡng chịu đựng của hệ thống. | Hệ thống bắt đầu chậm khi đạt 2.500 user đồng thời. Server phục hồi tự động sau khi giảm tải. | **Pass** |
| **TC\_PERF\_04** | Kiểm tra tốc độ tải trang (Page Load Speed) | **Google Lighthouse** | 1. Truy cập Trang chủ và Trang Chi tiết món ăn trên mạng 4G. | - Điểm Performance > 80.    - Thời gian hiển thị nội dung đầu tiên (FCP) < 1.5s. | Điểm: 85/100. FCP: 1.2s. | **Pass** |

**b. Kịch bản kiểm thử Bảo mật (Security Testing)**

*Mục tiêu:* Đảm bảo dữ liệu khách hàng (SĐT, Địa chỉ, Lịch sử đơn) được an toàn

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Tên kịch bản** | **Loại lỗ hổng** | **Các bước thực hiện** | **Kết quả mong đợi** | **Kết quả thực tế** | **Trạng thái** |
| **TC\_SEC\_01** | Chống SQL Injection tại form Đăng nhập | **SQL Injection** | 1. Nhập vào ô Email: ' OR '1'='1    2. Nhập Password bất kỳ.    3. Nhấn Đăng nhập. | Hệ thống báo lỗi "Sai thông tin" hoặc chặn ký tự đặc biệt. Không cho phép đăng nhập. | Hệ thống báo lỗi "Email không hợp lệ". | **Pass** |
| **TC\_SEC\_02** | Chống XSS tại phần Bình luận | **Cross-Site Scripting (XSS)** | 1. Tại ô Review món ăn, nhập script: <script>alert("Hacked")</script>    2. Gửi bình luận và tải lại trang. | Script không được thực thi (không hiện popup alert). Hiển thị dưới dạng văn bản thuần. | Script hiển thị dạng text bình thường. | **Pass** |
| **TC\_SEC\_03** | Kiểm tra quyền truy cập URL (IDOR) | **Broken Access Control** | 1. Đăng nhập User A.    2. Vào xem đơn hàng #100.    3. Đổi URL trên trình duyệt thành đơn #101 (của User B). | Hệ thống báo lỗi "403 Forbidden" hoặc "Bạn không có quyền xem đơn hàng này". | Chuyển hướng về trang chủ, báo lỗi 403. | **Pass** |
| **TC\_SEC\_04** | Bảo mật mật khẩu trong Database | **Sensitive Data Exposure** | 1. Truy cập trực tiếp vào Database (MongoDB/MySQL).    2. Kiểm tra bảng Users. | Cột password phải được mã hóa (MD5/Bcrypt), không hiển thị text rõ (plain text). | Mật khẩu lưu dạng chuỗi băm (Hash): $2b$10$w... | **Pass** |

**c. Kịch bản kiểm thử Giao diện và Trải nghiệm (UI/UX Testing)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Tên kịch bản** | **Môi trường** | **Các bước thực hiện** | **Kết quả mong đợi** | **Kết quả thực tế** | **Trạng thái** |
| **TC\_UI\_01** | Kiểm tra hiển thị trên Mobile (Responsive) | **iPhone 12 / Samsung S21** | 1. Mở web trên trình duyệt điện thoại.    2. Kiểm tra Menu, lưới danh sách món ăn. | - Menu thu gọn vào nút Hamburger.    - Món ăn xếp thành 1-2 cột (không bị tràn màn hình). | Giao diện tự động co giãn tốt, không bị vỡ khung. | **Pass** |
| **TC\_UI\_02** | Kiểm tra tính khả dụng của nút "Đặt hàng" | **Desktop & Mobile** | 1. Thêm món vào giỏ.    2. Kiểm tra vị trí và màu sắc nút "Thanh toán". | Nút nổi bật, dễ nhìn, luôn hiển thị (sticky) hoặc ở vị trí thuận tiện ngón tay cái (trên mobile). | Nút màu cam nổi bật, cố định ở dưới màn hình mobile. | **Pass** |
| **TC\_UI\_03** | Kiểm tra hiển thị hình ảnh món ăn | **Mạng chậm (3G)** | 1. Load danh sách món ăn khi mạng yếu.    2. Kiểm tra ảnh chưa tải xong. | Hiển thị ảnh thay thế (Placeholder/Skeleton loading) trong khi chờ ảnh thật tải xong. Không bị vỡ layout. | Có hiệu ứng Skeleton loading mượt mà. | **Pass** |
| **TC\_UI\_04** | Thông báo lỗi thân thiện (User Friendly) | **Mọi thiết bị** | 1. Cố tình nhập sai form hoặc ngắt mạng.    2. Quan sát thông báo. | Thông báo rõ ràng (VD: "Vui lòng kiểm tra kết nối") thay vì mã lỗi kỹ thuật (VD: "Error 500"). | Hiển thị Toast message góc phải: "Mất kết nối Internet". | **Pass** |

**4.4. Dữ liệu kiểm thử (Test Data Strategy)**

Để đảm bảo tính khách quan và bao phủ được mọi trường hợp nghiệp vụ của hệ thống FoodFast Delivery, dữ liệu kiểm thử được chia thành 2 nhóm chính: **Dữ liệu tĩnh (Master Data)** (danh mục, món ăn, cấu hình) và **Dữ liệu động (Transactional Data)** (đơn hàng, thanh toán, vị trí Drone).

Chiến lược chuẩn bị dữ liệu bao gồm:

1. **Dữ liệu hợp lệ (Positive Data):** Dùng để kiểm thử luồng nghiệp vụ chính (Happy Path).
2. **Dữ liệu không hợp lệ (Negative Data):** Dùng để kiểm thử khả năng bắt lỗi (Validation) và bảo mật.
3. **Dữ liệu biên (Boundary Data):** Dùng để kiểm thử các giới hạn tính toán (Giá tiền, Số lượng, Khoảng cách).

**4.4.1. Bộ dữ liệu Tài khoản người dùng (User Data)**

Hệ thống sử dụng các bộ tài khoản được tạo sẵn với các quyền hạn và trạng thái khác nhau:

| **Loại tài khoản** | **Mục đích kiểm thử** | **Dữ liệu mẫu (Credential)** | **Trạng thái** |
| --- | --- | --- | --- |
| **Khách hàng (Chuẩn)** | Test luồng đặt hàng thông thường. | customer\_01@email.com / 123456 | Active, Đã xác thực Email. |
| **Khách hàng (Mới)** | Test luồng người dùng mới (chưa có địa chỉ, chưa có đơn). | new\_user@email.com / 123456 | Active, Trống lịch sử. |
| **Khách hàng (Bị khóa)** | Test tính năng chặn đăng nhập. | banned\_user@email.com / 123456 | Locked/Banned. |
| **Admin Nhà hàng** | Test quy trình nhận đơn, quản lý menu. | merchant\_01@foodfast.com / admin123 | Active, Có quyền quản lý Shop A. |
| **Quản trị viên (Super Admin)** | Test toàn quyền hệ thống. | superadmin@foodfast.com / SysAdmin@ | Active, Full permission. |

**4.4.2. Bộ dữ liệu Sản phẩm & Thực đơn (Product Data)**

Dữ liệu món ăn cần đa dạng để test các logic tính toán giá tiền và hiển thị:

|  |  |  |
| --- | --- | --- |
| **Loại dữ liệu** | **Mô tả chi tiết** | **Mục đích kiểm thử** |
| **Món ăn đơn giản** | Cơm gà (35.000đ | Test luồng thêm vào giỏ nhanh. |
| **Món hết hàng** | Bún bò (Hết hàng - Out of Stock). | Test giao diện (nút Mua bị ẩn/mờ) và ngăn chặn hành vi hack request. |
| **Món có tên dài/kí tự lạ** | "Mỳ cay 7 cấp độ phiên bản đặc biệt siêu cấp vô địch..." | Test vỡ giao diện (UI Break) trên Mobile. |
| **Món giá trị cao/thấp** | Nước lọc (0đ) hoặc Tiệc combo (5.000.000đ). | Test định dạng tiền tệ và giới hạn thanh toán. |

**4.4.3. Bộ dữ liệu Drone & Vị trí (Drone & Location Data)**

Đây là phần dữ liệu đặc thù cho tính năng giao hàng bằng Drone:

| **Loại dữ liệu** | **Chi tiết dữ liệu** | **Mục đích kiểm thử** |
| --- | --- | --- |
| **Drone Sẵn sàng** | ID: DRONE\_01 | Pin: 100% | Trạng thái: Idle | Vị trí: Tại kho. | Test luồng điều phối đơn hàng thành công. |
| **Drone Pin yếu** | ID: DRONE\_LOW | Pin: 15% | Trạng thái: Idle. | Test logic bỏ qua Drone này, không gán đơn hàng. |
| **Drone Đang bận** | ID: DRONE\_BUSY | Trạng thái: Delivering. | Test logic xếp hàng chờ hoặc tìm Drone khác. |
| **Tọa độ Hợp lệ** | Lat: 10.762622, Lng: 106.660172 (Quận 5, HCM). | Test tính khoảng cách và hiển thị bản đồ. |
| **Tọa độ Ngoài vùng phục vụ** | Lat: 21.028511, Lng: 105.804817 (Hà Nội). | Test thông báo lỗi: "Địa chỉ này nằm ngoài phạm vi giao hàng". |

**4.4.4. Cơ chế làm sạch dữ liệu (Data Cleanup Strategy)**

Để đảm bảo các lần kiểm thử (Test Run) không ảnh hưởng lẫn nhau, nhóm sử dụng chiến lược:

* **Database Seeding:** Trước mỗi đợt test lớn, chạy script để đưa Database về trạng thái chuẩn (xóa các đơn hàng rác, reset tồn kho).
* **Hard Delete vs Soft Delete:**
  + Đối với *Test Automation*: Thực hiện xóa cứng (Hard Delete) tài khoản/đơn hàng test sau khi chạy xong.
  + Đối với *Manual Test*: Sử dụng cờ "Deleted" (Soft Delete) để giữ lại lịch sử tra cứu nếu cần.

**CHƯƠNG 5: KẾT QUẢ ĐẠT ĐƯỢC VÀ ĐÁNH GIÁ CHẤT LƯỢNG**

**5.1. Tổng hợp kết quả thực thi kiểm thử (Test Execution Summary)**

Dựa trên kế hoạch kiểm thử đã đề ra tại Chương 3 và các kịch bản thiết kế tại Chương 4, quá trình thực thi đã bao phủ các phân hệ: Khách hàng, Quản trị và Drone.

**5.1.1. Thống kê kết quả theo Test Case**

Tổng hợp dữ liệu từ Ma trận truy vết yêu cầu (RTM) và các bảng kết quả thực tế, tỷ lệ đạt được như sau:

*Bảng 5.1: Tổng hợp trạng thái Test Case*

| **Trạng thái (Status)** | **Số lượng** | **Tỷ lệ %** | **Ghi chú** |
| --- | --- | --- | --- |
| **Pass (Đạt)** | 28 | ~78% | Các chức năng Đăng ký, Đăng nhập, Tìm kiếm, Tracking Drone hoạt động ổn định. |
| **Fail (Lỗi)** | 7 | ~19% | Tập trung vào tính năng: Voucher, Tính phí ship, Hủy đơn hàng và Social Login. |
| **Pending (Chờ)** | 1 | ~3% | Chức năng "Gửi thông báo khi đơn gần đến" (TC\_NOTIFY\_01) chưa thực hiện xong5. |
| **Tổng cộng** | **36** | **100%** | (Số liệu dựa trên các bảng chi tiết đã liệt kê) |

**5.1.2. Thống kê kết quả theo Module chức năng**

Dữ liệu dưới đây được trích xuất từ các bảng kiểm thử chi tiết của từng phân hệ:

*Bảng 5.2: Kết quả kiểm thử theo phân hệ*

|  |  |  |  |
| --- | --- | --- | --- |
| **Phân hệ (Module)** | **Test Cases** | **Kết quả nổi bật** | **Nguồn dữ liệu** |
| **Xác thực (Auth)** | 5 TC | **Pass 3 / Fail 2**    (Lỗi: Đăng nhập Google/Facebook, Quên mật khẩu) | 666 |
| **Tìm kiếm (Search)** | 3 TC | **Pass 3 / Fail 0**    (Tìm kiếm và Lọc hoạt động tốt) | 7 |
| **Giỏ hàng (Cart)** | 4 TC | **Pass 2 / Fail 2**    (Lỗi: Áp dụng Voucher không trừ tiền) | 88 |
| **Thanh toán & Ship** | 4 TC | **Pass 3 / Fail 1**    (Lỗi: Tính phí ship hiển thị sai) | 99 |
| **Drone Delivery** | 4 TC | **Pass 4 / Fail 0**    (Điều phối và Tracking tốt) | 10 |
| **Admin/Nhà hàng** | 3 TC | **Pass 2 / Fail 1**    (Lỗi: Hủy đơn không hoàn tiền) | 11 |

**5.2. Báo cáo và Phân tích lỗi (Defect Report & Analysis)**

**5.2.1. Danh sách các lỗi điển hình phát hiện được**

Dựa trên cột "Kết quả thực tế" (Actual Result) từ các bảng thiết kế, nhóm đã ghi nhận các lỗi cụ thể sau:

*Bảng 5.3: Danh sách lỗi (Defect List)*

| **ID Lỗi** | **Tên lỗi & Mô tả** | **Test Case liên quan** | **Mức độ (Priority)** |
| --- | --- | --- | --- |
| **DEF\_01** | **Voucher hợp lệ không trừ tiền:** Khi nhập mã GIAM10K, hệ thống xác nhận nhưng tổng tiền thanh toán không thay đổi1212. | TC\_ORD\_01 | **Fail** |
| **DEF\_02** | **Voucher sai điều kiện vẫn áp dụng:** Mã GIAM50K (yêu cầu đơn 200k) vẫn áp dụng được cho đơn 50k, hệ thống không chặn1313. | TC\_ORD\_02 | **Fail** |
| **DEF\_03** | **Sai phí vận chuyển:** Phí ship hiển thị cố định hoặc sai công thức, không khớp với API Google Maps (ví dụ 5km phải là 25k)1414. | TC\_PAY\_01 | **Fail** |
| **DEF\_04** | **Lỗi hoàn tiền khi hủy đơn:** Nhà hàng hủy đơn (lý do hết món) nhưng ví tiền khách hàng không được cộng lại15. | TC\_ADM\_02 | **Fail** |
| **DEF\_05** | **Lỗi đăng nhập Mạng xã hội:** Tính năng Login Google/Facebook chưa hoạt động16. | TC\_AUTH\_05 | **Fail** |

**5.2.2. Phân tích lỗi theo mức độ nghiêm trọng**

Dựa vào cột "Mức độ ưu tiên" trong Ma trận RTM, ta phân tích các lỗi tồn đọng:

* **Lỗi Critical (Nghiêm trọng):**
  + **REQ\_SHIP\_01 (Tính phí vận chuyển):** Đây là lỗi nghiêm trọng nhất vì ảnh hưởng trực tiếp đến doanh thu và quyết định thanh toán của khách hàng.
* **Lỗi High (Cao):**
  + **REQ\_AUTH\_04 (Quên mật khẩu):** Ảnh hưởng đến khả năng truy cập lại tài khoản của người dùng.
* **Lỗi Medium (Trung bình):**
  + **REQ\_CART\_03 (Voucher):** Tính năng khuyến mãi bị lỗi logic nhưng luồng đặt hàng cơ bản vẫn chạy được.

**5.2.3. Đánh giá nguyên nhân (Root Cause)**

* **Lỗi Voucher (TC\_ORD\_01, 02):** Do logic xử lý backend chưa kiểm tra kỹ điều kiện biên của giá trị đơn hàng.
* **Lỗi Phí Ship (TC\_PAY\_01):** Có thể do tích hợp API Google Maps chưa nhận được tham số khoảng cách chính xác hoặc logic tính toán giá tiền nhân với số km bị sai.
* **Lỗi Hủy đơn (TC\_ADM\_02):** Module hoàn tiền (Refund) chưa được kích hoạt tự động khi trạng thái đơn chuyển sang "Cancelled".

**5.3. Đánh giá chung về chất lượng phần mềm**

**5.3.1. Các điểm đạt được**

* **Hiệu năng:** Hệ thống chịu tải tốt với 500 user đồng thời, thời gian phản hồi trung bình 1.2s.
* **Drone Delivery:** Tính năng phức tạp nhất là điều phối và tracking Drone hoạt động rất mượt mà, tọa độ cập nhật realtime qua Socket.
* **Bảo mật:** Đã ngăn chặn thành công các lỗi SQL Injection và XSS.

**5.3.2. Các điểm hạn chế và hướng khắc phục**

Mặc dù các luồng chính (Happy Path) đều ổn, nhưng hệ thống logic tính toán tiền (Voucher, Ship) đang gặp nhiều lỗi.

* **Khắc phục ngay:** Cần sửa gấp logic tính phí ship (TC\_PAY\_01) và Voucher (TC\_ORD\_01) vì đây là điều kiện tiên quyết để nghiệm thu sản phẩm.
* **Cải thiện sau:** Tính năng thông báo (TC\_NOTIFY\_01) và Đăng nhập mạng xã hội có thể hoàn thiện ở giai đoạn sau.