**ĐẠI HỌC PHENIKAA**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN**

A logo for a university

AI-generated content may be incorrect.

**HỌC PHẦN: PHÂN TÍCH VÀ THIẾT KẾ PHẦN MỀM**

**ĐỀ TÀI:**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn | : ThS.Vũ Quang Dũng  TS.Trịnh Thanh Bình |
| Nhóm 5 | : Nguyễn Thanh Phong - 22010251 |
|  | Trần Mạnh Hoàng - 23010290 |
|  | Đỗ Minh Thanh - 23012921 |
|  | Hà Đức Thắng - 23010284 |
|  | Lê Anh Minh - 23010228 |
| Lớp tín chỉ | * : [CSE703048-1-1-25(N06)](https://canvas.phenikaa-uni.edu.vn/courses/19865) |

**Hà Nội, tháng 10 năm 2025**

**📑 MỤC LỤC HOÀN CHỈNH — *Project Plan: Kế hoạch phát triển hệ thống Đặt Bánh Ngọt Online***

**1. GIỚI THIỆU CHUNG  
1.1. Mục tiêu dự án  
1.2. Phạm vi và đối tượng sử dụng  
1.3. Sản phẩm đầu ra mong đợi  
1.4. Lý do chọn đề tài**

**2. CƠ CẤU TỔ CHỨC NHÓM  
2.1. Danh sách thành viên và vai trò  
2.2. RACI Matrix (Phân công trách nhiệm)**

**3. PHẠM VI DỰ ÁN (PROJECT SCOPE)  
3.1. Phạm vi chức năng chính (MVP)  
3.2. Giới hạn phạm vi (Out of scope)  
3.3. Các giả định và ràng buộc**

**4. KẾ HOẠCH THỰC HIỆN (PROJECT SCHEDULE)  
4.1. Mô hình phát triển được chọn (Agile Lite / 4 Sprint)  
4.2. Kế hoạch chi tiết theo Sprint  
4.3. Biểu đồ Gantt (Timeline tổng thể)  
4.4. Lý do lựa chọn mô hình Agile và so sánh với các mô hình khác**

**5. PHÂN RÃ CÔNG VIỆC (WORK BREAKDOWN STRUCTURE – WBS)  
5.1. Danh sách các giai đoạn & nhiệm vụ  
5.2. Liên kết giữa các task chính (dependency)**

**6. NGUỒN LỰC VÀ CÔNG CỤ SỬ DỤNG  
6.1. Công cụ lập trình, thiết kế, kiểm thử  
6.2. Môi trường phát triển và triển khai  
6.3. Tài nguyên nhóm & hạ tầng**

**7. QUẢN LÝ RỦI RO (RISK MANAGEMENT PLAN)  
7.1. Danh sách rủi ro tiềm ẩn  
7.2. Đánh giá mức độ ảnh hưởng (Impact/Probability)  
7.3. Biện pháp giảm thiểu và kế hoạch dự phòng**

**8. KẾ HOẠCH KIỂM THỬ (TEST PLAN OVERVIEW)  
8.1. Chiến lược kiểm thử  
8.2. Tiêu chí pass/fail và công cụ test**

**9. KẾ HOẠCH QUẢN LÝ THAY ĐỔI VÀ BÁO CÁO TIẾN ĐỘ  
9.1. Cách xử lý thay đổi yêu cầu  
9.2. Cập nhật tiến độ & báo cáo hàng tuần**

**10. KẾ HOẠCH BÀN GIAO & ĐÁNH GIÁ KẾT QUẢ  
10.1. Các mốc kiểm duyệt (Milestones)  
10.2. Tiêu chí hoàn thành dự án**

**1. GIỚI THIỆU CHUNG**

**1.1. Mục tiêu dự án**

Dự án **“PHÁT TRIỂN WEBSITE ĐẶT BÁNH NGỌT ONLINE THEO KIẾN TRÚC MICROSERVICES”** được triển khai nhằm xây dựng một hệ thống web thương mại điện tử đơn giản nhưng vận hành đầy đủ, hướng tới việc cung cấp giải pháp đặt bánh trực tuyến hiện đại, nhanh chóng và dễ mở rộng.

Mục tiêu tổng thể của dự án là tạo ra một ứng dụng web có khả năng đáp ứng toàn bộ quy trình đặt hàng thực tế – từ việc khách hàng xem danh mục sản phẩm, lựa chọn bánh, thêm vào giỏ hàng, thanh toán mô phỏng qua VNPay, đến khi hệ thống ghi nhận đơn hàng và quản trị viên xử lý trên giao diện quản trị.

Song song với việc xây dựng sản phẩm, nhóm hướng tới mục tiêu học thuật là áp dụng toàn bộ quy trình **phân tích – thiết kế – hiện thực – kiểm thử phần mềm** theo chuẩn môn học. Dự án giúp các thành viên rèn luyện kỹ năng làm việc nhóm, lập kế hoạch, chia nhỏ nhiệm vụ, sử dụng công nghệ mới và vận hành quy trình phát triển phần mềm một cách chuyên nghiệp như trong môi trường doanh nghiệp.

Thông qua dự án này, nhóm mong muốn tạo ra một sản phẩm không chỉ chạy được mà còn thể hiện rõ khả năng thiết kế hệ thống có tính ổn định, linh hoạt, dễ bảo trì, đồng thời củng cố tư duy kiến trúc phần mềm và tổ chức mã nguồn theo mô hình Microservices.

**1.2. Phạm vi và đối tượng sử dụng**

Hệ thống hướng đến hai nhóm người dùng chính: **khách hàng** và **quản trị viên cửa hàng**.

Đối với khách hàng, website cung cấp đầy đủ các chức năng cơ bản như đăng ký, đăng nhập, xem danh sách sản phẩm, tìm kiếm, lọc theo danh mục, xem chi tiết, thêm hoặc xóa sản phẩm khỏi giỏ hàng, tiến hành thanh toán và xem lại lịch sử đơn hàng. Giao diện người dùng được thiết kế thân thiện, dễ thao tác và phản hồi nhanh nhằm mang lại trải nghiệm mua sắm thuận tiện nhất.

Đối với quản trị viên, hệ thống cung cấp một trang quản trị độc lập giúp quản lý toàn bộ hoạt động của cửa hàng, bao gồm quản lý sản phẩm, danh mục, người dùng, đơn hàng và theo dõi thống kê doanh thu hoặc số lượng đơn mới.

Phạm vi triển khai của dự án được giới hạn trong môi trường học tập, vận hành cục bộ bằng Docker Compose với dữ liệu mẫu phục vụ cho việc trình diễn và kiểm thử. Tuy nhiên, toàn bộ thiết kế kiến trúc, luồng xử lý và mã nguồn đều đảm bảo khả năng mở rộng để có thể triển khai thực tế trong tương lai.

**1.3. Sản phẩm đầu ra mong đợi**

Kết quả cuối cùng của dự án bao gồm **ba nhóm sản phẩm chính**: mã nguồn hệ thống, tài liệu kỹ thuật và sản phẩm trình diễn.

Mã nguồn bao gồm toàn bộ phần **Frontend (React + TypeScript)**, **Backend (NestJS, PostgreSQL, Redis, RabbitMQ)**, **API Gateway (Nginx)** và **môi trường container hóa (Docker Compose)**, được tổ chức rõ ràng và có thể chạy hoàn chỉnh.

Tài liệu kỹ thuật đi kèm thể hiện đầy đủ các giai đoạn của quy trình phát triển phần mềm: **đặc tả yêu cầu (SRS)**, **mô hình hóa (Use Case, DFD, ERD, Sequence Diagram)**, **thiết kế kiến trúc tổng thể và chi tiết**, **test plan**, cùng **báo cáo cuối kỳ** mô tả kết quả thực hiện.

Ngoài ra, nhóm còn chuẩn bị **slide trình bày và video demo**, minh họa quy trình hoạt động của toàn bộ hệ thống từ góc nhìn người dùng và quản trị viên. Sản phẩm cuối cùng cần thể hiện rõ năng lực của nhóm trong việc vận dụng kiến thức môn học vào dự án thực tế, đồng thời đáp ứng các tiêu chí về tính ổn định, hiệu năng và khả năng mở rộng của một hệ thống web hiện đại.

**1.4. Lý do chọn đề tài**

Trong những năm gần đây, **mua sắm trực tuyến** đã trở thành thói quen phổ biến của người tiêu dùng, không chỉ với các mặt hàng công nghệ mà cả trong lĩnh vực thực phẩm và đồ ngọt. Tuy nhiên, hầu hết các cửa hàng bánh ngọt quy mô nhỏ tại Việt Nam vẫn bán hàng thông qua mạng xã hội, thiếu nền tảng quản lý đơn hàng và khách hàng hiệu quả. Điều này dẫn đến việc xử lý đơn chậm, dễ sai sót và khó theo dõi dữ liệu kinh doanh.

Từ thực tế đó, nhóm nhận thấy việc xây dựng một **hệ thống đặt bánh trực tuyến chuyên biệt** là cần thiết, vừa có ý nghĩa thực tiễn vừa giúp củng cố kiến thức môn học. Đề tài “**Nền tảng Đặt Bánh Ngọt Online**” được lựa chọn vì nó kết hợp hài hòa giữa yếu tố nghiệp vụ thân quen, dễ mô phỏng, với các yêu cầu kỹ thuật đủ phức tạp để thử nghiệm nhiều công nghệ hiện đại.

Đặc biệt, nhóm mong muốn áp dụng **kiến trúc Microservices** thay cho mô hình Monolithic truyền thống. Cách tiếp cận này cho phép chia nhỏ hệ thống thành nhiều dịch vụ độc lập, mỗi dịch vụ đảm nhiệm một chức năng cụ thể như người dùng, sản phẩm, đơn hàng hay thanh toán. Việc tách biệt giúp dễ dàng mở rộng, bảo trì và triển khai song song nhiều thành phần mà không ảnh hưởng đến toàn hệ thống.

Ngoài ra, đề tài còn giúp nhóm rèn luyện tư duy thiết kế hướng dịch vụ (Service-Oriented Design), làm quen với công cụ triển khai container hóa (Docker), hệ thống giao tiếp bất đồng bộ (RabbitMQ), và giải pháp giám sát hệ thống (Loki, Grafana). Đây đều là các công nghệ đang được ứng dụng rộng rãi trong các doanh nghiệp phát triển phần mềm hiện nay, giúp dự án vừa mang tính học thuật vừa có giá trị thực tế cao.

**2.1. Danh sách thành viên và vai trò**

**Nhóm 5 gồm 5 thành viên, mỗi người đảm nhiệm một số vai trò cụ thể trong toàn bộ vòng đời dự án, từ giai đoạn lập kế hoạch đến triển khai và báo cáo. Việc phân công được dựa trên thế mạnh cá nhân, kinh nghiệm lập trình, khả năng trình bày và tư duy logic của từng thành viên, đảm bảo toàn bộ quy trình phát triển diễn ra nhịp nhàng, không chồng chéo trách nhiệm.**

| **STT** | **Họ và tên** | **Mã SV** | **Vai trò chính** | **Nhiệm vụ chi tiết** |
| --- | --- | --- | --- | --- |
| **1** | **Trần Mạnh Hoàng** | 23010290 | **Trưởng nhóm – Kiến trúc hệ thống & Hạ tầng triển khai** | Chịu trách nhiệm lập kế hoạch tổng thể, quản lý tiến độ và điều phối các giai đoạn phát triển. Thiết kế kiến trúc tổng thể của hệ thống; cấu hình API Gateway (Nginx); thiết lập môi trường Docker Compose; tích hợp Redis, RabbitMQ, PostgreSQL; triển khai hệ thống logging tập trung (Loki, Promtail, Grafana) và đóng gói sản phẩm cuối cùng. |
| **2** | **Đỗ Minh Thanh** | 23012921 | **Phân tích hệ thống & Thiết kế nghiệp vụ** | Phụ trách phân tích yêu cầu nghiệp vụ, xây dựng mô hình Use Case và DFD; tham gia thiết kế kiến trúc tổng thể; đảm bảo sự nhất quán giữa tài liệu SRS và thiết kế hệ thống; rà soát logic luồng nghiệp vụ trước khi triển khai; hỗ trợ kiểm thử và đánh giá tài liệu thiết kế. |
| **3** | **Nguyễn Thanh Phong** | 22010251 | **Trưởng nhóm Backend & Thiết kế chi tiết** | Thiết kế chi tiết kiến trúc và cơ sở dữ liệu; xây dựng các mô-đun backend chính (User, Product, Cart, Order, Payment, Notification); thiết kế ERD và viết tài liệu mô tả API chi tiết; tích hợp Redis và RabbitMQ để tối ưu hiệu năng; đảm bảo tính ổn định và khả năng mở rộng của hệ thống. |
| **4** | **Hà Đức Thắng** | 23010284 | **Frontend khách hàng & Kiểm thử** | Phát triển giao diện người dùng cho phía khách hàng (Home, Product List/Detail, Cart, Checkout, Order History, Account); thiết kế luồng tương tác thân thiện và phản hồi nhanh; viết test case; thực hiện kiểm thử hộp đen (Postman/Newman); ghi nhận và xử lý lỗi; biên tập báo cáo cuối kỳ và slide trình bày. |
| **5** | **Lê Anh Minh** | 23010228 | **Frontend quản trị & Dữ liệu mẫu** | Xây dựng trang quản trị (Admin) bao gồm đăng nhập, quản lý người dùng, sản phẩm, danh mục, đơn hàng, và dashboard thống kê. Tạo dữ liệu mẫu (seed data) phục vụ kiểm thử; phối hợp kiểm tra tích hợp giữa frontend–backend; thực hiện báo cáo hàng tuần và hoàn thiện báo cáo cuối kỳ. |

Nhờ sự phân chia nhiệm vụ rõ ràng, nhóm có thể làm việc song song ở nhiều giai đoạn khác nhau, đảm bảo tiến độ ổn định và giảm thiểu rủi ro trong quá trình phát triển.

**2.2. Ma trận trách nhiệm (RACI Matrix)**

**Ký hiệu:**

* **R (Responsible)** – Người trực tiếp thực hiện công việc.
* **A (Accountable)** – Người chịu trách nhiệm chính, phê duyệt kết quả.
* **C (Consulted)** – Người được tham khảo, góp ý.
* **I (Informed)** – Người được thông báo về tiến độ hoặc kết quả.

| **Hoạt động / Nhiệm vụ chính** | **Trần Mạnh Hoàng** | **Đỗ Minh Thanh** | **Nguyễn Thanh Phong** | **Hà Đức Thắng** | **Lê Anh Minh** |
| --- | --- | --- | --- | --- | --- |
| **1. Lập kế hoạch dự án (Project Plan)** | **A/R** | C | C | I | I |
| **2. Phân tích yêu cầu & Đặc tả SRS** | C | **A/R** | C | I | I |
| **3. Thiết kế kiến trúc tổng thể hệ thống** | **A/R** | C | C | I | I |
| **4. Thiết kế chi tiết & Cơ sở dữ liệu (ERD, API)** | C | I | **A/R** | I | I |
| **5. Phát triển Backend (6 services, Redis, MQ)** | **A/R** | I | **A/R** | I | I |
| **6. Phát triển Frontend – Khách hàng (Customer UI)** | I | I | C | **A/R** | I |
| **7. Phát triển Frontend – Quản trị (Admin UI)** | I | I | I | C | **A/R** |
| **8. Thiết lập hạ tầng (Docker Compose, Nginx, Logging)** | **A/R** | I | C | I | I |
| **9. Kiểm thử & Xử lý lỗi (Black-box Test)** | I | I | C | **A/R** | **C** |
| **10. Báo cáo & Slide trình bày cuối kỳ** | **A** | **C** | C | **A/R** | **A/R** |

Ma trận này giúp đảm bảo mỗi công việc đều có người chịu trách nhiệm rõ ràng, đồng thời khuyến khích trao đổi thường xuyên giữa các thành viên. Các buổi họp nhóm ngắn (stand-up meeting) được tổ chức hàng tuần để cập nhật tiến độ, nhận phản hồi và điều chỉnh kế hoạch nếu cần thiết.

Nhờ đó, nhóm duy trì được sự minh bạch trong công việc, tránh trùng lặp nhiệm vụ, đồng thời phát huy thế mạnh cá nhân của từng thành viên để hoàn thành dự án một cách hiệu quả và chuyên nghiệp.

**3. PHẠM VI DỰ ÁN (PROJECT SCOPE)**

**3.1. Phạm vi chức năng chính (MVP)**

Dự án **“Phát triển Website Đặt Bánh Ngọt Online theo kiến trúc Microservices”** được thiết kế nhằm mô phỏng một hệ thống thương mại điện tử cho phép người dùng đặt bánh trực tuyến và cửa hàng quản trị hoạt động kinh doanh hiệu quả.  
Hệ thống được xây dựng theo mô hình **Microservices**, tách biệt các chức năng thành các dịch vụ độc lập: người dùng, sản phẩm, đơn hàng, thanh toán, giỏ hàng, thông báo và đánh giá.

Phạm vi chức năng được xác định dựa trên sơ đồ Use Case tổng quan dưới đây:

* **Các tác nhân chính**:
  + *User* (Khách hàng)
  + *Admin* (Quản trị viên)
  + *VNPay* (Cổng thanh toán mô phỏng)
  + *Mail Server* (Dịch vụ gửi email tự động)

Các chức năng chi tiết tương ứng được thể hiện trong bảng sau:

| **Tác nhân / Nhóm chức năng** | **Mô tả chi tiết** |
| --- | --- |
| **User (Khách hàng)** | Người dùng có thể đăng ký tài khoản, đăng nhập, chỉnh sửa thông tin cá nhân và quản lý tài khoản. Họ có thể xem danh mục bánh, tìm kiếm, lọc và sắp xếp sản phẩm theo nhu cầu. Khi chọn sản phẩm, người dùng có thể thêm vào giỏ hàng, tùy chỉnh loại bánh, sau đó tiến hành đặt hàng và thanh toán mô phỏng qua VNPay. Sau khi hoàn tất, hệ thống gửi email xác nhận đơn hàng thông qua Mail Server. Người dùng có thể theo dõi trạng thái đơn hàng, viết đánh giá và xem lại lịch sử mua hàng. |
| **Admin (Quản trị viên)** | Quản trị viên chịu trách nhiệm vận hành toàn bộ hệ thống, có quyền quản lý người dùng, sản phẩm, danh mục, đơn hàng và đánh giá. Ngoài ra, admin có thể theo dõi thống kê doanh thu, lượng đơn hàng mới, sản phẩm bán chạy, và phản hồi của khách hàng. Các thao tác quản trị được thực hiện thông qua giao diện riêng với quyền truy cập cao hơn. |
| **VNPay (Cổng thanh toán)** | Tiếp nhận yêu cầu thanh toán từ hệ thống, xử lý luồng xác nhận **Return URL** và **IPN** để phản hồi kết quả thanh toán cho dịch vụ đặt hàng. Đây là phần thanh toán mô phỏng, không liên kết cổng thực tế. |
| **Mail Server** | Nhận thông tin từ dịch vụ đặt hàng và gửi email xác nhận tới khách hàng khi đơn hàng được tạo thành công hoặc trạng thái đơn thay đổi. |

Tất cả các chức năng trên được đóng gói trong kiến trúc Microservices, kết nối thông qua **API Gateway (Nginx)** và giao tiếp bất đồng bộ bằng **RabbitMQ**, giúp hệ thống có khả năng mở rộng, dễ bảo trì và triển khai độc lập từng dịch vụ.

**3.2. Giới hạn phạm vi (Out of Scope)**

Một số tính năng nâng cao chưa được triển khai trong giai đoạn này do giới hạn thời gian và phạm vi môn học:

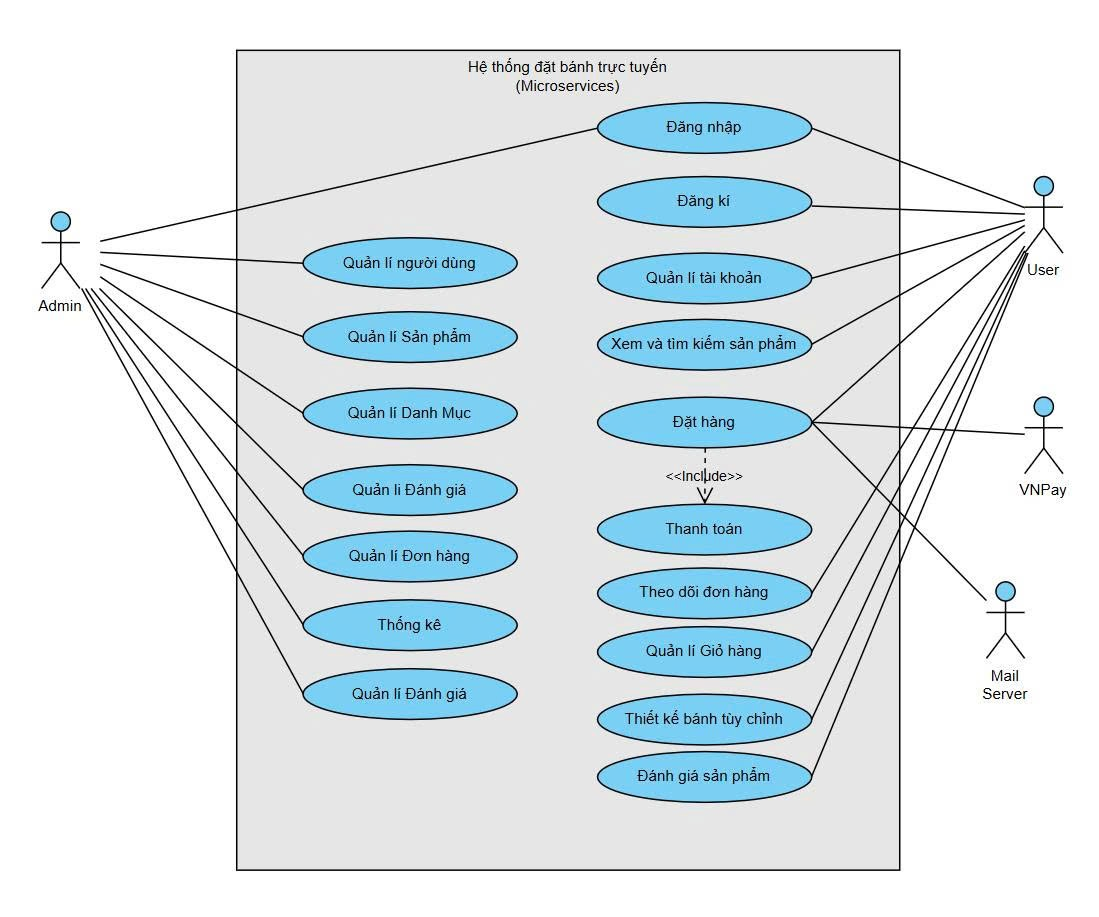
| **Hạng mục** | **Giải thích** |
| --- | --- |
| **1. Thanh toán thật qua VNPay hoặc MoMo** | Chỉ mô phỏng giao dịch, không kết nối với API thanh toán thực tế. |
| **2. Hệ thống giao hàng hoặc đối tác vận chuyển** | Không xử lý các nghiệp vụ vận chuyển, tính phí giao hàng hoặc theo dõi vị trí. |
| **3. Tính năng khuyến mãi, mã giảm giá, loyalty point** | Dự kiến bổ sung trong giai đoạn mở rộng. |
| **4. Ứng dụng di động (Mobile App)** | Hiện chỉ phát triển phiên bản web chạy trình duyệt. |
| **5. Báo cáo tài chính chi tiết & biểu đồ phân tích dữ liệu** | Dashboard chỉ hiển thị thống kê cơ bản. |
| **6. Cơ chế phân quyền phức tạp nhiều cấp** | Chỉ phân biệt hai nhóm quyền cơ bản: User và Admin. |

**3.3. Các giả định và ràng buộc**

| **Nhóm** | **Nội dung** |
| --- | --- |
| **Giả định 1** | Người dùng có thiết bị kết nối Internet và trình duyệt hỗ trợ JavaScript hiện đại. |
| **Giả định 2** | Các thành viên trong nhóm có thể cài đặt và sử dụng thành thạo Docker, PostgreSQL, Redis và RabbitMQ. |
| **Giả định 3** | Hệ thống hoạt động trên môi trường cục bộ (localhost), không yêu cầu hạ tầng đám mây. |
| **Giả định 4** | Dữ liệu ban đầu được tạo thủ công hoặc sinh bằng script seed phục vụ demo. |
| **Ràng buộc 1** | Thời gian thực hiện: 05 tuần – bao gồm toàn bộ các giai đoạn lập kế hoạch, phân tích, thiết kế, lập trình và kiểm thử. |
| **Ràng buộc 2** | Không sử dụng thư viện hoặc dịch vụ trả phí, toàn bộ công nghệ phải mã nguồn mở. |
| **Ràng buộc 3** | Tài nguyên phần cứng tối thiểu: CPU 4 nhân, RAM 16GB, dung lượng trống 20GB để chạy toàn bộ stack Docker Compose. |
| **Ràng buộc 4** | Hệ thống phải vận hành ổn định khi toàn bộ các service (6 Backend + FE + Gateway) cùng chạy song song. |

**3.4. Use Case tổng quan của hệ thống**

Hệ thống **“PHÁT TRIỂN WEBSITE ĐẶT BÁNH NGỌT ONLINE THEO KIẾN TRÚC MICROSERVICES”** được thiết kế nhằm hỗ trợ quá trình mua bán bánh trực tuyến giữa khách hàng và cửa hàng. Dưới đây là sơ đồ **Use Case tổng quan**, thể hiện toàn bộ các tác nhân và chức năng chính trong hệ thống:

**

**Mô tả sơ đồ:**

Sơ đồ thể hiện bốn tác nhân chính gồm **User**, **Admin**, **VNPay** và **Mail Server**, cùng các chức năng chính mà mỗi tác nhân thực hiện hoặc tương tác. Hệ thống vận hành theo mô hình **Microservices**, trong đó mỗi nhóm chức năng được triển khai như một dịch vụ riêng biệt, giao tiếp qua **API Gateway (Nginx)** và **RabbitMQ**.

* **User (Khách hàng)**: là người sử dụng cuối của hệ thống, có thể tạo tài khoản, đăng nhập, xem và tìm kiếm sản phẩm, thêm sản phẩm vào giỏ hàng, thiết kế bánh tùy chỉnh, đặt hàng, thanh toán, theo dõi đơn hàng và đánh giá sản phẩm.
* **Admin (Quản trị viên)**: có quyền quản trị toàn bộ hệ thống, bao gồm quản lý người dùng, sản phẩm, danh mục, đơn hàng, đánh giá và xem thống kê.
* **VNPay**: cổng thanh toán mô phỏng, được hệ thống gọi tới trong quá trình xử lý đơn hàng; sau khi giao dịch hoàn tất sẽ gửi phản hồi (IPN hoặc Return URL) để cập nhật trạng thái đơn hàng.
* **Mail Server**: dịch vụ gửi thông báo tự động đến người dùng, sử dụng trong các tình huống như xác nhận đơn hàng hoặc cập nhật trạng thái giao dịch.

Các chức năng của hệ thống được mô tả cụ thể trong bảng sau:

| **Tác nhân** | **Chức năng (Use Case)** | **Mô tả ngắn gọn** |
| --- | --- | --- |
| **User (Khách hàng)** | Đăng ký | Người dùng tạo tài khoản mới với thông tin cá nhân cơ bản. |
|  | Đăng nhập | Truy cập vào hệ thống để sử dụng các chức năng dành cho khách hàng. |
|  | Quản lý tài khoản | Cập nhật thông tin cá nhân, đổi mật khẩu hoặc xem lịch sử đặt hàng. |
|  | Xem và tìm kiếm sản phẩm | Duyệt danh mục bánh, tìm kiếm theo từ khóa, lọc theo loại, giá, sắp xếp kết quả. |
|  | Thiết kế bánh tùy chỉnh | Lựa chọn nguyên liệu, hương vị, kích thước và thiết kế mẫu bánh theo ý muốn. |
|  | Quản lý giỏ hàng | Thêm, xóa hoặc cập nhật số lượng sản phẩm trong giỏ hàng. |
|  | Đặt hàng | Tạo đơn hàng mới dựa trên các sản phẩm trong giỏ. |
|  | Thanh toán *(include trong Đặt hàng)* | Tiến hành thanh toán trực tuyến qua VNPay mô phỏng. |
|  | Theo dõi đơn hàng | Xem trạng thái các đơn hàng đã đặt, bao gồm “Đang xử lý”, “Hoàn tất” hoặc “Hủy”. |
|  | Đánh giá sản phẩm | Viết nhận xét và cho điểm sản phẩm đã mua. |
| **Admin (Quản trị viên)** | Quản lý người dùng | Thêm, sửa, xóa hoặc thay đổi quyền của tài khoản người dùng. |
|  | Quản lý sản phẩm | Thêm, chỉnh sửa hoặc xóa sản phẩm; cập nhật hình ảnh và thông tin chi tiết. |
|  | Quản lý danh mục | Tạo mới, đổi tên hoặc xóa danh mục bánh. |
|  | Quản lý đơn hàng | Xem danh sách đơn hàng, cập nhật trạng thái và theo dõi tiến độ xử lý. |
|  | Quản lý đánh giá | Duyệt, ẩn hoặc xóa các đánh giá vi phạm. |
|  | Thống kê | Xem các chỉ số tổng quan như doanh thu, số đơn hàng, sản phẩm bán chạy. |
| **VNPay** | Xử lý thanh toán | Nhận yêu cầu thanh toán, phản hồi kết quả giao dịch đến hệ thống. |
| **Mail Server** | Gửi thông báo | Gửi email xác nhận đơn hàng và trạng thái giao dịch cho khách hàng. |

**Nhận xét:**  
Sơ đồ Use Case cho thấy hệ thống bao quát đầy đủ các nghiệp vụ cốt lõi của một website thương mại điện tử, nhưng vẫn được tổ chức gọn gàng nhờ việc tách chức năng theo tác nhân. Mối quan hệ *«include»* giữa **Đặt hàng** và **Thanh toán** giúp thể hiện đúng quy trình nghiệp vụ, trong đó thanh toán chỉ được thực hiện khi người dùng hoàn tất đặt hàng.

Các Use Case được thiết kế bám sát phạm vi của hệ thống MVP, đảm bảo tính khả thi và khả năng mở rộng cho các giai đoạn phát triển sau này như tích hợp vận chuyển, khuyến mãi hoặc ứng dụng di động.

**4. KẾ HOẠCH THỰC HIỆN (PROJECT SCHEDULE – 10 TUẦN)**

**4.1. Mô hình và phương pháp áp dụng**

Nhóm áp dụng **mô hình Agile “Lite”** với **4 Sprint ngắn (2–2.5 tuần/Sprint)**.  
Mỗi Sprint bao gồm các hoạt động:

* **Planning** (lập kế hoạch Sprint, chia backlog theo story/feature)
* **Daily check-in** (báo cáo nhanh tiến độ nhóm)
* **Review** (demo kết quả đạt được)
* **Retrospective** (đánh giá và cải tiến quy trình)

Ngoài ra, nhóm vẫn đảm bảo các **artefacts (tài liệu)** theo yêu cầu mô hình **V-Model**:

| **Pha (V-Model)** | **Artefact trong Agile** | **Sprint tương ứng** |
| --- | --- | --- |
| Phân tích yêu cầu | SRS, Use Case, Scope | Sprint 1 |
| Thiết kế hệ thống | Kiến trúc, ERD, API Spec | Sprint 2 |
| Cài đặt | Source code backend, frontend | Sprint 3 |
| Kiểm thử | Test plan, Test evidence, báo cáo | Sprint 4 |

**4.2. Tổng quan thời gian**

* **Tổng thời lượng:** 10 tuần
* **Số Sprint:** 4 Sprint (mỗi Sprint ~2–2.5 tuần)
* **Nhóm họp chính:** 2–3 lần/tuần (qua Discord/Zalo)
* **Tổng thời gian dự án:** 01/09 – 10/11 (dự kiến)

**Bảng phân bổ thời gian tổng quát**

| **Giai đoạn / Sprint** | **Thời gian (tuần)** | **Mục tiêu chính** |
| --- | --- | --- |
| **Sprint 1 – Khởi tạo & Phân tích yêu cầu** | Tuần 1–2 | Hoàn thiện SRS, Use Case, phạm vi và kế hoạch dự án |
| **Sprint 2 – Thiết kế & Cấu hình kiến trúc** | Tuần 3–4 | Xây dựng kiến trúc hệ thống, ERD, API spec, môi trường dev |
| **Sprint 3 – Phát triển & Tích hợp tính năng chính** | Tuần 5–7 | Phát triển backend, frontend user, Redis/MQ, test hộp đen đợt 1 |
| **Sprint 4 – Hoàn thiện, Kiểm thử & Báo cáo** | Tuần 8–10 | Hoàn thiện admin FE, test tổng thể, logging stack, slide & báo cáo |

**4.3. Artefacts theo từng Sprint**

| **Sprint** | **Artefacts chính** |
| --- | --- |
| **Sprint 1** | Project Plan, Scope, Use Case/DFD khung, SRS v1.0 |
| **Sprint 2** | Kiến trúc tổng thể, API Spec, ERD chi tiết, Docker Compose skeleton |
| **Sprint 3** | Coding services + Frontend người dùng, Redis/MQ tích hợp, test hộp đen đợt 1 |
| **Sprint 4** | Frontend Admin, test tích hợp, logging & báo cáo, slide demo cuối kỳ |

**4.4. Chi tiết từng Sprint**

**Sprint 1 – Khởi tạo & Phân tích yêu cầu (Tuần 1–2)**

**Mục tiêu:**  
Xác định toàn bộ yêu cầu hệ thống, phạm vi, tác nhân, chức năng chính, mô hình dữ liệu sơ bộ và lập kế hoạch tổng thể.

**Công việc chính:**

1. Thu thập yêu cầu và phân tích nghiệp vụ (đặt bánh, thanh toán, quản lý, đánh giá).
2. Xây dựng **SRS (Software Requirement Specification)** bản 1.0.
3. Thiết kế **Use Case tổng quan & Use Case chi tiết**.
4. Xây dựng **DFD mức 0–1**, **phạm vi dự án**, và **Project Plan**.
5. Thiết lập môi trường làm việc (GitHub, Docker, Trello, cấu hình repo).

**Deliverables:**

* Tài liệu SRS v1.0
* Use Case diagram + mô tả chi tiết
* DFD sơ bộ
* Phạm vi dự án (Scope)
* Project Plan hoàn chỉnh

**Mốc hoàn thành:** Cuối tuần 2 (M1)  
**Rủi ro:** Thiếu đồng thuận về yêu cầu → Giải pháp: họp nhóm tổng hợp yêu cầu cuối Sprint.

**Sprint 2 – Thiết kế & Cấu hình kiến trúc (Tuần 3–4)**

**Mục tiêu:**  
Hoàn thiện thiết kế hệ thống tổng thể, kiến trúc microservices, API, CSDL, và môi trường phát triển ban đầu.

**Công việc chính:**

1. Thiết kế **Architecture Diagram** (Frontend – Gateway – Backend – Database – MQ – Cache).
2. Thiết kế **ERD chi tiết** (PostgreSQL) và tài liệu bảng dữ liệu.
3. Xây dựng **API Spec (OpenAPI / Swagger)** cho các service chính:
   * User Service
   * Product Service
   * Cart & Order Service
   * Payment (VNPay Mock)
   * Review Service
4. Tạo **Docker Compose skeleton**: PostgreSQL, Redis, RabbitMQ, MailServer, Nginx.
5. Kiểm thử môi trường – chạy được stack cơ bản.

**Deliverables:**

* Architecture Document
* API Specification
* ERD + bảng mô tả thuộc tính
* Docker Compose hoạt động
* Prototype UI (trang chủ, login, danh mục sản phẩm)

**Mốc hoàn thành:** Cuối tuần 4 (M2)  
**Rủi ro:** Kết nối giữa service lỗi → giải pháp: viết file .env.example và test container riêng lẻ.

**Sprint 3 – Phát triển & Tích hợp tính năng chính (Tuần 5–7)**

**Mục tiêu:**  
Triển khai toàn bộ các chức năng lõi: đăng nhập, giỏ hàng, đặt hàng, thanh toán, và đánh giá.

**Công việc chính:**

1. **Backend:**
   * Implement các service (User, Product, Cart, Order, Payment).
   * Tích hợp **Redis cache** (lưu giỏ hàng) và **RabbitMQ** (event xử lý đơn hàng & mail).
2. **Frontend (User):**
   * Trang sản phẩm, chi tiết, giỏ hàng, thanh toán, theo dõi đơn hàng.
3. **Kiểm thử:**
   * Viết **unit test** và **black-box test** đợt 1 cho API chính.
4. **Tích hợp VNPay Mock** → xử lý IPN, cập nhật trạng thái đơn hàng.

**Deliverables:**

* Source code backend & frontend (User)
* Redis & MQ hoạt động
* Testcases cơ bản pass ≥ 80%
* Postman Collection minh chứng

**Mốc hoàn thành:** Cuối tuần 7 (M3)  
**Rủi ro:** Lỗi tích hợp API → Giải pháp: kiểm thử từng module độc lập và mock dữ liệu trước khi merge.

**Sprint 4 – Hoàn thiện, Kiểm thử & Báo cáo (Tuần 8–10)**

**Mục tiêu:**  
Hoàn thiện toàn bộ hệ thống, giao diện quản trị, logging stack, kiểm thử tổng thể và chuẩn bị báo cáo.

**Công việc chính:**

1. **Frontend (Admin):**
   * Dashboard quản trị, CRUD sản phẩm, đơn hàng, người dùng, thống kê doanh thu.
2. **Kiểm thử:**
   * Kiểm thử tích hợp (Integration) & hệ thống (System test).
   * Kiểm thử hồi quy (Regression) sau sửa lỗi.
3. **Logging & Monitoring:**
   * Cấu hình Loki / Promtail để thu thập log.
4. **Tài liệu & Báo cáo:**
   * Hoàn thiện báo cáo đồ án, slide trình bày và video demo (7–10 phút).
   * Đóng gói sản phẩm (release .zip chứa code, tài liệu, hướng dẫn chạy Docker).

**Deliverables:**

* FE Admin hoàn chỉnh
* Báo cáo & slide thuyết trình
* Video demo
* Release package (.zip)

**Mốc hoàn thành:** Cuối tuần 10 (M4 – Final Delivery)  
**Rủi ro:** Phát hiện lỗi muộn → Giải pháp: kiểm thử sớm và lập buglist xử lý trong 2 ngày cuối.

**4.5. Sprint Backlog chi tiết**

Dưới đây là bảng kế hoạch cụ thể của từng Sprint: nhiệm vụ, người phụ trách, thời gian dự kiến và sản phẩm bàn giao.

**Sprint 1 – Khởi tạo & Phân tích yêu cầu (Tuần 1–2)**

| **Hạng mục** | **Nhiệm vụ cụ thể** | **Phụ trách** | **Thời gian (ngày)** | **Sản phẩm bàn giao** |
| --- | --- | --- | --- | --- |
| Thu thập yêu cầu | Phỏng vấn, phân tích nghiệp vụ đặt bánh & thanh toán | *Nhóm toàn bộ* | 2 | Bản mô tả nghiệp vụ thô |
| Xác định phạm vi dự án | Phân loại User/Admin, chức năng chính, ngoài phạm vi | *Trưởng nhóm* | 1 | Bảng Scope |
| Viết tài liệu SRS | Mô tả chức năng, phi chức năng | *Thành viên 1 & 2* | 3 | SRS v1.0 |
| Thiết kế Use Case | Tổng quan + chi tiết từng tác nhân | *Thành viên 3* | 2 | Use Case Diagram |
| Thiết kế DFD | Luồng dữ liệu chính, mức 0–1 | *Thành viên 4* | 2 | DFD sơ bộ |
| Lập kế hoạch dự án | Timeline, phân chia sprint, rủi ro, artefacts | *Trưởng nhóm* | 2 | Project Plan (file này) |

**Deliverables Sprint 1:**  
SRS v1.0, Use Case Diagram, Scope, DFD, Project Plan hoàn chỉnh.  
**Mốc:** Cuối tuần 2 – M1

**Sprint 2 – Thiết kế & Kiến trúc hệ thống (Tuần 3–4)**

| **Hạng mục** | **Nhiệm vụ cụ thể** | **Phụ trách** | **Thời gian (ngày)** | **Sản phẩm bàn giao** |
| --- | --- | --- | --- | --- |
| Thiết kế kiến trúc hệ thống | Vẽ sơ đồ kiến trúc tổng thể (FE–Gateway–BE–DB–Cache–MQ) | *Trưởng nhóm* | 2 | Architecture Diagram |
| Thiết kế ERD | Mô hình quan hệ dữ liệu, mô tả bảng | *Thành viên 2* | 2 | ERD chi tiết |
| Thiết kế API | Xây dựng Swagger/OpenAPI cho User, Product, Order, Payment | *Thành viên 3* | 3 | API Spec |
| Xây dựng Docker Compose | PostgreSQL, Redis, RabbitMQ, Mail, Gateway | *Thành viên 4* | 3 | docker-compose.yml |
| Test môi trường dev | Kiểm tra chạy stack, ghi log lỗi | *Nhóm toàn bộ* | 2 | Hệ thống chạy ổn định |
| Thiết kế UI prototype | Trang chủ, đăng nhập, xem sản phẩm | *Thành viên 5* | 2 | Giao diện mẫu (Figma hoặc HTML) |

**Deliverables Sprint 2:**

Architecture Doc, ERD, API Spec, Docker Compose, Prototype UI.  
**Mốc:** Cuối tuần 4 – M2

**Sprint 3 – Phát triển & Tích hợp tính năng chính (Tuần 5–7)**

| **Hạng mục** | **Nhiệm vụ cụ thể** | **Phụ trách** | **Thời gian (ngày)** | **Sản phẩm bàn giao** |
| --- | --- | --- | --- | --- |
| Backend – User Service | Đăng ký, đăng nhập, quản lý tài khoản | *Thành viên 1* | 3 | API user hoạt động |
| Backend – Product Service | CRUD sản phẩm, tìm kiếm, phân loại | *Thành viên 2* | 3 | API sản phẩm |
| Backend – Order Service | Đặt hàng, lưu đơn, cập nhật trạng thái | *Thành viên 3* | 3 | API đơn hàng |
| Payment Integration | Kết nối VNPay mock, IPN callback | *Thành viên 4* | 3 | Thanh toán mô phỏng thành công |
| Redis + RabbitMQ | Cache giỏ hàng, hàng đợi gửi mail | *Thành viên 2 & 4* | 2 | Redis & MQ hoạt động |
| Frontend (User) | Trang sản phẩm, giỏ hàng, thanh toán | *Thành viên 5* | 5 | FE người dùng hoàn thiện |
| Kiểm thử đợt 1 | Unit test, black-box test 20 case | *Cả nhóm* | 2 | Test pass ≥ 80% |

**Deliverables Sprint 3:**  
 Backend + FE user hoạt động, Redis/MQ, test pass.  
**Mốc:** Cuối tuần 7 – M3

**Sprint 4 – Hoàn thiện, Kiểm thử & Báo cáo (Tuần 8–10)**

| **Hạng mục** | **Nhiệm vụ cụ thể** | **Phụ trách** | **Thời gian (ngày)** | **Sản phẩm bàn giao** |
| --- | --- | --- | --- | --- |
| Frontend (Admin) | Dashboard quản trị, CRUD người dùng, sản phẩm, đơn hàng | *Thành viên 5* | 5 | FE Admin hoàn chỉnh |
| Kiểm thử tích hợp | Toàn bộ flow đặt hàng – thanh toán – đánh giá | *Thành viên 1–3* | 3 | Integration Test pass |
| Logging & Monitoring | Cấu hình Loki / Promtail, theo dõi log container | *Thành viên 4* | 2 | Hệ thống log hoạt động |
| Báo cáo & Slide | Soạn báo cáo đồ án, slide trình bày, video demo | *Nhóm toàn bộ* | 3 | Báo cáo + Slide + Video |
| Kiểm thử cuối cùng | Regression + User Acceptance | *Cả nhóm* | 2 | Kết quả test pass ≥ 90% |
| Release & Nộp | Đóng gói file .zip (source + hướng dẫn + seed) | *Trưởng nhóm* | 1 | Bản nộp chính thức |

**Deliverables Sprint 4:**  
FE Admin, Test tổng thể, Logging, Báo cáo & Demo hoàn chỉnh.  
**Mốc:** Cuối tuần 10 – M4

**4.6. Biểu đồ tiến độ (Gantt chart – dạng bảng)**

| **Công việc / Tuần** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SRS, Use Case, DFD, Plan | X | X |  |  |  |  |  |  |  |  |
| Thiết kế hệ thống, ERD, API |  |  | X | X |  |  |  |  |  |  |
| Docker Compose, UI Prototype |  |  | X | X |  |  |  |  |  |  |
| Backend core + Redis/MQ |  |  |  |  | X | X |  |  |  |  |
| Frontend người dùng |  |  |  |  | X | X |  |  |  |  |
| Test đợt 1 |  |  |  |  |  | X |  |  |  |  |
| Frontend admin |  |  |  |  |  |  | X | X |  |  |
| Logging + Kiểm thử tổng thể |  |  |  |  |  |  | X | X |  |  |
| Báo cáo + Slide + Demo |  |  |  |  |  |  |  |  | X | X |

**4.7. Lý do chọn mô hình Agile (so với Waterfall, V-Model, Spiral)**

* **Agile “Lite”** linh hoạt, cho phép cập nhật backlog giữa các Sprint mà không làm gián đoạn toàn bộ tiến độ như Waterfall.
* Mỗi Sprint có sản phẩm chạy được (incremental delivery), giúp nhóm phát hiện và sửa lỗi sớm.
* **Phù hợp với microservices:** dễ chia việc cho nhiều thành viên, mỗi người phụ trách 1 service.
* **Kết hợp ưu điểm V-Model:** các tài liệu (SRS, thiết kế, kiểm thử) vẫn được chuẩn hóa để đáp ứng yêu cầu học phần.
* **Spiral** không được chọn vì phức tạp và mất thời gian phân tích rủi ro không cần thiết cho dự án học kỳ.

**4.8. Rủi ro và hướng xử lý**

| **Rủi ro** | **Ảnh hưởng** | **Biện pháp xử lý** |
| --- | --- | --- |
| Chậm tiến độ ở Sprint 2–3 | Ảnh hưởng phần phát triển chính | Ưu tiên feature cốt lõi (User, Product, Order), cắt bớt chức năng phụ |
| Lỗi kết nối RabbitMQ/Redis | Ảnh hưởng luồng đặt hàng | Thiết lập env mẫu + kiểm thử container riêng |
| Backend–Frontend không đồng bộ API | Giao diện lỗi | Cập nhật Swagger liên tục, họp sync API mỗi tuần |
| Thiếu thành viên do lịch học | Trễ deadline | Phân công chéo, ghép cặp hỗ trợ (pair programming) |
| Demo lỗi kỹ thuật | Mất điểm trình bày | Chuẩn bị video dự phòng & bản chạy local |

**4.9. Mốc bàn giao (Milestones)**

| **Mốc** | **Tuần** | **Nội dung bàn giao** |
| --- | --- | --- |
| **M1** | Tuần 2 | SRS v1.0, Use Case, Scope, Project Plan |
| **M2** | Tuần 4 | Architecture, API Spec, ERD, Docker Compose |
| **M3** | Tuần 7 | Backend + FE người dùng hoạt động |
| **M4** | Tuần 10 | FE Admin, test hoàn tất, báo cáo & demo |

**5. QUẢN LÝ DỰ ÁN (PROJECT MANAGEMENT)**

**5.1. Cơ cấu tổ chức nhóm (Team Structure)**

Dự án được thực hiện bởi **Nhóm 5 – Lớp PTTKPM25-26\_N06**, gồm 5 thành viên.  
Cơ cấu nhóm được tổ chức theo mô hình **Agile Scrum**, trong đó có **Product Owner (PO)**, **Scrum Master**, và **Development Team**.  
Mỗi thành viên chịu trách nhiệm một mảng chính, đồng thời hỗ trợ lẫn nhau trong suốt 10 tuần phát triển.

| **Họ và tên** | **Vai trò trong nhóm** | **Nhiệm vụ chính** |
| --- | --- | --- |
| **Trần Mạnh Hoàng** | *Scrum Master* | Quản lý tiến độ, tổng hợp báo cáo, điều phối các Sprint, cấu hình Gateway & Docker Compose, hỗ trợ Backend, triển khai Logging, đóng gói hồ sơ dự án. |
| **Nguyễn Thanh Phong** | *Product Owner / Backend Lead/Team Leader* | Thiết kế kiến trúc chi tiết (SOLO ), thiết kế chức năng (SOLO), xây dựng ERD (SOLO ), phát triển dịch vụ Backend chính, tích hợp Redis và RabbitMQ. |
| **Đỗ Minh Thanh** | *Business Analyst / Planner* | Lập kế hoạch dự án (Project Plan), viết SRS (đồng), Use Case & DFD, tham gia thiết kế kiến trúc (đồng), review API & giao diện FE. |
| **Hà Đức Thắng** | *Frontend Lead (User) / Tester* | Phụ trách FE giao diện khách hàng, viết báo cáo tuần, test hộp đen, hỗ trợ báo cáo và slide trình bày. |
| **Lê Anh Minh** | *Frontend Admin / Tester* | Xây dựng FE giao diện Admin, viết báo cáo tuần, test hộp đen, tạo seed data và hỗ trợ tổng hợp báo cáo. |

**5.2. Phân công công việc chi tiết**

| **Mảng công việc** | **Thành viên phụ trách** | **Nội dung cụ thể** |
| --- | --- | --- |
| **Phân tích & Lập kế hoạch** | Hoàng, Thanh | Viết SRS, lập kế hoạch chi tiết, xây dựng Use Case, DFD, và phạm vi hệ thống. |
| **Thiết kế kiến trúc hệ thống** | Phong (chính), Hoàng, Thanh | Thiết kế tổng thể microservices, vẽ ERD, xác định API, định nghĩa dữ liệu và luồng xử lý. |
| **Frontend – Khách hàng (User)** | Thắng | Xây dựng giao diện người dùng, giỏ hàng, đặt bánh, thanh toán, đánh giá sản phẩm. |
| **Frontend – Quản trị (Admin)** | Minh | Xây dựng dashboard quản lý, CRUD sản phẩm, thống kê, seed data. |
| **Backend Services** | Phong (lead), Hoàng (support) | Phát triển API cho người dùng, sản phẩm, đơn hàng, thanh toán, đánh giá; triển khai Redis, RabbitMQ, logging. |
| **Triển khai & Kết nối hệ thống** | Hoàng | Thiết lập Docker Compose, Gateway, cấu hình container & môi trường chạy đồng bộ. |
| **Kiểm thử & Đánh giá** | Thắng, Minh | Viết test case, kiểm thử hộp đen, kiểm tra tích hợp dịch vụ, tổng hợp bug report. |
| **Báo cáo & Demo** | Cả nhóm | Viết báo cáo, làm slide, quay video demo và trình bày đồ án. |

**5.3. Quy trình phát triển (Development Process)**

Nhóm áp dụng mô hình **Agile Lite**, chia thành **4 Sprint** trong tổng thời gian **10 tuần**.  
Mỗi Sprint kéo dài khoảng 2–2.5 tuần, có đầy đủ các bước: *Planning – Development – Review – Retrospective*.

**Cấu trúc làm việc:**

* **Sprint Planning:** Xác định mục tiêu, backlog và phân công chi tiết.
* **Development:** Thành viên thực hiện task được giao, có daily check-in (2–3 lần/tuần).
* **Review:** Cuối mỗi Sprint, nhóm demo sản phẩm cho Product Owner và cập nhật backlog.
* **Retrospective:** Nhận xét kết quả, ghi nhận khó khăn và đề xuất cải tiến.

**Công cụ hỗ trợ:**  
GitHub (code), Trello (quản lý tiến độ), Docker (triển khai), Postman (test API), Google Docs/Slides (báo cáo), Discord & Zalo (trao đổi nội bộ).

**5.4. Quản lý tiến độ & kiểm soát công việc**

Nhóm quản lý công việc qua **Trello Board** với 3 cột chính: *To Do – Doing – Done*.  
Mỗi task có người chịu trách nhiệm, deadline cụ thể, và được đánh giá theo “Definition of Done (DoD)” trước khi chuyển sang hoàn tất.

**Tiêu chí “Definition of Done” (DoD):**

1. Code merge thành công, không lỗi.
2. Được test pass tại local và trong Docker.
3. Có log ổn định, không lỗi terminal.
4. Được review bởi Scrum Master hoặc PO.

**Báo cáo tiến độ định kỳ:**

* Họp nhóm online: tối Thứ 3 và Thứ 7.
* Họp review cuối sprint: có biên bản và checklist hoàn thành.
* Giảng viên hướng dẫn: báo cáo tuần qua form Google Sheet.

**5.5. Quản lý rủi ro (Risk Management)**

| **Mã** | **Rủi ro** | **Ảnh hưởng** | **Mức độ** | **Giải pháp phòng ngừa** |
| --- | --- | --- | --- | --- |
| R1 | Thành viên bận học, trễ tiến độ | Giảm hiệu suất | Cao | Dự phòng 1 người hỗ trợ mỗi task chính |
| R2 | Lỗi kết nối Docker hoặc MQ | Gián đoạn hệ thống | Trung bình | Chuẩn hóa file .env, test lại Compose trước mỗi Sprint |
| R3 | FE–BE không đồng bộ API | Lỗi giao diện | Trung bình | Định nghĩa Swagger, review API định kỳ |
| R4 | Lỗi build cuối | Demo bị gián đoạn | Cao | Test môi trường sớm (tuần 8–9), backup bản stable |
| R5 | Thiếu tài liệu, log lỗi | Báo cáo thiếu sót | Thấp | Ghi log chuẩn, cập nhật báo cáo hằng tuần |

**6. NGUỒN LỰC VÀ CÔNG CỤ SỬ DỤNG**

**6.1. Công cụ lập trình, thiết kế và kiểm thử**

Nhóm sử dụng kết hợp nhiều công cụ phục vụ cho quá trình **phân tích, thiết kế, phát triển, kiểm thử và báo cáo**.  
Các công cụ được lựa chọn dựa trên **tính ổn định, phổ biến, dễ tích hợp trong môi trường microservices** và khả năng phối hợp làm việc nhóm.

| **Loại công cụ** | **Tên công cụ / Công nghệ** | **Mục đích sử dụng** |
| --- | --- | --- |
| **Ngôn ngữ lập trình** | **JavaScript (Node.js, ReactJS)** | - Node.js: xử lý backend, xây dựng API services.  - ReactJS: phát triển giao diện người dùng (User & Admin). |
| **Framework / Thư viện chính** | **ExpressJS, Axios, React Router, Bootstrap, Ant Design** | - ExpressJS: định nghĩa route, middleware và logic xử lý.  - Axios: giao tiếp giữa frontend và backend.  - React Router & Ant Design: hỗ trợ điều hướng và UI trực quan. |
| **Cơ sở dữ liệu** | **PostgreSQL, Redis** | - PostgreSQL: lưu trữ dữ liệu chính của hệ thống.  - Redis: lưu cache và hỗ trợ hàng đợi (queue). |
| **Hệ thống message** | **RabbitMQ** | Dùng cho giao tiếp bất đồng bộ giữa các microservices (đặt hàng, mail, thanh toán). |
| **Thiết kế & mô hình hóa** | **Draw.io, Figma, Visual Paradigm, Lucidchart** | - Vẽ Use Case, DFD, ERD, kiến trúc hệ thống.  - Thiết kế giao diện web và sơ đồ kiến trúc microservices. |
| **Triển khai & quản lý môi trường** | **Docker, Docker Compose** | - Chạy độc lập các service (user, order, product, payment, mail,...).  - Tích hợp nhanh qua file docker-compose.yml. |
| **Quản lý mã nguồn & tiến độ** | **GitHub, Trello** | - GitHub: lưu trữ source code, kiểm soát version.  - Trello: theo dõi backlog, tiến độ và nhiệm vụ theo sprint. |
| **Kiểm thử & giám sát** | **Postman, Swagger, Jest** | - Postman: kiểm thử API RESTful.  - Swagger: mô tả và xác minh tài liệu API.  - Jest: kiểm thử đơn vị backend. |
| **Giao tiếp & phối hợp nhóm** | **Discord, Zalo, Google Meet** | - Thảo luận, họp sprint, chia sẻ màn hình demo. |
| **Soạn thảo & báo cáo** | **Google Docs, Google Slides, Canva** | - Viết báo cáo, chuẩn bị slide trình bày và tài liệu nhóm. |

**Nhận xét:**  
Các công cụ trên được nhóm chọn vì tương thích tốt với mô hình microservices, hỗ trợ CI/CD nội bộ và dễ thao tác trong phạm vi học phần.

**6.2. Môi trường phát triển và triển khai**

Dự án được triển khai hoàn toàn trong **môi trường Docker** nhằm tách biệt các dịch vụ, đảm bảo tính độc lập và dễ mở rộng.  
Cấu hình được chuẩn hóa thông qua file **docker-compose.yml**, cho phép chạy toàn bộ hệ thống chỉ bằng một lệnh duy nhất.

**6.2.1. Môi trường phát triển (Development Environment)**

| **Thành phần** | **Cấu hình / Phiên bản** | **Mục đích** |
| --- | --- | --- |
| Hệ điều hành | Windows 10 / macOS / Ubuntu | Môi trường phát triển của từng thành viên |
| IDE | Visual Studio Code | Viết code, gỡ lỗi, quản lý dự án |
| Node.js | v18.x | Runtime cho backend services |
| PostgreSQL | v16 | Cơ sở dữ liệu chính |
| Redis | v7 | Cache và hỗ trợ hàng đợi |
| RabbitMQ | v3.12 | Giao tiếp bất đồng bộ |
| Docker Desktop | v4.x | Triển khai toàn bộ stack microservices |
| Git | v2.45+ | Quản lý mã nguồn |
| Trình duyệt | Chrome / Edge | Kiểm thử giao diện người dùng |

**6.2.2. Môi trường triển khai (Deployment Environment)**

| **Thành phần** | **Mô tả** |
| --- | --- |
| **Hạ tầng chạy thử** | Máy chủ cục bộ (Localhost / Docker Compose) |
| **Cấu trúc microservices** | Các service độc lập: User, Product, Order, Payment, Mail, API Gateway |
| **Cổng giao tiếp chính** | Nginx Gateway (định tuyến và cân bằng tải) |
| **Dữ liệu mẫu (Seed Data)** | Sinh tự động qua script trong Docker |
| **Kênh kiểm thử** | Postman Collection, Swagger UI, giao diện React |
| **Kết nối nội bộ** | Docker Network (bridge mode), dùng alias giữa services |

**Quy trình triển khai:**

1. Cài đặt Docker và tải source code từ GitHub.
2. Chạy lệnh docker-compose up -d.
3. Hệ thống tự khởi tạo các service: DB, Redis, RabbitMQ, API, FE, Mail, Gateway.
4. Kiểm tra log và truy cập giao diện tại http://localhost:3000.

**6.3. Tài nguyên nhóm & hạ tầng sử dụng**

**6.3.1. Tài nguyên nhân lực**

| **Nguồn lực** | **Chi tiết** |
| --- | --- |
| **Số lượng thành viên** | 5 sinh viên (lớp PTTKPM\_N06 – Nhóm 5) |
| **Thời gian làm việc** | 10 tuần (từ khi lập kế hoạch → triển khai → kiểm thử → báo cáo) |
| **Tổng giờ làm việc nhóm** | ≈ 350–400 giờ (ước lượng trung bình 8h/tuần/người) |
| **Phương thức làm việc** | Hybrid (online & offline), họp nhóm 2–3 lần/tuần |

**6.3.2. Tài nguyên phần cứng & hạ tầng**

| **Hạng mục** | **Chi tiết cấu hình / Hạ tầng** |
| --- | --- |
| **Máy cá nhân lập trình** | Laptop cấu hình tối thiểu: CPU i5, RAM 8GB, SSD 256GB |
| **Môi trường container hóa** | Docker Engine (Desktop / CLI) |
| **Lưu trữ mã nguồn & tài liệu** | GitHub repository riêng của nhóm |
| **Báo cáo & tài liệu** | Google Drive (Docs, Sheets, Slides, PDF) |
| **Truyền thông & họp nhóm** | Zalo / Discord / Google Meet |
| **Công cụ kiểm thử chung** | Postman, Swagger, Browser Developer Tools |

**7.1. Danh sách rủi ro tiềm ẩn**

Bảng dưới đây liệt kê các rủi ro chính mà nhóm có thể gặp trong quá trình thực hiện, được chia thành 4 nhóm: **(1) Con người, (2) Kỹ thuật, (3) Tiến độ, (4) Quản lý và hạ tầng.**

| **Mã rủi ro** | **Loại rủi ro** | **Mô tả chi tiết** | **Nguyên nhân chính** |
| --- | --- | --- | --- |
| **R1** | Nhân sự | Thành viên bận học, lịch thi hoặc nghỉ giữa chừng dẫn đến gián đoạn tiến độ. | Trùng lịch học phần, sức khỏe, hoặc thiếu phối hợp. |
| **R2** | Tiến độ | Ước lượng thời gian chưa chính xác, backlog tăng nhanh khiến Sprint trễ. | Phân tích yêu cầu ban đầu chưa đầy đủ hoặc có thay đổi giữa chừng. |
| **R3** | Kỹ thuật | Lỗi cấu hình môi trường Docker, Redis hoặc RabbitMQ gây xung đột dịch vụ. | Không đồng nhất hệ điều hành hoặc phiên bản phần mềm. |
| **R4** | Kỹ thuật | API giữa frontend – backend không đồng bộ (khác endpoint, payload). | Giao tiếp chưa thống nhất giữa team FE và BE. |
| **R5** | Kỹ thuật | Lỗi thanh toán VNPay mock hoặc giao tiếp bất đồng bộ qua MQ. | Dữ liệu truyền sai định dạng hoặc lỗi xác thực callback. |
| **R6** | Chất lượng | Không đủ thời gian test, còn bug tồn tại khi demo. | Thiếu test case hoặc kiểm thử muộn. |
| **R7** | Hạ tầng | Lỗi phần cứng cá nhân hoặc mất kết nối mạng khi chạy Docker Compose. | Máy yếu, thiếu RAM/SSD, hoặc lỗi Docker Engine. |
| **R8** | Tài liệu & báo cáo | Tài liệu chưa đồng nhất giữa các sprint, format khác nhau. | Soạn song song, thiếu người kiểm tra tổng hợp. |
| **R9** | Quản lý | Scrum Master hoặc Product Owner thiếu cập nhật backlog, gây hiểu nhầm nhiệm vụ. | Quá tải task hoặc quên cập nhật Trello. |
| **R10** | Bảo mật dữ liệu | Thông tin người dùng, token hoặc mật khẩu bị lưu trữ sai. | Không mã hóa hoặc thiếu validation khi đăng ký/đăng nhập. |

**7.2. Đánh giá mức độ ảnh hưởng (Impact/Probability)**

Mỗi rủi ro được đánh giá theo **hai tiêu chí**:

* **Xác suất xảy ra (Probability – P):** Mức độ có khả năng xảy ra (Thấp / Trung bình / Cao)
* **Mức độ tác động (Impact – I):** Mức độ ảnh hưởng nếu rủi ro xảy ra (Thấp / Trung bình / Cao)

Kết hợp hai yếu tố này, nhóm xác định **mức ưu tiên (Priority)** để đưa ra kế hoạch xử lý phù hợp.

| **Mã** | **Rủi ro** | **Xác suất (P)** | **Tác động (I)** | **Mức ưu tiên (Priority)** | **Nhận định** |
| --- | --- | --- | --- | --- | --- |
| **R1** | Thành viên bận, gián đoạn tiến độ | Trung bình | Cao | ⚠️ **Cao** | Có thể ảnh hưởng đến cả Sprint nếu không có người backup. |
| **R2** | Trễ Sprint do backlog thay đổi | Trung bình | Trung bình | ⚠️ **Trung bình** | Cần kiểm soát backlog và họp lại kế hoạch. |
| **R3** | Lỗi cấu hình Docker hoặc MQ | Trung bình | Cao | ⚠️ **Cao** | Là rủi ro kỹ thuật nghiêm trọng trong môi trường Microservices. |
| **R4** | Lỗi API giữa FE–BE | Cao | Trung bình | ⚠️ **Cao** | Ảnh hưởng trực tiếp đến giao diện và trải nghiệm người dùng. |
| **R5** | Lỗi tích hợp thanh toán | Thấp | Cao | ⚠️ **Trung bình** | Có thể xử lý bằng mock, nhưng cần test sớm. |
| **R6** | Bug khi demo, thiếu test | Trung bình | Cao | ⚠️ **Cao** | Thường xảy ra ở Sprint cuối, cần kiểm thử sớm. |
| **R7** | Lỗi phần cứng / mạng | Thấp | Trung bình | ⚠️ **Thấp** | Ảnh hưởng cục bộ, có thể khắc phục. |
| **R8** | Tài liệu không đồng nhất | Trung bình | Thấp | ⚠️ **Thấp** | Dễ phát hiện, có thể chỉnh sửa nhanh. |
| **R9** | Thiếu cập nhật backlog | Trung bình | Trung bình | ⚠️ **Trung bình** | Giảm hiệu suất làm việc nhóm. |
| **R10** | Lỗ hổng bảo mật | Thấp | Cao | ⚠️ **Cao** | Cần kiểm tra kỹ validation và lưu trữ dữ liệu. |

**Tổng quan đánh giá:**

* Rủi ro **cao nhất**: R1, R3, R4, R6, R10.
* Rủi ro **trung bình**: R2, R5, R9.
* Rủi ro **thấp**: R7, R8.

**7.3. Biện pháp giảm thiểu và kế hoạch dự phòng**

Nhóm xây dựng các **biện pháp phòng ngừa, hành động khi xảy ra, và kế hoạch dự phòng (Contingency Plan)** cho từng nhóm rủi ro.  
Điều này đảm bảo dù rủi ro có xảy ra, dự án vẫn được duy trì đúng tiến độ và chất lượng.

| **Mã** | **Rủi ro** | **Biện pháp phòng ngừa** | **Kế hoạch dự phòng / Hành động khi xảy ra** |
| --- | --- | --- | --- |
| **R1** | Thành viên bận hoặc nghỉ đột xuất | - Cập nhật tiến độ trên Trello hằng ngày. - Giao chéo task để có người backup. | - Scrum Master tạm thời phân bổ lại công việc. - Ưu tiên hoàn thiện chức năng cốt lõi trước. |
| **R2** | Trễ Sprint / backlog tăng | - Review backlog trước mỗi sprint. - Đặt giới hạn tối đa số lượng task. | - Giảm phạm vi sprint, chia tính năng lớn thành 2 phần nhỏ. |
| **R3** | Lỗi Docker / Redis / MQ | - Thống nhất version và file .env.  - Có script reset môi trường. | - Xóa cache, rebuild container, backup DB thủ công. - Tạo hướng dẫn cài đặt nhanh (Quick Setup). |
| **R4** | API giữa FE–BE không đồng bộ | - Ghi tài liệu Swagger. - Họp sync giữa FE–BE mỗi sprint. | - Fix ngay tại sprint hiện tại, commit bản vá riêng. |
| **R5** | Lỗi tích hợp thanh toán | - Sử dụng sandbox VNPay mock. - Test callback nhiều lần trước demo. | - Tắt tính năng VNPay tạm thời, chuyển sang COD (demo). |
| **R6** | Bug khi demo / thiếu test | - Viết test case sớm. - Chạy Postman collection mỗi sprint. | - Ưu tiên fix bug showstopper, giữ bản stable backup để demo. |
| **R7** | Lỗi máy cá nhân / mạng | - Lưu file Docker & code trên GitHub. - Mỗi thành viên có bản local. | - Chạy dự án trên máy khác hoặc Docker cloud. |
| **R8** | Tài liệu không đồng nhất | - Dùng chung template báo cáo Google Docs. - Một người kiểm tra format trước nộp. | - Rà soát và hợp nhất toàn bộ trước ngày nộp báo cáo. |
| **R9** | Thiếu cập nhật backlog | - Scrum Master nhắc nhở định kỳ. - Tự động hóa checklist trong Trello. | - Họp khẩn cập nhật lại tiến độ, tạm ngưng task phụ. |
| **R10** | Lỗ hổng bảo mật | - Kiểm tra input validation, hash mật khẩu (bcrypt). - Ẩn key môi trường. | - Tạo script reset dữ liệu test, kiểm tra token và log cảnh báo. |

**Tổng kết:**

* Các rủi ro có mức độ cao (R1, R3, R4, R6, R10) đều đã có **giải pháp cụ thể và kế hoạch ứng phó**.
* Nhóm áp dụng cơ chế **kiểm tra định kỳ mỗi sprint**, giúp phát hiện rủi ro sớm và giảm thiểu thiệt hại.
* Việc phân công nhiệm vụ chéo và sao lưu dữ liệu thường xuyên giúp dự án **luôn duy trì ổn định** dù có sự cố kỹ thuật hoặc nhân sự.

**8. KẾ HOẠCH KIỂM THỬ (TEST PLAN OVERVIEW)**

**8.1. Chiến lược kiểm thử**

**8.1.1. Mục tiêu kiểm thử**

Mục tiêu của kiểm thử là đảm bảo hệ thống **đạt yêu cầu chức năng và phi chức năng** đã định trong SRS, hoạt động ổn định end-to-end, an toàn ở mức cơ bản, và có thể demo chạy trên Docker Compose. Cụ thể:

* Xác thực tính đúng đắn của các API và UI cho các luồng chính: đăng ký/đăng nhập, duyệt sản phẩm, giỏ hàng, đặt hàng, thanh toán (VNPay mock), IPN, quản trị đơn hàng.
* Kiểm chứng tích hợp giữa services qua REST và RabbitMQ.
* Đảm bảo cache (Redis) hoạt động đúng cho giỏ hàng và cache sản phẩm.
* Phát hiện và loại trừ lỗi nghiêm trọng (showstopper) trước demo.
* Đảm bảo các NFR cơ bản: thời đáp ứng, bảo mật JWT, không rò rỉ dữ liệu.

**8.1.2. Phân loại kiểm thử (Test Types)**

Nhóm sẽ thực hiện các loại kiểm thử sau, sắp xếp theo thứ tự ưu tiên thực hiện trong chu kỳ dự án:

1. **Unit Test**
   * Phạm vi: hàm, service, module backend (NestJS/Node).
   * Mục tiêu: kiểm tra logic nội bộ, validate edge cases.
   * Công cụ: Jest.
2. **API Integration Test**
   * Phạm vi: endpoints REST, xác thực JWT, luồng trả về IPN.
   * Mục tiêu: kiểm tra contract API + DB, xử lý lỗi ngoại lệ.
   * Công cụ: Postman + Newman, Jest (integration).
3. **Component / UI Test (mức cơ bản)**
   * Phạm vi: hành vi giao diện quan trọng (form đăng nhập, giỏ hàng, checkout).
   * Mục tiêu: kiểm tra tương tác cơ bản, validation, routing.
   * Công cụ: Manual test trên trình duyệt (DevTools); (tuỳ điều kiện có thể dùng Playwright/RTL).
4. **End-to-End (E2E) Test**
   * Phạm vi: luồng “browsing → add-to-cart → checkout → VNPay → IPN → order updated”.
   * Mục tiêu: đảm bảo toàn bộ flow hoạt động liên tục qua gateway và MQ.
   * Công cụ: Postman Collection (simulate API flows) + Manual UI run.
5. **Black-box Test (User Acceptance Test)**
   * Phạm vi: các kịch bản theo yêu cầu môn học (login success/fail, checkout success/fail, admin update).
   * Mục tiêu: phù hợp với tiêu chí chấm; tạo evidence (screenshots, log).
   * Công cụ: Postman / Browser + ghi log.
6. **Regression Test**
   * Phạm vi: các chức năng đã pass sau mỗi lần sửa lỗi.
   * Mục tiêu: tránh lỗi tái xuất hiện, đảm bảo không phá vỡ chức năng cũ.
   * Công cụ: Newman chạy Postman collection.
7. **Non-functional Test (cơ bản)**
   * Phạm vi: response time cơ bản (< 2s cho trang chính), stress nhẹ (số lượng requests giới hạn).
   * Mục tiêu: đánh giá sơ bộ performance cho demo.
   * Công cụ: trực quan + browser tools; (khi cần dùng artillery/ab).
8. **Security Smoke Test (cơ bản)**
   * Phạm vi: verify JWT expiry, password hashing, input validation (SQLi/XSS simple).
   * Mục tiêu: không để lộ password plaintext, token leak.
   * Công cụ: manual checks.

**8.1.3. Phạm vi kiểm thử / Ngoại phạm vi**

**Phạm vi (In-scope):**

* Tất cả API public/private đã nêu trong SRS: Auth, User, Product, Cart, Order, Payment, Admin.
* Luồng thanh toán mô phỏng VNPay (Return + IPN).
* Redis (cart/cache) và RabbitMQ (event publish/consume).
* UI Customer & Admin (chức năng CRUD, dashboard).
* Logging stack (xác minh tồn tại logs, errors được ghi).

**Ngoại phạm vi (Out-of-scope):**

* Thanh toán thực qua cổng VNPay/Bank; tính phí vận chuyển thực tế; tích hợp SMS/3rd-party ngoài scope.
* Load test quy mô lớn, penetration test chuyên sâu.
* Mobile app native.

**8.1.4. Chiến lược test theo Sprint**

* **Sprint 1:** chuẩn bị test plan, test data, viết test cases mẫu cho Use Case; build Postman skeleton.
* **Sprint 2:** unit tests for API contracts (skeleton services), validate ERD → run smoke tests trên Docker skeleton.
* **Sprint 3:** tập trung Unit + Integration + E2E flows; black-box test đợt 1; tạo evidence (screenshots, logs).
* **Sprint 4:** regression, system test, UAT, non-functional checks, final test evidence chuẩn bị nộp.

**8.1.5. Test responsibilities (Vai trò)**

* **Test Lead / Coordinator:** Hà Đức Thắng — lập kế hoạch test, quản lý test cases, chạy black-box & tổng hợp evidence.
* **Tester / FE Admin support:** Lê Anh Minh — chạy test integration, kiểm thử Admin UI, seed data.
* **Backend devs:** Nguyễn Thanh Phong & Trần Mạnh Hoàng — viết unit tests, sửa bug, cung cấp mocks.
* **Business Analyst / PO:** Đỗ Minh Thanh — xác minh acceptance criteria, phê duyệt UAT.

**8.2. Tiêu chí pass/fail và công cụ test**

**8.2.1. Môi trường test & dữ liệu**

* **Môi trường:** Local Docker Compose, với các container: postgres, redis, rabbitmq, services, nginx, fe.
* **Endpoint base:** http://localhost:<port> (theo docker-compose).
* **Dữ liệu test:** seed script tạo 50 products, 6 categories, 3 users (1 admin, 2 customers).
* **Test accounts:**
  + admin@test.com / Admin123!
  + user1@test.com / User123!
* **Cách tạo dữ liệu bổ sung:** script npm run seed hoặc POST qua Admin API.

**8.2.2. Công cụ & artifact**

* **Jest** — unit test backend (coverage báo cáo cơ bản).
* **Postman / Newman** — collection cho API, dùng script pre-request/post-test; Newman để chạy CI-like.
* **Swagger (OpenAPI)** — kiểm tra contract API; làm reference cho test.
* **Browser (Chrome)** — manual regression, screenshots.
* **Grafana / Loki** — lấy logs (error traces) nếu cần.
* **GitHub Issues** — log defects; assign & triage.
* **Google Drive** — lưu test evidence (screenshots, Postman reports).

**8.2.3. Tiêu chí Pass / Fail (chi tiết)**

**Quy tắc chung:** Mỗi test case có kết quả PASS / FAIL / BLOCKED.

**Unit Test:**

* **Pass:** Các assertion đúng, coverage critical modules ≥ 60% (mục tiêu).
* **Fail:** Assertion thất bại → dev sửa.

**API Test (Integration):**

* **Pass nếu:**
  + HTTP status code đúng (200/201/204 cho success; 4xx/5xx theo spec cho lỗi).
  + Response schema khớp OpenAPI (các trường bắt buộc tồn tại).
  + DB state thay đổi như mong đợi (e.g., order row created).
* **Fail nếu:** status code khác mong đợi, schema thiếu trường, hoặc DB không update.

**E2E / Black-box:**

* **Pass nếu:** user có thể hoàn thành kịch bản không có lỗi nghiêm trọng (no uncaught exceptions), order status cập nhật đúng sau IPN, email xác nhận (có record gửi).
* **Fail nếu:** bất kỳ step chính (checkout, payment callback, order update) bị lỗi.

**Non-functional (perf/security):**

* **Pass:** trang chủ render < 2s trên môi trường local (tham chiếu), mật khẩu được hash (bcrypt), token expiring / refresh hoạt động.
* **Fail:** response quá chậm (nhiều lần), mật khẩu lưu plaintext, token leak trong logs.

**Acceptance (bằng mắt / demo):**

* **Pass:** demo end-to-end thành công, không có lỗi critical trong log.
* **Fail:** lỗi critical xuất hiện.

**8.2.4. Defect severity & priority**

* **Critical (P0):** showstopper — chức năng cốt lõi không hoạt động (checkout/DB down). Ưu tiên fix ngay trong 24h.
* **High (P1):** ảnh hưởng lớn nhưng không block demo toàn bộ (login bug, admin unable update order). Fix trong sprint current.
* **Medium (P2):** lỗi chức năng phụ, UX minor. Fix trong Sprint backlog.
* **Low (P3):** cosmetic, typo. Fix nếu còn thời gian.

**8.2.5. Test case mẫu (template & ví dụ)**

**Test case template (bắt buộc):**

* TC ID: e.g., TC-API-001
* Title: Mô tả ngắn
* Precondition: seed data, login token
* Steps: liệt kê từng bước rõ ràng
* Expected result: chính xác, measurable
* Actual result: (điền khi chạy)
* Status: PASS / FAIL / BLOCKED
* Evidence: screenshot / Postman run / log file
* Assignee: người chạy test
* Date: ngày chạy

**Ví dụ TC (Checkout E2E):**

* **TC ID:** TC-E2E-Checkout-001
* **Title:** Khách hàng hoàn tất checkout và nhận IPN → order trạng thái = PAID
* **Precondition:** user1 logged in; cart có 2 sản phẩm; VNPay mock sẵn sàng
* **Steps:**
  1. User mở /cart → click Checkout.
  2. Hệ thống tạo order (POST /api/orders).
  3. Redirect tới VNPay mock; simulate success (return URL).
  4. VNPay mock gửi IPN tới /api/payment/ipn.
  5. Kiểm tra /api/orders/{orderId} trả về status = “PAID”.
* **Expected:** HTTP 201 khi tạo order; after IPN, order.status == PAID; email confirmation được tạo (log).
* **Evidence:** screenshot order detail, Postman logs, service logs.
* **Pass criteria:** tất cả step không có lỗi, DB reflect đúng, evidence kèm theo.

**8.2.6. Chỉ số và báo cáo test**

* **Test coverage (unit):** % lines covered cho các module chính (target ≥ 60%).
* **Test execution rate:** #TC executed / #TC planned per sprint (mục tiêu ≥ 90% cho sprint cuối).
* **Defect density:** # defects / # testcases.
* **Mean Time to Fix (MTTFix):** thời gian trung bình fix defect critical.
* **Acceptance Rate:** % testcases pass ở UAT.

Báo cáo test được tạo theo template: Summary → Test environment → Testcases executed (pass/fail) → Defects list (severity) → Recommendations → Evidence links.

**8.3. Quy trình xử lý lỗi (Defect lifecycle)**

1. **Phát hiện lỗi:** Tester log bug trên GitHub Issue theo template (steps, evidence, severity).
2. **Triage:** Test Lead và PO phân loại severity & assign dev.
3. **Fixing:** Dev tạo branch hotfix, commit và mở PR.
4. **Code review:** PR được review bởi 1 reviewer; unit tests pass.
5. **Merge & Deploy:** Merge vào main / release; run docker-compose up --build.
6. **Regression check:** Tester chạy lại liên quan testcases, mark defect closed nếu pass.
7. **Postmortem (nếu critical):** document root cause và preventive action.

**8.4. Lịch kiểm thử chi tiết (mapping to 4 Sprint / 10 tuần)**

| **Sprint** | **Kiểm thử chính** | **Người phụ trách** | **Kết quả mong đợi** |
| --- | --- | --- | --- |
| Sprint 1 | Chuẩn bị Test Plan, Testcases mẫu, tạo seed data | Thắng (lead) | Postman skeleton, TC template |
| Sprint 2 | Unit tests cho skeleton services; smoke test trên Docker Compose | Phong + Hoàng | Unit tests pass basic; smoke OK |
| Sprint 3 | Integration tests, E2E checkout flows, black-box test đợt 1 | Thắng + Minh | E2E flow pass; defects critical fix |
| Sprint 4 | Regression, System test, UAT, non-functional checks, final evidence | Cả nhóm | UAT pass ≥ 90%, final report + screenshots |

**8.5. Exit criteria & Acceptance**

**Exit criteria (kết thúc kiểm thử hệ thống):**

* Tất cả testcases critical (P0/P1) được pass hoặc có workaround documented.
* Không còn defect Critical open.
* Postman report & screenshots cho tất cả kịch bản cần demo.
* Docker compose có bản release chạy ổn định (khởi động toàn bộ service healthy).

**Final acceptance:** Product Owner + Scrum Master nghiệm thu dựa trên checklist: feature list hoàn, test evidence đủ, demo end-to-end thành công.

**8.6. Ghi chú & best-practices**

* Luôn chạy npm run seed và docker-compose down && up --build trước khi test E2E để đảm bảo môi trường sạch.
* Sử dụng branch feature + PR cho mọi fix; tránh commit trực tiếp lên main.
* Lưu giữ mọi evidence (screenshots, Postman run reports) trong thư mục docs/test-evidence và drive nhóm.
* Chạy Newman nightly (hoặc trước demo) để đảm bảo không regress.

**9. KẾ HOẠCH QUẢN LÝ THAY ĐỔI VÀ BÁO CÁO TIẾN ĐỘ**

**9.1. Cách xử lý thay đổi yêu cầu (Change Management Process)**

Trong mô hình Agile, yêu cầu có thể thay đổi theo phản hồi từ Product Owner hoặc giảng viên hướng dẫn.  
Để kiểm soát các thay đổi mà không ảnh hưởng đến tiến độ, nhóm thiết lập **quy trình quản lý thay đổi (Change Control Process)** như sau:

**9.1.1. Quy trình xử lý thay đổi**

| **Bước** | **Mô tả hoạt động** | **Trách nhiệm** | **Kết quả đầu ra (Output)** |
| --- | --- | --- | --- |
| **1. Đề xuất thay đổi (Change Request)** | Thành viên hoặc PO phát hiện yêu cầu thay đổi, ghi nhận qua Trello hoặc GitHub Issue. | Người phát hiện / PO | Phiếu đề xuất (Change Request Form). |
| **2. Đánh giá tác động (Impact Analysis)** | Scrum Master và Backend Lead đánh giá ảnh hưởng tới thời gian, chi phí, và kiến trúc. | Hoàng, Phong | Báo cáo đánh giá rủi ro, effort ước lượng. |
| **3. Phê duyệt thay đổi (Approval)** | PO xem xét và ra quyết định chấp nhận, hoãn hoặc từ chối. | PO (Phong) | Quyết định cập nhật backlog. |
| **4. Cập nhật kế hoạch (Plan Update)** | Scrum Master cập nhật Sprint backlog, tài liệu SRS và lịch làm việc. | Hoàng, Thanh | Trello + SRS + Plan mới. |
| **5. Triển khai thay đổi (Implementation)** | Thành viên phụ trách code theo backlog mới, test lại các module liên quan. | Dev liên quan | Code, test & commit thay đổi. |
| **6. Kiểm tra & ghi nhận (Review & Record)** | Thực hiện regression test, lưu kết quả & log thay đổi trong file CHANGELOG.md. | Tester + PO | Báo cáo kiểm thử sau thay đổi. |

**Nguyên tắc:**

* Mọi thay đổi phải **ghi nhận bằng văn bản (issue, comment, commit message)**.
* Không được thay đổi trực tiếp trên môi trường main branch khi chưa được **Scrum Master** hoặc **PO** chấp thuận.
* Với thay đổi lớn (impact đến kiến trúc hoặc DB), nhóm sẽ tạo **Sprint phụ (Mini-sprint)** hoặc **hotfix branch**.

**9.2. Cập nhật tiến độ & báo cáo hàng tuần**

**9.2.1. Cơ chế cập nhật tiến độ**

Nhóm duy trì **báo cáo tiến độ định kỳ hàng tuần**, nhằm theo dõi tình trạng công việc, kiểm soát backlog và phát hiện sớm vấn đề.  
Các thông tin được tổng hợp qua Trello, GitHub và Google Sheets.

| **Hình thức báo cáo** | **Tần suất** | **Công cụ sử dụng** | **Nội dung cập nhật** |
| --- | --- | --- | --- |
| **Báo cáo nhóm nội bộ (Internal Update)** | 2–3 lần/tuần | Trello + Discord | Cập nhật trạng thái task, tiến độ (%) từng người, vấn đề phát sinh. |
| **Báo cáo tuần (Weekly Report)** | Mỗi tuần | Google Sheets | Tổng hợp task hoàn thành, task pending, vấn đề chính, kế hoạch tuần tới. |
| **Sprint Review** | Cuối mỗi Sprint | Google Meet / Slide | Demo kết quả, nhận phản hồi từ PO và GV hướng dẫn. |
| **Progress Summary** | Toàn dự án (tuần 10) | Word / Slide | Báo cáo tiến độ tổng hợp, biểu đồ burndown, đánh giá hiệu suất nhóm. |

**9.2.2. Cách ghi nhận và đo tiến độ**

* Sử dụng **Trello Board (To Do → Doing → Done)** để cập nhật trạng thái task.
* Scrum Master kiểm tra tiến độ 2 lần/tuần, đối chiếu với backlog.
* Mỗi task được gán **thời lượng dự kiến (Estimation)** và **% hoàn thành thực tế (Actual Progress)**.
* Báo cáo tiến độ được xuất sang Google Sheet, tính **tổng hiệu suất (Productivity Index)** theo công thức:

Hiệu suất = (Tổng task Done / Tổng task Planned) × 100%

**9.2.3. Biểu đồ tiến độ minh họa**

*(Có thể chèn biểu đồ burndown chart hoặc cột “Task Completed / Week” trong file Word)*

**10. KẾ HOẠCH BÀN GIAO & ĐÁNH GIÁ KẾT QUẢ**

**10.1. Các mốc kiểm duyệt (Milestones)**

Dự án được chia thành 4 Sprint chính trong 10 tuần. Mỗi Sprint có **một mốc kiểm duyệt (checkpoint)**, kèm nội dung và kết quả bàn giao rõ ràng.  
Bảng dưới thể hiện các **milestone cụ thể**:

| **Mốc thời gian (Tuần)** | **Tên Milestone / Giai đoạn** | **Kết quả bàn giao (Deliverables)** | **Người phụ trách** |
| --- | --- | --- | --- |
| **Tuần 1–2** | **M1 – Phân tích & Kế hoạch** | - Project Plan bản nháp - SRS v1.0, Use Case, DFD khung - Repository GitHub khởi tạo | Thanh, Hoàng |
| **Tuần 3–4** | **M2 – Thiết kế hệ thống** | - Kiến trúc tổng thể (Microservices diagram) - ERD, API Spec, Docker Compose skeleton | Phong, Hoàng |
| **Tuần 5–7** | **M3 – Phát triển & Kiểm thử giai đoạn 1** | - Backend services (User, Product, Order, Payment) - Frontend (User) - Unit & Integration Test đợt 1 | Phong, Thắng, Minh |
| **Tuần 8–9** | **M4 – Hoàn thiện & Đánh giá nội bộ** | - FE Admin + Dashboard - E2E Test, Regression, Logging - Báo cáo & Slide bản gần cuối | Minh, Thắng |
| **Tuần 10** | **M5 – Bàn giao & Demo chính thức** | - Bản chạy Docker hoàn chỉnh - Báo cáo & Slide cuối cùng - Demo sản phẩm & biên bản nghiệm thu | Cả nhóm |

**Nhận xét:**  
Tất cả các mốc đều gắn với **Sprint Review**, giúp nhóm kiểm tra chất lượng sản phẩm định kỳ và đảm bảo không dồn lỗi vào cuối kỳ.

**10.2. Tiêu chí hoàn thành dự án (Completion Criteria)**

Để được xem là **hoàn thành dự án**, nhóm 5 phải đáp ứng đồng thời các tiêu chí **kỹ thuật, quy trình và tài liệu** sau:

| **Nhóm tiêu chí** | **Tiêu chí chi tiết** |
| --- | --- |
| **Kỹ thuật (Technical)** | - Tất cả dịch vụ microservice hoạt động ổn định trên Docker Compose. - FE (User/Admin) kết nối thành công tới API Gateway. - Redis & RabbitMQ hoạt động đúng chức năng. - Thanh toán VNPay mock và gửi mail xác nhận thành công. |
| **Chức năng (Functional)** | - 100% các Use Case chính trong SRS được triển khai. - Không còn lỗi critical hoặc major trong test cuối. |
| **Kiểm thử (Testing)** | - ≥ 90% test case pass. - Có evidence test (Postman report, screenshots). |
| **Quản lý dự án (Process)** | - Tuân thủ quy trình Agile 4 Sprint. - Có báo cáo tiến độ hàng tuần và Sprint Review. |
| **Tài liệu (Documentation)** | - Báo cáo Project Plan, SRS, Kiến trúc, Test Plan hoàn chỉnh. - Slide trình bày, biên bản nghiệm thu. |
| **Bàn giao (Delivery)** | - Docker Compose chạy được toàn bộ hệ thống. - Demo thực tế thành công trước giảng viên. |

**10.3. Kế hoạch bàn giao chi tiết**

| **Nội dung bàn giao** | **Định dạng / Công cụ** | **Người chịu trách nhiệm** | **Thời điểm bàn giao** |
| --- | --- | --- | --- |
| Mã nguồn hệ thống (FE + BE + Gateway) | GitHub Repository | Phong, Hoàng | Tuần 10 |
| Cấu hình Docker Compose + .env mẫu | .yml + .env.example | Hoàng | Tuần 10 |
| CSDL mẫu (seed data) | .sql / script | Minh | Tuần 10 |
| Báo cáo hoàn chỉnh | .docx / .pdf | Thanh, Thắng | Tuần 10 |
| Slide trình bày | .pptx / Google Slides | Thắng | Tuần 10 |
| Biên bản nghiệm thu (demo) | .docx | Nhóm + GVHD | Cuối tuần 10 |

**10.4. Đánh giá sau bàn giao**

Sau khi bàn giao sản phẩm, nhóm sẽ thực hiện **Sprint Retrospective cuối cùng**, đánh giá lại:

* Điểm mạnh: tiến độ đúng hạn, teamwork tốt, không tồn backlog.
* Điểm yếu: áp lực thời gian ở sprint cuối, code review còn thủ công.
* Cải tiến cho dự án sau: tự động hóa CI/CD, tách deployment thật, thêm test coverage.