# **1. Giới thiệu**

Dự án GoShop là một hệ thống thương mại điện tử full‑stack: backend TypeScript/Node.js (Prisma, Express/Koa hoặc tương tự), frontend khách hàng và admin sử dụng Vite + React/TS. Mục tiêu chính là cung cấp nền tảng quản lý sản phẩm, giỏ hàng, thanh toán và quản trị đơn hàng cho cửa hàng trực tuyến.

## **1.1. Mục đích**

Xây dựng nền tảng bán hàng trực tuyến ổn định, mở rộng được và dễ vận hành. Hỗ trợ quy trình bán hàng: quản lý danh mục, sản phẩm, đơn hàng, thanh toán, thống kê. Cung cấp giao diện admin để quản trị và xử lý đơn hàng nhanh chóng.

## **1.2. Bối cảnh**

Dự án Goshop là một hệ thống ứng dụng web (Web Application) phục vụ nhu cầu thương mại điện tử. Hệ thống được phát triển chủ yếu dựa trên ngôn ngữ TypeScript (76.1%), kết hợp với HTML và JavaScript để xây dựng giao diện người dùng và logic xử lý.

Đặc điểm kỹ thuật của dự án đòi hỏi việc kiểm thử phải chú trọng vào tính tương thích của trình duyệt (do sử dụng HTML/JS) và tính chặt chẽ của dữ liệu (do đặc thù của TypeScript) trong các luồng giao dịch mua sắm.

## **1.3. Phạm vi**

Phạm vi kiểm thử của dự án GoShop tập trung vào việc xác minh và đảm bảo các chức năng chính của hệ thống thương mại điện tử hoạt động đúng theo yêu cầu nghiệp vụ và mang lại trải nghiệm người dùng ổn định.

**Chức năng người dùng (Frontend):**

Đăng ký và đăng nhập tài khoản.

Xem danh mục sản phẩm và chi tiết sản phẩm.

Tìm kiếm và lọc sản phẩm theo tiêu chí.

Thêm, chỉnh sửa, hoặc xóa sản phẩm trong giỏ hàng.

Thực hiện quy trình thanh toán (checkout).

Xem lịch sử và trạng thái đơn hàng.

**Chức năng quản trị (Admin):**

Quản lý danh mục và sản phẩm.

Quản lý người dùng, đơn hàng

**Kiểm thử tích hợp (Integration Test):**

Kiểm tra luồng dữ liệu giữa Frontend – Backend – Cơ sở dữ liệu.

Kiểm thử các API chính như đăng nhập, thanh toán, và truy xuất sản phẩm.

Kiểm thử giao diện và khả năng sử dụng (UI/UX Test):

Đảm bảo bố cục, màu sắc, và trải nghiệm người dùng nhất quán.

**Kiểm thử hồi quy (Regression Test):**

Đảm bảo các chức năng cũ vẫn hoạt động bình thường sau khi cập nhật hoặc thêm mới tính năng.

## **1.4. Danh sách rủi ro**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Rủi ro tiềm ẩn** | **Mức độ ảnh hưởng** | **Biện pháp giảm thiểu** |
| 1 | Môi trường kiểm thử chưa được thiết lập đúng cách | **Cao** | Kiểm tra kỹ cấu hình (configuration) trước mỗi lần build. |
| 2 | Dữ liệu dùng để kiểm thử thiếu hoặc chưa đầy đủ | **Trung bình** | Chuẩn bị và rà soát bộ dữ liệu mẫu (mock data) đầy đủ hơn. |
| 3 | Lỗi tích hợp cổng thanh toán Stripe | **Cao** | Kiểm tra lại cấu hình và sandbox của cổng thanh toán. |
| 4 | Các module Tìm Kiếm hoặc Lọc hoạt động chậm | **Trung bình** | Sử dụng bộ dữ liệu test có kích thước vừa đủ để tối ưu hiệu năng. |
| 5 | Giao diện hiển thị sai lệch giữa các trình duyệt/thiết bị | **Cao** | Thực hiện kiểm thử tương thích (Cross-browser/device testing) trên nhiều môi trường. |

# **2. Hạng mục được kiểm thử**

## **2.1. Chức năng**

Tập trung kiểm tra các luồng nghiệp vụ cốt lõi của hệ thống thương mại điện tử GoShop để đảm bảo hoạt động đúng theo yêu cầu thiết kế.

Quản lý tài khoản (Auth): Đăng ký, Đăng nhập (Email/Password, Social Login nếu có), Quên mật khẩu, Đổi mật khẩu, Cập nhật thông tin cá nhân, Quản lý địa chỉ giao hàng.

Quản lý sản phẩm (Product): Hiển thị danh sách sản phẩm, Tìm kiếm, Lọc sản phẩm (theo giá, danh mục), Xem chi tiết sản phẩm.

Giỏ hàng (Cart): Thêm sản phẩm, Cập nhật số lượng, Xóa sản phẩm, Tính toán tổng tiền tạm tính.

Đặt hàng và Thanh toán (Checkout): Nhập địa chỉ, Chọn phương thức thanh toán, Hoàn tất đơn hàng.

Quản lý đơn hàng (Order): Xem lịch sử đơn hàng, Theo dõi trạng thái đơn hàng (Pending, Accepted, Rejected).

Thống kê đơn hàng (Stats): Xem biểu đồ lợi nhuận, biểu đồ tròn tỉ lệ đơn hàng, xuất biểu đồ dưới dạng excel.

## **2.2. Tương thích**

Do dự án sử dụng TypeScript/JavaScript và HTML (Web application), cần kiểm thử trên nhiều môi trường khác nhau.

Trình duyệt (Browser): Chrome, Firefox, Safari, Edge (các phiên bản mới nhất và phiên bản cũ hơn 1-2 đời).

Hệ điều hành (OS): Windows, macOS, Linux (cho phía server/dev), iOS, Android (trên trình duyệt mobile).

Thiết bị: Laptop/Desktop (các độ phân giải màn hình khác nhau), Tablet, Mobile (Responsive Design).

## **2.3. Giao diện**

Kiểm tra chi tiết các thành phần giao diện người dùng dựa trên mã nguồn Frontend (HTML/CSS/TS).

Responsive: Kiểm tra hiển thị trên các breakpoint (Mobile, Tablet, Desktop). Đảm bảo không bị vỡ layout, chồng chéo nội dung.

Form Elements: Kiểm tra sự hiển thị và hoạt động của các textbox, checkbox, radio button, dropdown list, button (active/disabled states).

Hình ảnh & Media: Đảm bảo hình ảnh sản phẩm tải đúng, không bị méo, hiển thị placeholder khi lỗi tải ảnh.

## **2.4. Bảo mật**

Authentication & Authorization: Kiểm tra phân quyền truy cập (Người dùng thường không thể truy cập trang Admin). Kiểm tra session timeout, logout an toàn.

Input Validation: Kiểm tra chống các lỗi phổ biến như SQL Injection (nếu dùng SQL) hoặc NoSQL Injection (nếu dùng MongoDB), XSS (Cross-site Scripting) tại các ô tìm kiếm và bình luận.

Data Protection: Đảm bảo mật khẩu được mã hóa (hashing) trong DB, thông tin nhạy cảm không hiển thị trên URL hoặc Console log.

## **2.5. Cơ sở dữ liệu**

Kiểm tra tính toàn vẹn và chính xác của dữ liệu (dựa trên các model được định nghĩa trong code TypeScript/Backend).

Data Integrity: Dữ liệu lưu xuống DB phải khớp với dữ liệu nhập từ Frontend (Ví dụ: Số tiền, số lượng tồn kho).

Data Constraints: Kiểm tra các ràng buộc (Not Null, Unique Key cho Email/Username, Foreign Key giữa Order và User).

CRUD Operations: Đảm bảo các thao tác Thêm, Sửa, Xóa cập nhật đúng trạng thái trong database.

## **2.6. Hiệu năng**

Đánh giá khả năng chịu tải của hệ thống GoShop, đặc biệt là phía Backend (API).

Load Testing: Kiểm tra tốc độ tải trang chủ, trang chi tiết sản phẩm khi có 10, 50, 100 người dùng truy cập đồng thời.

Response Time: Đo thời gian phản hồi của các API quan trọng (Search, Add to Cart, Checkout). Mục tiêu < 2 giây.

Stress Testing: Tìm điểm giới hạn của hệ thống khi lượng truy cập tăng đột biến (ví dụ mô phỏng ngày sale).

## **2.7. Đám đông**

Mở dự án cho nhiều người dùng thực tế tham gia test.

Beta Testing: Mời một nhóm người dùng hạn chế trải nghiệm thử sản phẩm để thu thập feedback về trải nghiệm thực tế và các lỗi lạ mà team nội bộ chưa phát hiện.

## **2.8. API**

Endpoint Validations: Kiểm tra tất cả các endpoint (GET, POST, PUT, DELETE) trả về đúng Status Code (200, 201, 400, 401, 404, 500).

Payload Verification: Kiểm tra cấu trúc JSON request và response trả về có đúng định dạng document không.

Business Logic qua API: Kiểm tra logic tính toán (giá tiền, tồn kho) trực tiếp qua API mà không cần thông qua giao diện.

Error Handling: Gửi các request thiếu trường bắt buộc hoặc sai định dạng để kiểm tra phản hồi lỗi của Backend.

# **3. Hạng mục không được kiểm thử**

## **3.1. Các chức năng thuộc hệ thống bên thứ ba**

Nhóm không thực hiện kiểm thử các chức năng xử lý nội bộ của cổng thanh toán bên thứ 3. Chỉ đảm bảo tính ổn định, hiệu năng khi kết nối với công thanh toán.

## **3.2. Kiểm thử bảo mật nâng cao**

Không thực hiện kiểm thử xâm nhập (Penetration Testing) và các bài kiểm thử bảo mật chuyên sâu.

Nhóm chỉ thực hiện kiểm thử bảo mật ở mức cơ bản, tập trung vào kiểm soát truy cập và phân quyền người dùng.

## **3.3. Kiểm thử khả năng sử dụng chuyên sâu**

Khảo sát người dùng thực tế (User Acceptance Survey): Không tổ chức các phiên trải nghiệm thực tế với nhóm người dùng mẫu (Beta testers) trong giai đoạn này (trừ khi có yêu cầu UAT riêng biệt).

Tiêu chuẩn tiếp cận (Accessibility Compliance): Không kiểm thử sự tuân thủ nghiêm ngặt các tiêu chuẩn WCAG (Web Content Accessibility Guidelines) dành cho người khuyết tật, trừ các yêu cầu cơ bản về độ tương phản và điều hướng bằng bàn phím.

# **4. Tiêu chí kiểm thử chấp nhận**

## **4.1. Độ bao phủ kiểm thử**

Tiêu chí chấp nhận kiểm thử – GoShop

Bao phủ kiểm thử ≥ 95% chức năng.

Hiệu năng tải trang ≤ 5 giây.

Mật khẩu mã hóa, không lộ dữ liệu người dùng.

Trưởng nhóm xác nhận trước chuyển sang UAT.

## **4.2. Tỷ lệ trường hợp kiêm thử đạt**

Tỷ lệ test case pass ≥ 90%.

Hạn chế các lỗi Critical/High severity.

## **4.3. Số lượng trường hợp kiểm thử**

Toàn bộ test case ở các cấp độ sau phải được thực thi 100%:

Kiểm thử đơn vị (Unit Test)

Kiểm thử tích hợp (Integration Test)

Kiểm thử hệ thống (System Test)

## **4.4. Số lượng lỗi**

Lỗi Critical (Nghiêm trọng): Tối đa 4 lỗi. Hệ thống không được phép dính nhiều lỗi gây crash, mất dữ liệu, hoặc chặn luồng thanh toán tại thời điểm bàn giao.

Lỗi Major (Lớn): Tối đa 08 lỗi. Các lỗi này phải có phương án khắc phục tạm thời (Workaround) được Product Owner chấp thuận và có kế hoạch sửa chữa (Hotfix) ngay sau khi release.

Lỗi Minor/Cosmetic (Nhỏ/Giao diện): Cho phép tồn tại một số lượng nhỏ nhưng không được ảnh hưởng đến trải nghiệm người dùng chính và phải được ghi nhận vào backlog để xử lý ở Sprint kế tiếp.

## **4.5. Mật độ kiểm thử đơn vị**

Mật độ kiểm thử phải đạt tối thiểu 30 Unit Test Case trên 500 dòng mã nguồn đối với các module xử lý logic nghiệp vụ (Backend Services, Utility Functions).

## **4.6. Độ bao phủ mã nguồn**

Để đảm bảo chất lượng mã nguồn và giảm thiểu tối đa các lỗi tiềm ẩn trong logic xử lý, nhóm đã thiết lập các ngưỡng (thresholds) về độ bao phủ mã nguồn như một phần của tiêu chí chấp nhận kiểm thử (Acceptance Criteria). Các thông số này được đo lường thông qua các công cụ kiểm thử tự động (như Jest, JaCoCo hoặc Istanbul) trong quá trình thực thi các bài kiểm thử đơn vị (Unit Test).

1. Độ bao phủ câu lệnh (Statement Coverage) ≥ 70%

Định nghĩa: Đây là chỉ số đo lường tỷ lệ các dòng lệnh (statements) trong mã nguồn đã được thực thi ít nhất một lần bởi bộ kiểm thử.

Ý nghĩa: Với ngưỡng 70%, nhóm đảm bảo rằng phần lớn các logic xử lý, các dòng lệnh gán, và các lệnh gọi hàm đã được kiểm tra. Chỉ số này giúp xác nhận rằng không có đoạn mã "chết" (dead code) hoặc các khối logic quan trọng nào bị bỏ qua trong các module cốt lõi của GoShop.

2. Độ bao phủ nhánh (Branch Coverage) ≥ 50%

Định nghĩa: Chỉ số này đo lường tỷ lệ các nhánh quyết định (như các cấu trúc if-else, switch-case) đã được kiểm tra qua cả hai trạng thái Đúng (True) và Sai (False).

Ý nghĩa: Ngưỡng 50% tập trung vào việc kiểm soát các luồng rẽ nhánh logic. Mặc dù thấp hơn Statement Coverage do tính phức tạp của các tổ hợp logic, nhưng con số này đảm bảo rằng các kịch bản ngoại lệ (exception paths) và các luồng nghiệp vụ rẽ nhánh quan trọng (ví dụ: nếu thanh toán thất bại thì làm gì, nếu hết hàng thì thông báo ra sao) đã được bao phủ đáng kể.

3. Độ bao phủ hàm (Function Coverage) ≥ 70%

Định nghĩa: Đo lường tỷ lệ các hàm (functions) hoặc phương thức (methods) trong toàn bộ dự án đã được gọi và thực thi bởi bộ kiểm thử.

Ý nghĩa: Với con số 70%, nhóm khẳng định rằng gần như toàn bộ các tính năng nghiệp vụ được đóng gói trong các hàm (như hàm tính tổng tiền, hàm xác thực JWT, hàm xử lý giỏ hàng) đã được kiểm thử trực tiếp. Đây là chỉ số quan trọng nhất để minh chứng cho tính toàn vẹn của các dịch vụ (services) trong hệ thống.

Việc thiết lập các ngưỡng bao phủ này giúp nhóm đạt được các mục tiêu sau:

Định lượng chất lượng: Thay vì nói "đã test kỹ", các con số 70%, 50%, và 70% cung cấp bằng chứng cụ thể cho hội đồng chấm đồ án về mức độ hoàn thiện của mã nguồn.

Tự động hóa quy trình QA: Các thông số này được tích hợp vào quy trình CI/CD. Nếu mã nguồn mới không đạt được các ngưỡng này, hệ thống sẽ cảnh báo, buộc lập trình viên phải bổ sung Unit Test.

Giảm thiểu nợ kỹ thuật (Technical Debt): Việc duy trì độ bao phủ cao giúp việc nâng cấp và bảo trì hệ thống GoShop trong tương lai trở nên an toàn hơn, tránh lỗi hồi quy (regression bugs).

## **4.7. Quy trình CI/CD**

Mọi luồng công việc (Workflows) được thiết lập trên GitHub Actions, bao gồm các công đoạn từ đóng gói mã nguồn, kiểm tra định dạng (Linting) cho đến xây dựng (Build) dự án GoShop, đều phải được thực thi trọn vẹn và đạt trạng thái "Success" (được đánh dấu tích xanh).

Quy trình này được coi là không đạt nếu có bất kỳ "Job" hoặc "Step" nào trong luồng CI/CD bị ngắt quãng, gặp lỗi hệ thống hoặc kết thúc với trạng thái "Failed". Việc đạt được trạng thái thành công hoàn toàn là điều kiện tiên quyết để đảm bảo mã nguồn mới không phá vỡ cấu trúc hiện tại và sẵn sàng cho các giai đoạn triển khai tiếp theo.Không được tồn tại bất kỳ lỗi kiểm thử nào trong pipeline tự động.

Trong quá trình thực thi Pipeline, hệ thống tự động khởi chạy toàn bộ bộ kiểm thử (bao gồm Unit Test và Integration Test). Tiêu chí chấp nhận bắt buộc là tỷ lệ vượt qua phải đạt 100%, đồng nghĩa với việc không có bất kỳ kịch bản kiểm thử nào bị thất bại (Zero Failure).

Bất kỳ một lỗi kiểm thử nào phát sinh trong Pipeline tự động, dù là nhỏ nhất, cũng sẽ trực tiếp ngăn chặn quy trình "Merge" mã nguồn vào nhánh chính. Điều này đảm bảo tính ổn định tuyệt đối của hệ thống GoShop, ngăn ngừa các lỗi hồi quy (Regression bugs) và duy trì chất lượng phần mềm ở mức cao nhất trước khi tích hợp.

# **5. Chiến lược kiểm thử**

## **5.1. Phương pháp kiểm thử**

Để đảm bảo tính chặt chẽ và chất lượng xuyên suốt vòng đời phần mềm, dự án tuân thủ mô hình **V-Model**. Tại đây, quy trình kiểm thử được tiến hành song song và đối xứng với các giai đoạn phát triển. Hệ thống được kiểm soát qua 4 cấp độ: Unit Test, Integration Test, System Test và Acceptance Test, với các Test Case được xây dựng bám sát yêu cầu chức năng và Use Case. Ngoài ra, quy trình CI/CD được thiết lập bằng **GitHub Actions** giúp tự động hóa toàn bộ chuỗi Build, Test và Deploy, nâng cao hiệu suất làm việc.

## 

## **5.2. Loại kiểm thử**

|  |  |
| --- | --- |
| Activity | Vai trò thực hiện |
| Kiểm thử chức năng | Được thực hiện bởi nhóm kiếm thử để đáp ứng yêu cầu chức năng đã thống nhất của ứng dụng website GoShop. Các chức năng sau sẽ được kiểm tra: 1. Đăng Ký 2. Đăng Nhập 3. Giỏ Hàng 4. Thanh Toán 5. Danh Mục Sản Phẩm  6. Sản Phẩm Yêu Thích 7 Quản Lý Sản Phẩm 8 Kiểm Tra Toàn Bộ Đơn Hàng 9 Quản Lý Tài Khoản Người Dùng.  Tất cả các tính năng được kiểm thử sẽ được mô tả chi tiết trong tài liệu về Test Scenario và Test Case . |
| Kiểm thử dữ liệu | Được thực hiện bởi nhóm kiểm thử để xác định cơ sở dữ liệu chứa thông tin của GoShop website quản lý, chạy truy vấn và đáp ứng yêu cầu lưu trữ dữ liệu của người dùng. |
| Kiểm thử bảo mật | Được thực hiện bởi nhóm kiểm thử để đáp ứng các yêu cầu bảo mật đã thỏa thuận cửa GoShop Website. |
| Kiểm thử giao diện | Được thực hiện bởi nhóm kiểm thử để đáp ứng giao diện người dùng được thiết kế đúng, dễ nhìn, dễ dùng và trải nghiệm người dùng. |
| Kiểm thử hiệu năng | Được thực hiện bởi nhóm kiểm thử nhằm đo lường hiệu suất hệ thống - tốc độ, khả năng chịu tải (load), và khối lượng dữ liệu (volume). |
| Kiểm thử tương thích | Được thực hiện trong giai đoạn Integration Testing hoặc System Testing để kiểm tra website có tương thích với cấu hình của các trình duyệt hay không. a. Google Chrome  b. Mozilla Firefox c. Opera d. Mircosoft Edge e. Brave |
| Kiểm thử API | Được thực hiện bởi nhóm kiểm thử để kiểm tra tính chính xác và ổn định của các API về dữ liệu trả về, status code, tính hợp lệ của request và response |

## **5.3. Cấp độ kiểm thử**

|  |  |  |  |
| --- | --- | --- | --- |
| **Loại kiểm thử** | **Mục đích** | **Kỹ thuật** | **Phương pháp** |
| Kiểm thử đơn vị | Kiểm tra tính đúng đắn của các hàm/phương thức | Hộp trắng | Kiểm thử tự động bằng JUnit và Mockito ở môi trường Development/CI |
| Kiểm thử tích hợp | Kiểm tra sự tương tác giữa các module chức năng | Hộp đen | Kiểm thử tự động bằng MockMvc và H2 ở môi trường Development/CI |
| Kiểm thử hệ thống | Kiểm tra các luồng nghiệp vụ và giao diện | Hộp đen | Kiểm thử tự động bằng Selenium (UI, E2E) và K6 (hiệu năng) trong môi trường Development |
| Kiểm thử chấp nhận | Kiểm tra và nghiệm thu sản phẩm | Hộp đen | Kiểm thử thủ công ở môi trường Production |

## 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Loại kiểm thử** | **Cấp độ kiểm thử** | | | |
| **Đơn vị** | **Tích hợp** | **Hệ thống** | **Chấp nhận** |
| Kiểm thử chức năng | X | X | X | X |
| Kiểm thử giao diện |  |  | X | X |
| Kiểm thử hiệu năng |  |  | X |  |
| Kiểm thử tương thích |  |  | X | X |
| Kiểm thử bảo mật | X |  | X |  |
| Kiểm thử dữ liệu |  | X | X |  |
| Kiểm thử đám đông |  |  | X | X |
| Kiểm thử API | X | X |  |  |

# **6. Kế hoạch thực thi kiểm thử**

## **6.1. Nguồn nhân lực**

|  |  |  |
| --- | --- | --- |
| **Người thực hiện** | **Vai Trò** | **Trách nhiệm chính** |
| Võ Văn Luân | Kiểm thử viên/Lập trình viên | Lập kế hoạch kiểm thử, phân bổ nguồn lực, kiểm soát tiến độ và chất lượng đầu ra. |
| Đỗ Tấn Lộc | Kiểm thử viên/Lập trình viên | Thiết kế Test Case, thực hiện kiểm thử các chức năng Frontend (Auth, Cart, Admin), log bug và verify bug. |
| Nguyễn Hưu Anh Khoa | Kiểm thử viên/Lập trình viên | Viết Unit Test , cấu hình pipeline CI/CD trên GitHub Actions, Chịu trách nhiệm review kết quả Test CI/CD. |
| Lê Duy Tín | Kiểm thử viên/Lập trình viên | Hỗ trợ làm rõ yêu cầu nghiệp vụ, thực hiện UAT để nghiệm thu sản phẩm. |

## **6.2. Lịch trình kiểm thử**

# **7. Môi trường kiểm thử**

## **7.1. Phần cứng**

|  |  |  |
| --- | --- | --- |
| **Mục** | **Thành phần** | **Mục đích** |
| Development | Máy tính cá nhân | Lập trình, chạy kiểm thử đơn vị và kiểm thử tích hợp |
| Thiết bị | Desktop, Android, iOS | Kiểm thử giao diện và kiểm thử tương thích |

## **7.2. Phần mềm**

|  |  |  |
| --- | --- | --- |
| **Mục** | **Thành phần** | **Mục đích** |
| Development | VSCode, ReactJS, Local MySQL | Lập trình, chạy kiểm thử đơn vị và kiểm thử tích hợp |
| Trình duyệt | Chrome, Firefox, Brave | Kiểm thử giao diện và kiểm thử tương thích |

## **7.3. Hạ tầng**

|  |  |  |
| --- | --- | --- |
| **Mục** | **Thành phần** | **Mục đích** |
| Production | MySQL, Frontend và Backend deploy trên Railway | Triển khai phần mềm thực tế, thực hiện kiểm thử hệ thống và chấp nhận |
| CI/CD | GitHub Actions | Tự động hóa quy trình Build, Test và Deploy |

# **Tài liệu bàn giao**

|  |  |  |
| --- | --- | --- |
| **Loại hạng mục** | **Định dạng** | **Chi tiết nội dung** |
| **Tài liệu dự án** | Word (.docx) | • Architecture Design & Database Design  • Test Plan  • Báo cáo tổng kết dự án |
| **Tài liệu kiểm thử & Chi tiết** | Excel (.xlsx) | • Danh sách giao diện & Đặc tả Use Case  • Review Checklists  • Test Design, Test Case, Test Report |
| **Mã nguồn & Tự động hóa** | GitHub | • Source Code  • Test Scripts (Unit, Integration, System)  • CI/CD Pipeline (.github/workflows/ci.yml , CD ở railway) |