A blue and white rectangular frame

AI-generated content may be incorrect.**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**A blue circle with text

AI-generated content may be incorrect.**

**BÁO CÁO**

Đề Tài: MicroShop – Website thương mại điện tử với Microservices.

Nhóm sinh viên thực hiện – Nhóm 3

Nguyễn Đức Tài - 3121411186

Lâm Kiến Minh - 3121411132

Mai Hải Hậu - 3121411067

*Thành phố Hồ Chí Minh, tháng 04 năm 2025*

MỤC LỤC

[**I, Tổng quan đề tài** 1](#_Toc195984877)

[1, Giới thiệu đề tài 1](#_Toc195984878)

[2, Lý do chọn đề tài 1](#_Toc195984879)

[3, Phạm vi và đối tượng sử dụng của hệ thống 2](#_Toc195984880)

[4, Quy trình phát triển phần mềm theo mô hình Scrum. 2](#_Toc195984881)

[5, Outline Planning trong Scrum 3](#_Toc195984882)

[6, Thiết kế kiến trúc phần mềm với phương pháp Scrum. 4](#_Toc195984883)

[7, Test-first development trong Scrum 4](#_Toc195984884)

[**II, Phân tích công nghệ** 5](#_Toc195984885)

[1, Nodejs 5](#_Toc195984886)

[2, Một số kiến trúc phổ biến 5](#_Toc195984887)

[3, Các công nghệ được sử dụng trong ShopMicro 8](#_Toc195984888)

[**III, Phân tích và thiết kế dự án** 9](#_Toc195984889)

[1, Giới thiệu dự án 9](#_Toc195984890)

[2, Use Case Diagram 13](#_Toc195984891)

[3, User Story 16](#_Toc195984892)

[4, SPA – Single Page Application 20](#_Toc195984893)

[5, C4 Model 21](#_Toc195984894)

[6, Deploy hệ thống 25](#_Toc195984895)

[7, Giao diện người dùng 26](#_Toc195984896)

HÌNH ẢNH

Hình 2.1 – UC01: Dịch vụ xác thực

Hình 2.2 – UC02: Dịch vụ người dùng

Hình 2.3 – UC03: Dịch vụ sản phẩm

Hình 2.4 – UC04: Dịch vụ thông báo

Hình 2.5 – UC05: Dịch vụ đơn hàng

Hình 2.6 – UC06: Dịch vụ trò chuyện

Hình 2.7 – UC07: Dịch vụ đánh giá

Hình 4 – Sequence diagram cho SPA

Hình 5.1 – System Context Diagram

Hình 5.2 – Container Diagram (Ex: Chức năng đăng nhập)

Hình 5.3 – Component Diagram (Ex: Chức năng đăng nhập)

Hình 6 - Deployment Diagram

Hình 7.1 – Wireframe trang chủ

Hình 7.2 – Wireframe xác minh Email

Hình 7.3 – Wireframe Danh sách Hồ sơ Người bán

Hình 7.4 – Wireframe popup đăng ký(1)

Hình 7.5 – Wireframe popup đăng ký(2)

Hình 7.6 – Wireframe popup đăng nhập

Hình 7.7 – Wireframe chi tiết sản phẩm

Hình 7.8 – Wireframe thanh toán

Hình 7.9 – Wireframe trò chuyện(1)

Hình 7.10 – Wireframe thanh toán thành công

Hình 7.11 – Wireframe quản lý đơn hàng

Hình 7.12 – Wireframe “Trở thành người bán”

Hình 7.13 – Wireframe hồ sơ người bán

Hình 7.14 – Wirefame trò chuyện(2)

Hình 7.15 – Wireframe tạo sản phẩm

Hình 7.16 – Wireframe trạng thái dự án

PHÂN CÔNG

|  |  |  |
| --- | --- | --- |
| **Sinh viên thực hiện** | **Công việc** | **Thái độ** |
| Mai Hải Hậu | Thực hiện Chương 1 + Chương 2:  - **Chương 1**: Mục 1 - 4  - **Chương 2**: Mục 3 | Tốt |
| Lâm Kiến Minh | Thực hiện Chương 1 + Chương 3:  - **Chương 1**: Mục 5 - 7  - **Chương 3**: Mục 7 | Tốt |
| Nguyễn Đức Tài | Thực hiện Chương 2 + Chương 3:  - **Chương 2**: Mục 1, 2  - **Chương 3**: Mục 1 - 6 | Tốt |

# **I, Tổng quan đề tài**

## **1, Giới thiệu đề tài**

Trong thời đại công nghệ số phát triển mạnh mẽ như hiện nay, các nền tảng thương mại điện tử không ngừng được nâng cấp và đổi mới để đáp ứng nhu cầu ngày càng cao của người tiêu dùng và người kinh doanh. Bên cạnh các mô hình bán lẻ truyền thống, xu hướng cung cấp dịch vụ cá nhân theo dạng freelance (lao động tự do) đang ngày càng phổ biến và tạo ra nhiều cơ hội phát triển.

MicroShop là một website thương mại điện tử theo mô hình freelance, nơi kết nối giữa người mua dịch vụ và người bán dịch vụ trên môi trường trực tuyến. Người bán có thể đăng tải các sản phẩm/dịch vụ của mình, tương tác trực tiếp với khách hàng thông qua hệ thống trò chuyện, và nhận đơn hàng để thực hiện. Đồng thời, người mua có thể tìm kiếm, thương lượng và đánh giá dịch vụ sau khi sử dụng. Hệ thống hỗ trợ các chức năng cốt lõi như đăng nhập, xác thực người dùng, trò chuyện real-time, xử lý đơn hàng, thông báo, và đánh giá sản phẩm.

Dự án này được xây dựng theo kiến trúc Microservices hiện đại nhằm đảm bảo khả năng mở rộng, triển khai độc lập và dễ dàng bảo trì hệ thống. Các công nghệ chủ đạo bao gồm Node.js, React, TypeScript, Docker, RabbitMQ, Jenkins, Redis, và Elasticsearch. Dự án được thực hiện theo định hướng học tập thông qua khóa học *Microservices with NodeJS, React, TypeScript, and Kubernetes* trên nền tảng Udemy.

Mục tiêu của đề tài là xây dựng và triển khai thành công hệ thống thương mại điện tử freelance MicroShop, đồng thời áp dụng kiến thức về kiến trúc Microservices để mô hình hóa các chức năng nghiệp vụ phức tạp thành các dịch vụ độc lập, dễ mở rộng và tích hợp.

## **2, Lý do chọn đề tài**

Trong những năm gần đây, kiến trúc Microservices đã trở thành một xu hướng nổi bật trong phát triển phần mềm hiện đại. Thay vì xây dựng hệ thống theo mô hình nguyên khối (Monolithic), Microservices cho phép chia nhỏ hệ thống thành nhiều dịch vụ độc lập, mỗi dịch vụ đảm nhiệm một chức năng riêng biệt và có thể triển khai, mở rộng hoặc bảo trì mà không ảnh hưởng đến toàn bộ hệ thống. Kiến trúc này giúp tăng tính linh hoạt, khả năng mở rộng và dễ dàng tích hợp với các công nghệ mới.

Bên cạnh đó, Node.js là một trong những nền tảng phổ biến trong phát triển backend nhờ khả năng xử lý bất đồng bộ hiệu quả, hiệu suất cao và hệ sinh thái phong phú. Node.js kết hợp rất tốt với kiến trúc Microservices nhờ tốc độ xử lý nhanh, nhẹ và dễ tích hợp với các dịch vụ khác thông qua API hoặc message queue như RabbitMQ.

## **3, Phạm vi và đối tượng sử dụng của hệ thống**

* **Phạm vi:**
  + Xây dựng hệ thống với các dịch vụ chính: Authentication, User, Product, Chat, Order, Review, Notification.
  + Hệ thống hoạt động ở mức cơ bản (prototype), phục vụ mục đích học tập và trình bày trong báo cáo, chưa hướng đến triển khai thương mại.
* **Đối tượng sử dụng:**
  + Người học hoặc nhà phát triển đang tìm hiểu về kiến trúc Microservices.
  + Sinh viên công nghệ thông tin áp dụng làm đồ án hoặc tài liệu tham khảo học tập.
  + Người dùng thử nghiệm hệ thống như khách hàng và người bán dịch vụ trong mô hình freelance.

## **4, Quy trình phát triển phần mềm theo mô hình Scrum.**

**4.1, Scrum là gì?**

Scrumlà một phương pháp agile tổng quát nhưng tập trung vào quản lý phát triển tăng dần (iterative development).

**4.2, Quy trình phát triển phần mềm và vai trò của các quy trình theo mô hình Scrum**

**Trong quy trình phát triển phần mềm theo mô hình Scrum gồm có 3 quy trình:**

* **Outline Planning và Architectural Design**
  + Thu thập và làm rõ các yêu cầu chức năng, phi chức năng của hệ thống.
  + Tạo danh sách (Product Backlog) các yêu cầu hoặc tính năng (User Story) mà sản phẩm cần có.
  + Xác định mô hình tổng thể, các module, giao diện, nền tảng, công nghệ và các quy ước kỹ thuật.
* **Sprint Cycle** 
  + Điểm khởi đầu cho lập kế hoạch là lập danh sách các công việc cần thực hiện cho dự án.
  + **Select:** Team phát triển phải làm việc với khách hàng vàchọn các tính năng và chức năng cần phát triển trong vòng sprint đó.
  + **Develop:** Team sẽ thực hiện thiết kế chi tiết, lập trình, kiểm thử, tích hợp, ...
  + **Review:** Kết thúc Sprint, nhóm tổ chức review, trình bày sản phẩm cho Product Owner và bên đặt hàng.
  + **Assess:** Nhóm tự tổ chức đánh giá quá trình làm việc của Sprint vừa qua. Sau đó vòng sprint tiếp theo sẽ bắt đầu.
* **Project Closure**
  + Khi các vòng Sprint đã tạo ra sản phẩm hoàn chỉnh theo yêu cầu, dự án sẽ kết thúc.

## **5, Outline Planning trong Scrum**

**5.1, Outline Planning trong Scrum là gì?**

**Outline Planning** là giai đoạn lập kế hoạch sơ bộ ban đầu trong quy trình phát triển Scrum. Giúp team và Product Owner hiểu rõ sản phẩm mình cần làm là gì và có kế hoạch sơ bộ để thực hiện.

**5.2, Vai trò của Outline planning trong Scrum**

* Xác định tầm nhìn và mục tiêu, có mục bức tranh tổng quan giúp các thành viên hiểu rõ đích đến cuối cùng.
* Ưu tiên các Epic/User Stories quan trọng nhất.
* Ước lượng tổng quát khối lượng công việc để hoàn thành sản phẩm.
* Thiết lập roadmap, xác định các điểm mốc quan trọng của dự án.

**5.3, Ảnh hưởng của Outline planning đến thành công của dự án**

* Giúp định hướng rõ ràng và xác định rõ mục đích và yêu cầu công việc, đích đến cuối cùng
* Tối ưu hóa nguồn lực và đặt những việc ưu tiên lên đầu tránh lãng phí nguồn lực tài nguyên
* Quản lý rủi ro hiệu quả từ kế hoạch tổng quan so với thực hiện thực tế giúp nhận thấy được các rủi ro tiềm tàng từ đó có hướng giải quyết cũng như kế hoạch dự phòng hợp lý, giảm thiểu tối đa các trường hợp không mong muốn có thể xảy đến.

**6, Thiết kế kiến trúc phần mềm với phương pháp Scrum.**

**6.1, Ý nghĩa**

* Thiết kế kiến trúc phần mềm trong phương Scrum có vai trò rất quan trọng, dù Scrum là phương pháp phát triển linh hoạt, ưu tiên phản hồi nhanh và cải tiến liên tục. Nhưng kiến trúc phần mềm được ví như nền móng của một ngôi nhà, nếu không vững chắc hoặc thiếu định hướng từ đầu, sản phẩm sẽ khó mở rộng, bảo trì hoặc thích ứng khi có thay đổi.

**6.2, Lý do cần thực hiện thiết kế kiến trúc ngay giai đoạn đầu**

* Giúp đội phát triển hệ thống hiểu rõ mình cần xây gì, dùng công nghệ nào, cấu trúc hệ thống ra sao.
* Kiến trúc được thiết kế tốt từ đầu dễ thích ứng hơn với các thay đổi yêu cầu trong các Sprint sau.
* Thiết kế cẩn thận từ đầu tránh được việc "đập đi xây lại", tiết kiệm thời gian và nguồn lực.
* Kiến trúc tốt giúp hệ thống ổn định, dễ kiểm thử và triển khai CI/CD liên tục mà không bị gián đoạn.

**7, Test-first development trong Scrum**

**7.1, Test-first development là gì?**

**Test-First Development** là một phương pháp phát triển phần mềm mà trong đó viết các bài kiểm thử (test case) được thực hiện trước khi viết mã thực thi.

**7.2, Vai trò của Test-first development trong Scrum**

* Test-First Development giúp team viết test trước khi code, đảm bảo:
  + Mọi chức năng được kiểm tra ngay từ đầu.
  + Giảm lỗi nghiêm trọng khi phần mềm đến tay khách hàng.
* Việc tích hợp Test-First Development giúp tự động hóa quá trình kiểm thử, phục vụ tốt cho các pipeline (CI/CD).
* Trong mô hình scrum thường có những yêu cầu thay đổi chức năng từ khách hàng, khi có bộ test đầy đủ viết từ trước nhờ Test-First Development, thì team dễ dàng refactor code hoặc thêm chức năng mới mà không lo phá vỡ chức năng cũ.

# **II, Phân tích công nghệ**

## **1, Nodejs**

**1.1, Nodejs là gì?**

* Node.js là một môi trường runtime mã nguồn mở dành cho JavaScript, được xây dựa trên V8 Engine của Google Chrome. Node.js giúp chạy JavaScript phía server, thay vì chỉ hoạt động trên trình duyệt như trước đây.

**1.2, Đặc điểm nổi bật của Node.js:**

* Node.js sử dụng mô hình non-blocking I/O, cho phép xử lý đồng thời hàng nghìn kết nối mà không gây tắc nghẽn, giúp tối ưu hiệu suất và khả năng mở rộng của ứng dụng.​
* Nodejs sử dụng JavaScript cho cả phía máy khách và máy chủ, giúp đồng bộ hóa và đơn giản hóa quá trình phát triển ứng dụng web.​

**1.3, Ứng dụng của Node.js:**

* Xây dựng các dịch vụ web như RESTful và GraphQL để giao tiếp giữa các hệ thống.​
* Tạo các ứng dụng như chat trực tuyến, thông báo tức thì và công cụ cộng tác.​
* Triển khai được một số kiến trúc phổ biến như: Microservices, MVC, Event-Driven, Layered, … để xây dựng ứng dụng, tùy thuộc vào quy mô, yêu cầu và đặc điểm của dự án.

## **2, Một số kiến trúc phổ biến**

**2.1, Monolithic Architecture**

**Định nghĩa**

Monolithic là kiến trúc truyền thống, nơi toàn bộ ứng dụng được xây dựng như một khối duy nhất, trong đó tất cả các thành phần (UI, logic xử lý, cơ sở dữ liệu) được đóng gói vào một ứng dụng duy nhất.

**Đặc điểm**

* Mọi chức năng đều nằm trong một codebase duy nhất.
* Các thành phần trong ứng dụng chia sẻ tài nguyên chung.
* Việc triển khai và bảo trì đơn giản nhưng khó mở rộng khi ứng dụng lớn.

**Ưu điểm**

* Dễ dàng triển khai, quản lý.
* Không cần quản lý giao tiếp giữa các service.
* Phù hợp với các ứng dụng nhỏ, startup.

**Nhược điểm**

* Khó mở rộng khi ứng dụng phát triển lớn.
* Mỗi lần cập nhật yêu cầu deploy lại toàn bộ hệ thống.
* Dễ bị quá tải do tất cả chức năng chạy trong một tiến trình.

**2.2, Microservices Architecture**

**Định nghĩa**

Microservices là mô hình kiến trúc chia ứng dụng thành nhiều dịch vụ nhỏ, độc lập. Mỗi dịch vụ đảm nhiệm một chức năng riêng biệt và giao tiếp với nhau qua API.

**Đặc điểm**

* Mỗi service hoạt động độc lập, có cơ sở dữ liệu riêng.
* Giao tiếp giữa các service thường qua REST API, GraphQL, hoặc gRPC.
* Dễ dàng mở rộng và triển khai từng phần riêng lẻ.

**Ưu điểm**

* Dễ mở rộng (Scale từng service riêng lẻ).
* Dễ bảo trì, cập nhật từng service mà không ảnh hưởng toàn hệ thống.
* Cải thiện hiệu suất do các service có thể chạy trên nhiều server khác nhau.

**Nhược điểm**

* Phức tạp trong quản lý giao tiếp giữa các service.
* Cần DevOps để triển khai, giám sát hiệu quả.
* Chi phí vận hành cao hơn Monolith.

**2.3, Layered Architecture (Kiến trúc phân tầng)**

**Định nghĩa**

Kiến trúc phân tầng là mô hình tổ chức code thành nhiều tầng khác nhau để tăng tính tổ chức và dễ bảo trì.

**Các tầng phổ biến**

* Presentation Layer (Giao diện): Nhận request từ client.
* Service Layer (Dịch vụ): Xử lý logic nghiệp vụ.
* Data Access Layer (Dữ liệu): Tương tác với cơ sở dữ liệu.

**Ưu điểm**

* Mã nguồn có tổ chức, dễ bảo trì.
* Dễ kiểm tra, debug do từng tầng có nhiệm vụ riêng biệt.

**Nhược điểm**

* Hiệu suất có thể giảm do dữ liệu cần đi qua nhiều tầng.
* Thiếu linh hoạt khi mở rộng.

**2.4, Clean Architecture**

**Định nghĩa**

Clean Architecture là một kiến trúc tập trung vào sự tách biệt giữa các tầng của hệ thống, đảm bảo rằng code có thể tái sử dụng, dễ bảo trì và mở rộng.

**Các nguyên tắc chính**

* Entities (Thực thể): Định nghĩa dữ liệu cốt lõi của ứng dụng.
* Use Cases (Trường hợp sử dụng): Chứa logic nghiệp vụ, độc lập với framework.
* Interface Adapters (Giao diện kết nối): Chuyển đổi dữ liệu giữa Use Cases và Database/API.
* Infrastructure (Hạ tầng): Các framework, thư viện, cơ sở dữ liệu.

**Ưu điểm**

* Mã nguồn dễ mở rộng, dễ test.
* Giúp ứng dụng ít phụ thuộc vào framework cụ thể.

**Nhược điểm**

* Mất nhiều thời gian thiết kế ban đầu.
* Yêu cầu lập trình viên có hiểu biết vững về kiến trúc phần mềm.

## **3, Các công nghệ được sử dụng trong ShopMicro**

* **Front end: React + TailwindCSS + TypeScript**
  + **React:** là 1 thư viện JavaScript phổ biến để xây dựng giao diện người dùng (UI) theo dạng component.
  + **TailwindCSS:** là 1 thư viện CSS utility-first, giúp tạo UI nhanh chóng mà không cần viết nhiều CSS thủ công.
  + **TypeScript:** là một phiên bản nâng cao của JavaScript với việc bổ sung tính năng kiểu dữ liệu tĩnh, giúp cải thiện khả năng phát triển và bảo trì mã nguồn.
* **Backend: Node.js + Express + TypeScript**
  + **Node.js:** là một môi trường chạy JavaScript trên server, giúp xây dựng backend mà không cần dùng các ngôn ngữ như PHP, Java, Python.
  + **Express.js:** là 1 framework nhẹ và phổ biến của Node.js, giúp tạo API RESTful nhanh chóng và dễ dàng.
* **RabbitMQ (Message Broker - Event-driven)**
  + RabbitMQ là một message broker giúp các dịch vụ trong hệ thống microservices giao tiếp với nhau thông qua cơ chế event-driven.
  + Thay vì gọi API trực tiếp (synchronous), RabbitMQ giúp gửi và nhận tin nhắn giữa các service một cách bất đồng bộ (asynchronous).
* **Redis (Caching & Message Queue)**
  + Redis là một cơ sở dữ liệu dạng key-value lưu trữ trên RAM, giúp truy xuất dữ liệu nhanh hơn so với truy vấn từ database chính.
  + **Caching:** Lưu trữ dữ liệu tạm thời (ví dụ: dữ liệu người dùng, kết quả truy vấn) để tăng tốc độ xử lý.
  + **Message Queue:** Redis cũng có thể hoạt động như một message broker giống RabbitMQ nhưng nhẹ hơn, thích hợp cho các tác vụ đơn giản.
* **Elasticsearch + Kibana (Logging & Search Engine)**
  + **Elasticsearch:** Là một công cụ để tìm kiếm và hiển thị các log,…
  + **Kibana:** Là một dashboard cho công cụ elasticsearch.
* **MySQL, PostgreSQL, MongoDB (Databases per Service)**
  + **MySQL:** Là mộtcơ sở dữ liệu quan hệ phổ biến, có tính toàn vẹn dữ liệu cao.
  + **PostgreSQL:** Tương tự như MySQL nhưng mạnh hơn về tính năng.
  + **MongoDB:** Cơ sở dữ liệu NoSQL, lưu trữ dữ liệu dưới dạng JSON, phù hợp với dữ liệu không có cấu trúc cố định.
* **Docker**
  + **Docker** là một nền tảng phần mềm mã nguồn mở giúp tự động hóa việc triển khai và quản lý các ứng dụng trong môi trường container. Nó cho phép đóng gói ứng dụng cùng với tất cả các thành phần phụ thuộc vào một đơn vị độc lập gọi là "container", đảm bảo ứng dụng chạy nhất quán trên mọi môi trường.
  + **Container** là các gói phần mềm nhẹ, có tính di động cao và chứa tất cả các thành phần cần thiết để chạy ứng dụng (bao gồm mã nguồn, runtime, thư viện, và công cụ hệ thống).

**III, Phân tích và thiết kế dự án**

**1, Giới thiệu dự án**

Website MicroShop là một website freelance thương mại điện tử có các chức năng cơ bản như: dịch vụ **Thông báo**, dịch vụ **Xác thực**, dịch vụ **Người dùng**, dịch vụ **Sản phẩm**, dịch vụ **Trò chuyện**, dịch vụ **Đơn hàng** và dịch vụ **Đánh giá**.

Với dịch vụ **Xác thực**, người sử dụng hệ thống (khách hàng hoặc người bán) sẽ có những chức năng sau:

* **Đăng ký**
  + Người dùng đăng ký tài khoản.
* **Đăng nhập**
  + Người dùng nhập email và mật khẩu đã đăng ký để đăng nhập. Hệ thống kiểm tra thông tin, nếu hợp lệ sẽ cấp quyền truy cập vào hệ thống.
* **Xác thực tài khoản**
  + Người dùng phải xác thực tài khoản sau khi đăng nhập. Nếu không xác thực tài khoản thì người dùng sẽ bị bạn chế truy cập một số chức năng trong hệ thống.
* **Gửi lại email xác thực tài khoản**
  + Nếu người dùng đăng nhập mà email xác thực hết hạn hoặc email gửi bị lỗi, người dùng có thể nhấn và “Resend Verify Email” để nhận lại email xác thực lại tài khoản.
* **Quên mật khẩu** 
  + Khi quên mật khẩu, người dùng bấm vào "Quên mật khẩu" trên giao diện đăng nhập.
  + Người dùng sẽ được chuyển đến trang nhập email đã đăng ký để lấy lại mật khẩu.
  + Hệ thống sẽ gửi đến email người dùng nhập và tiến hành đổi mật khẩu.
* **Tìm kiếm sản phẩm**
  + Người dùng tìm kiếm sản phẩm bằng tên sản phẩm, thời gian giao hàng, khoảng giá.

Với dịch vụ **Người dùng**, người sử dụng hệ thống sẽ có những chức năng sau:

* **Trở thành người bán hàng**
  + Sau khi đăng ký thì người dụng mặc định là người mua hàng. Nếu người mua hàng muốn trở thành người bán thì nhấn vào “Become a seller”, người mua hàng cập nhật hồ sơ của mình thành người bán hàng.
* **Quản lý profile người bán**
  + Người bán có thể cập nhật hồ sơ của mình.
  + Cập nhật số lượng công việc
  + Cập nhật số lượng công việc đang làm
  + Cập nhật số lượng công việc đã hoàn thành
  + Cập nhật số lượng công việc bị hủy

Với dịch vụ **Thông báo,** sẽ quản lý các email thông báo gửi đến dịch vụ khác như dịch vụ **Xác thực, Trò chuyện** và **Đơn hàng.**

* Đối với dịch vụ **Xác thực** thì sẽ có email thông báo “Xác thực tài khoản”, “Quên mật khẩu” và “Đổi mật khẩu”.
* Đối với dịch vụ **Trò chuyện**, có thông báo trò chuyện real-time.
* Dịch vụ **Đơn hàng** sẽ có 5 email gửi cho 5 trường hợp sau:
  + Email 1 là email offer đơn hàng. Người bán sẽ offer một đơn hàng cho khách hàng.
  + Email 2 là email đặt hàng. Nếu người mua đặt hàng thì sẽ có email thông báo gửi đến người bán.
  + Email 3 là gửi email thông báo hóa đơn mua hàng cho người mua khi người mua đã thanh toán đơn hàng.
  + Email 4 là gửi email thông báo yêu cầu chờ hàng khi người bán hàng cần thêm thời gian chuẩn bị đơn hàng.
  + Email 5 là gửi email thông báo xác nhận yêu cầu chờ hàng tới khách hàng, khách hàng sẽ chấp nhận hoặc từ chối yêu cầu chờ hàng của người bán.
  + Email 6 là gửi email thông báo đơn hàng đã được gửi đi từ người bán hàng.

Với dịch vụ **Sản phẩm**, người sử dụng hệ thống sẽ có những chức năng sau:

* **Quản lý sản phẩm**
  + Đối với người bán hàng sẽ có chức năng quản lý sản phẩm như thêm/xóa/sửa sản phẩm.

Với dịch vụ **Trò chuyện**, người bán hàng và khách hàng sẽ trò chuyện với nhau để trao đổi mua bán.

* **Trò chuyện real-time**
  + Khách hàng tìm thấy sản phẩm mà họ quan tâm và nhấp vào nút “Liên hệ”, hệ thống hiển thị một chat box nhỏ để trò chuyện với người bán, khi khách hàng gửi tin nhắn trong chat box thì người bán hàng sẽ nhận được tin nhắn ngay lập tức.
  + Mục đích của dịch vụ Trò chuyện là khách hàng liên hệ với người bán để tạo ra một sản phẩm mà khách hàng cần.
* **Đánh dấu đã đọc tin nhắn**
  + Người dùng đánh dấu tin nhắn đã đọc (1 hoặc nhiều tin nhắn)
* **Offer sản phẩm (Người bán)**
  + Người bán có thể “Offer” sản phẩm được hoành thành trong một khoảng thời gian đề xuất với khách hàng trong trang tin nhắn.

Với dịch vụ **Đơn hàng,** khách hàng có thể mua hàng bằng cách liên hệ với người bán thông qua dịch vụ **Trò chuyện**.

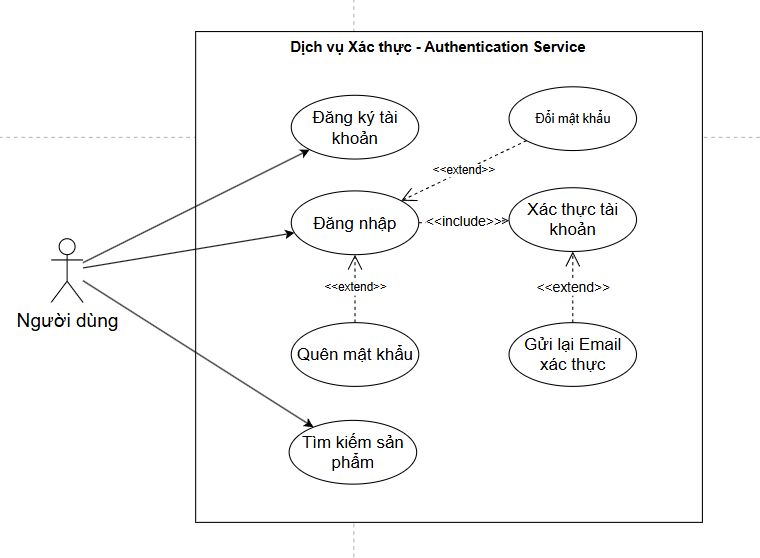
* **Đặt đơn hàng – Người mua**
  + Khi khách hàng nhận được một email “Offer đơn hàng” từ người bán.
  + Khách hàng chọn “Accept Offer” từ email thông báo của người bán thì khách hàng được chuyển sang trang thánh toán.
  + Sau khi khách hàng “Accept Offer”, người bán nhận được một email “Đặt hàng” từ khách hàng.
* **Thanh toán đơn hàng – Người mua**
  + Khách hàng tiến hành thanh toán đơn hàng. Thanh toán thành công, khách hàng nhận được một email “Hóa đơn”.
  + Thanh toán thành công hệ thống hiển thị “Start Order”, khách hàng nhấp vào “Start Order” để đơn hàng bắt đầu được thực hiện từ người bán hàng.
* **Chấp nhận/từ chối thời gian chuẩn bị đơn hàng – Người bán**
  + Khách hàng sẽ nhận được email “Thêm thời gian chuẩn bị đơn hàng” từ người bán.
  + Khách hàng có thể chấp nhận/từ chối yêu cầu.
* **Yêu cầu thêm thời gian chuẩn bị đơn hàng – Người bán**
  + Khi người bán hàng chưa chuẩn bị xong đơn hàng trước dead-line được đề xuất thì người bán nhấp vào “Extend Delivery Date” để yêu cầu khách hàng thêm thời gian chuẩn bị đơn hàng.
  + Người bán sẽ nhận được email “Chấp nhận/từ chốithêm thời gian chuẩn bị đơn hàng” từ khách hàng.
* **Tạo đơn hàng mới – Người bán**
  + Người bán tạo đơn hàng mới với người mua
  + Email thông báo “Người bán đã đặt hàng sản phẩm của bạn” gửi đến người mua.
* **Giao hàng – Người bán**
  + Khi người bán chuẩn bị xong đơn hàng thì người bán nhấp vào “Delivery Now” để thực hiện gửi đơn hàng cho khách hàng.
  + Khách hàng sẽ nhận được email “Giao hàng” từ người bán hàng.

Với dịch vụ **Đánh giá**

* Đánh giá & xếp hạng
  + Khách hàng có để lại bình luận đánh giá và mức độ hài lòng đối với đơn hàng.
  + Người bán có thể phản hồi lại bình luận đánh giá của khách hàng.

**2, Use Case Diagram**

**2.1, UC01: Dịch vụ Xác thực**



*Hình 2.1 – UC01: Dịch vụ xác thực*

**2.2, UC02: Dịch vụ Người dùng**

A diagram of a service

AI-generated content may be incorrect.

*Hình 2.2 – UC02: Dịch vụ người dùng*

**2.3, UC03: Dịch vụ sản phẩm**

A diagram of a service

AI-generated content may be incorrect.

*Hình 2.3 – UC03: Dịch vụ sản phẩm*

**2.4, UC04: Dịch vụ Thông báo**

A diagram of a email notification service

AI-generated content may be incorrect.

*Hình 2.4 –* *UC04: Dịch vụ thông báo*

**2.5, UC05: Dịch vụ Đơn hàng**

A diagram of a customer service

AI-generated content may be incorrect.

*Hình 2.5 – UC05: Dịch vụ đơn hàng*

**2.6, UC06: Dịch vụ trò chuyện**

A diagram of a service

AI-generated content may be incorrect.

*Hình 2.6 – UC06: Dịch vụ trò chuyện*

**2.7, UC07: Dịch vụ đánh giá**

A diagram of a service

AI-generated content may be incorrect.

*Hình 2.7 – UC07: Dịch vụ đánh giá*

**3, User Story**

**3.1, Xác thực & Truy cập của người dùng**

3.1.1, Đăng ký tài khoản người dùng

* User Story:
  + **Với** tư cách là người dùng mới.
  + **Tôi muốn** đăng ký tài khoản (theo cách “Sign Up” cho khách hàng hoặc “Become a Seller” cho người bán).
  + **Để** có thể sử dụng các chức năng của website.

3.1.2, Đăng nhập

* User Story:
  + **Với** tư cách là người dùng đã đăng ký (Khách hàng/Người bán).
  + **Tôi muốn** đăng nhập bằng email và mật khẩu.
  + **Để** truy cập vào tài khoản của mình.

3.1.3, Quên mật khẩu

* User Story:
  + **Với** tư cách là người dùng.
  + **Tôi muốn** có chức năng “Quên mật khẩu”.
  + **Để** lấy lại mật khẩu khi không nhớ để đảm bảo tôi luôn có thể truy cập tài khoản.

3.1.4, Đổi mật khẩu

* User Story:
  + **Với** tư cách là người dùng (Khách hàng/Người bán).
  + **Tôi muốn** thay đổi mật khẩu từ liên kết email “Quên mật khẩu”
  + **Để** tôi có thể bảo mật tài khoản của mình sau khi đặt lại mật khẩu.

**3.2, Quản lý người dùng & Dashboard**

3.2.1, Quản lý người dùng

* User Story:
  + **Với** tư cách là người dùng (Khách hàng/Người bán).
  + **Tôi muốn** cập nhật thông tin Profile của mình.
  + **Để** đảm bảo dữ liệu cá nhân luôn được cập nhật và chính xác.

3.2.2, Lịch sử đơn hàng

* User Story:
  + **Với** tư cách là khách hàng.
  + **Tôi muốn** xem lịch sử đơn hàng của mình.
  + **Để** tôi có thể theo dõi các giao dịch đã mua và trạng thái đơn hàng của mình.

3.2.3, Dashboard người bán hàng

* User Story:
  + **Với** tư cách là người bán hàng.
  + **Tôi muốn** xem dashboard bán hàng của mình.
  + **Để** tôi có thể theo dõi sản phẩm, đơn hàng và đánh giá của khách hàng.

**3.3, Quản lý sản phẩm & trò chuyện**

3.3.1, Xem danh mục & Tìm kiếm sản phẩm

* User Story:
  + **Với** tư cách là khách hàng.
  + **Tôi muốn** xem danh mục sản phẩm và sử dụng chức năng tìm kiếm.
  + **Để** tôi có thể nhanh chóng tìm ra sản phẩm phù hợp với nhu cầu của mình.

3.3.2, Trò chuyện real-time giữa khách hàng và người bán

* User Story:
  + **Với** tư cách là người bán hàng.
  + **Tôi muốn** xem dashboard bán hàng của mình.
  + **Để** tôi có thể theo dõi sản phẩm, đơn hàng và đánh giá của khách hàng.

3.3.3, Offer sản phẩm qua chat

* User Story:
  + **Với** tư cách là người bán hàng.
  + **Tôi muốn** gửi “Offer” sản phẩm cùng với thời gian hoàn thành dự kiến qua chat.
  + **Để** tôi có thể theo dõi sản phẩm, đơn hàng và đánh giá của khách hàng.

**3.4, Xử lý đơn hàng**

3.4.1, Offer sản phẩm qua chat

* User Story:
  + **Với** tư cách là người bán.
  + **Tôi muốn** gửi “Offer” sản phẩm cùng với thời gian hoàn thành dự kiến qua chat.
  + **Để** khách hàng có thể cân nhắc và chốt giao dịch.

3.4.2, Đặt đơn hàng

* **User Story:**
  + **Với** tư cách là khách hàng.
  + **Tôi muốn** “Accept Offer” từ email thông báo và chuyển sang trang thanh toán.
  + **Để** hoàn tất đơn hàng.

3.4.3, Thanh toán đơn hàng

* **User Story:**
  + **Với** tư cách là khách hàng.
  + **Tôi muốn** thanh toán đơn hàng một cách an toàn.
  + **Để** nhận được hóa đơn và kích hoạt quá trình thực hiện đơn hàng.

3.4.4, Yêu cầu thêm thời gian chuẩn bị đơn hàng

* **User Story:**
  + **Với** tư cách là người bán.
  + **Tôi muốn** yêu cầu thêm thời gian chuẩn bị đơn hàng khi cần.
  + **Để** đảm bảo đơn hàng được xử lý đúng hạn và đúng yêu cầu của khách hàng dù có sự chậm trễ.

3.4.5, Giao hàng

* **User Story:**
  + **Với** tư cách là người bán.
  + **Tôi muốn** đánh dấu đơn hàng đã được giao bằng cách nhấp “Delivery Now”.
  + **Để** khách hàng nhận được đơn hàng và thông báo về trạng thái giao hàng.

## **4, SPA – Single Page Application**

* **Single Page Application (SPA)** là một loại ứng dụng web mà:
  + Toàn bộ ứng dụng được tải chỉ một lần duy nhất từ server.
  + Sau đó, mọi chuyển trang và tương tác không làm tải lại trang (no reload).
  + Các dữ liệu mới được lấy qua API và hiển thị bằng JavaScript ngay trên trang hiện tại.
* **Ưu điểm:**
  + Tăng trải nghiệm người dùng (UX) mượt mà, nhanh hơn.
  + Tốc độ điều hướng nhanh (không reload).

A diagram of a program

AI-generated content may be incorrect.

Hình 4 – Sequence diagram cho SPA

* **Cách hoạt động:**
  + Truy cập trang:
    - Người dùng truy cập trang “shopmicro.com”
    - Trình duyệt gửi request GET/ đến nginx (static server)
    - Nginx trả về index.html + React APP (toàn bộ JS, CSS,… của ứng dụng)
    - SPA lúc này được tải về trình duyệt.
  + Điều hướng trong SPA:
    - Người dùng click chuyển đến trang đăng nhập (/login)
    - React Router sẽ xử lý route “/login” nội bộ.
    - React app sẽ hiển thị trang đăng nhập nhưng không bị reload lại trang.

## **5, C4 Model**

**C4 Model** là một cách tiếp cận trực quan để mô tả kiến trúc phần mềm ở 4 cấp độ chi tiết: Context → Container → Component → Code.

**5.1, System Context Model**

**System Context diagram** là sơ đồ giúp người xem hiểu hệ thống phần mềm đang làm gì, ai tương tác với nó, và liên kết với hệ thống nào khác. Cung cấp bức tranh tổng thể cho người không chuyên về kỹ thuật.

* Xác định rõ ranh giới của hệ thống.
* Giúp hiểu ai sử dụng hệ thống và những hệ thống nào tương tác với nó.

**A diagram of a software application

AI-generated content may be incorrect.**

*Hình 5.1 – System Context Diagram*

* **Giải thích sơ đồ:**
  + Client: là người dùng, sẽ tương tác với hệ thống.
  + API Gateway: là “trung tâm” tiếp nhận request giữa client và các Services.
  + Services: Là hệ thống được chia nhỏ ra thành các dịch vụ.
  + Database: Là thành phần lưu trữ dữ liệu tương ứng cho từng services.
  + Request/Response: Mối liên hệ giữa các thực thể.
* **Giải thích cách hoạt động:**
  + Request bắt đầu từ phía client và gửi đến API Gateway
  + API Gateway có nhiệm vụ xác định request và định tuyến request đến đúng Service tương ứng của request đó.
  + Service sẽ xử lý logic của request này và response về phía client.

**5.2, Container Diagram**

**Container Diagram** là một sơ đồ thể hiện kiến trúc phần mềm ở mức độ tổng quan và cách các chức năng được phân chia giữa các container. Giúp các developer và kiến trúc sư hiểu được kiến trúc tổng thể của hệ thống ở cấp triển khai logic.

* Mỗi container là một ứng dụng thực thi độc lập, có vai trò rõ ràng trong hệ thống.
* Mô tả cách các container giao tiếp với nhau thông qua các giao thức như HTTP, AMQP,...
* Hiển thị công nghệ sử dụng trong mỗi container.

A diagram of a software system

AI-generated content may be incorrect.

*Hình 5.2 – Container Diagram (Ex: Chức năng đăng nhập)*

* **Giải thích sơ đồ:**
  + User: là người dùng, sẽ tương tác với hệ thống.
  + Container: Là một phần của hệ thống chạy độc lập, thực hiện một vai trò cụ thể như frontend, backend, database...
  + HTTP/HTTPS: Là giao thức truyền tải dữ liệu giữa client và server (Giao tiếp không mã hóa/Giao tiếp mã hóa).
  + SQL/TCP: Ngôn ngữ dùng để truy vấn và thao tác với cơ sở dữ liệu quan hệ. Và giao thức truyền thông tin giữa các máy tính qua mạng.
* **Giải thích cách hoạt động – Sign In Feature:**
  + Người dụng nhập thông tin vào form đăng nhập rồi bấm nút “Sign In”.
  + Lúc này, request “Sign In” được gửi từ Web Browser đến API Gateway bằng phương thức HTTP/HTTPS.
  + API Gateway sẽ kiểm tra và xác đỉnh request nên được gửi đến Authenthication Service.
  + Authenthication Service sẽ xử lý logic của chức năng Sign In và xảy ra 2 luồng:
    - Luồng 1: Xử lý logic của chức năng Sign In với database.
    - Luồng 2: Xử lý logic gửi mail của chức năng Sign In với Notification Services. => Gửi mail đến cho người dùng.

**5.3, Component Diagram**

**Component Diagram** là sơ đồ mô tả chi tiết hệ thống bên trong các container như thế nào. Để giúp các developer hiểu rõ kiến trúc bên trong một module hay dịch vụ.

* + Hiển thị cấu trúc nội tại và cách các component tương tác với nhau.
  + Các component thường đại diện cho một nhóm chức năng cụ thể.
  + Hữu ích để hiểu phân chia trách nhiệm bên trong mỗi ứng dụng.

A diagram of a software application

AI-generated content may be incorrect.

*Hình 5.3 – Component Diagram (Ex: Chức năng đăng nhập)*

* **Giải thích sơ đồ:**
  + Component: Là các phần logic nhỏ hơn nằm bên trong một Container, như các module, class, service, controller...
  + Container: Là một dịch vụ riêng biệt trong hệ thống.
  + Software System: Là một hệ thống thực hiện một chức năng nhất định cho người dùng hoặc các hệ thống khác.
* **Giải thích cách hoạt động – Sign In Feature:**
  + Người dụng nhập thông tin vào form đăng nhập rồi bấm nút “Sign In”.
  + Lúc này, request “Sign In” được gửi từ Web Browser đến API Gateway bằng phương thức HTTP/HTTPS.
  + API Gateway sẽ kiểm tra và xác đỉnh request nên được gửi đến Authenthication Service.
  + Request đi vào Componet “Auth Route” của Auth Services. Nhiệm vụ của “Auth Route” sẽ định nghĩa endpoint của “/signin”, endpoint “/signin” là một URL cụ thể tại một HTTP method mà Auth Service cung cấp. Tiếp tục request được định tuyến đến “Auth Controller”
  + “Auth Controller” có 2 luồng đi:
    - Luồng thứ 1: Xử lý logic request Sign In bằng cách gửi request đến “Auth Services”, “Auth Services” có nhiệm vụ lấy dữ liệu của người dùng đã đăng ký trong DB, để “Auth Controller” xử lý logic.
    - Luồng thứ 2: Trong trường hợp người dùng đăng nhập trên một thiết bị khác hoặc trình duyệt khác. “Auth Controller” sẽ kiểm tra và gửi một event (dạng message) đến RabbitMQ (Message Broker). RabbitMQ sẽ queue event này đến Notification Service. Notification Service sẽ nhận được event và thực hiện gửi email chứa mã OTP xác thực đến người dùng.

**6, Deploy hệ thống**

**Deployment Diagram** là sơ đồ dùng để mô tả cách các thành phần phần mềm được triển khai trên phần cứng, tức là cách các module phần mềm được phân phối và chạy trên các node vật lý hoặc ảo như server, container, máy chủ ảo,...

* Mô tả cấu trúc hạ tầng của hệ thống.
* Cho biết các thành phần phần mềm chạy ở đâu.
* Biểu diễn mối quan hệ triển khai, kết nối mạng, và giao tiếp giữa các thành phần hệ thống ở cấp độ vật lý.

A diagram of a docker host

AI-generated content may be incorrect.

*Hình 6 - Deployment Diagram*

**7, Giao diện người dùng**

**7.1. Giao diện Trang chủ**

* **Header**:
  + Thương hiệu: Logo "SHOP MICRO".
  + Điều hướng: Các tùy chọn "Trở thành Người bán", "Đăng nhập" và "Đăng ký".
* **Content**:
  + Một banner nổi bật, có thể hiển thị khuyến mãi hoặc dịch vụ nổi bật.
  + Danh sách các danh mục dịch vụ (Dịch vụ A, Dịch vụ B, Dịch vụ C, Dịch vụ D, Dịch vụ E) để dễ dàng điều hướng.
* **Footer**: Footer tiêu chuẩn với các liên kết hoặc tùy chọn điều hướng bổ sung.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.1 – Wireframe trang chủ*

**7.2, Giao diện xác minh Email**

* **Purpose**: Yêu cầu người dùng xác minh địa chỉ email trong quá trình đăng ký.
* **Header**:
  + Thương hiệu ứng dụng: "Jobber".
  + Điều hướng: "Đơn hàng" và "Trở thành Người bán".
* **Content**:
  + Một banner, có thể để đảm bảo tính nhất quán về mặt hình ảnh hoặc cung cấp thông tin ngữ cảnh.
  + Danh sách các hồ sơ người bán được hiển thị dưới dạng lưới hoặc danh sách (các hồ sơ được liệt kê), cho phép người dùng duyệt qua các người bán có sẵn.
* **Footer**: Nhất quán với các Giao diện khác, có thể chứa điều hướng hoặc thông tin ứng dụng.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.2 – Wireframe xác minh Email*

**7.3, Giao diện Danh sách Hồ sơ Người bán**

* **Purpose**: Hiển thị danh sách toàn diện các hồ sơ người bán.
* **Header**: Tương tự Trang 2, với thương hiệu "Jobber" và các tùy chọn điều hướng ("Đơn hàng", "Trở thành Người bán").
* **Content**:
  + Một banner để tạo sức hút thị giác hoặc nội dung nổi bật.
  + Danh sách mở rộng các hồ sơ người bán (15 hồ sơ được liệt kê), cho thấy có thể sử dụng phân trang hoặc cuộn vô hạn để duyệt.
* **Footer**: Thiết kế footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.3 – Wireframe Danh sách Hồ sơ Người bán*

**7.4, Popup đăng ký - Bước 1**

* **Purpose**: Bắt đầu quá trình đăng ký người dùng.
* **Header**: Header "THAM GIA JOBBER".
* **Content**:
  + Các trường nhập liệu:
  + Tên người dùng
  + Email
  + Mật khẩu
  + Nút "TIẾP TỤC" để chuyển sang bước tiếp theo.
* **Thiết kế**: Sạch sẽ và tối giản, tập trung vào nhập liệu người dùng với nhãn rõ ràng.

A screenshot of a login form

AI-generated content may be incorrect.

*Hình 7.4 – Wireframe popup đăng ký(1)*

**7.5, Popup Đăng ký - Bước 2**

* **Purpose**: Hoàn tất quá trình đăng ký người dùng.
* **Header**: Header "THAM GIA JOBBER".
* **Content**:
  + Các trường nhập liệu:
  + Lựa chọn quốc gia
  + Tải ảnh hồ sơ
  + Nút "ĐĂNG KÝ" để hoàn tất đăng ký.
* **Thiết kế**: Tiếp tục phong cách tối giản, đảm bảo dễ sử dụng.

A screenshot of a sign up form

AI-generated content may be incorrect.

*Hình 7.5 – Wireframe popup đăng ký(2)*

**7.6, Popup Đăng nhập**

* **Purpose**: Cho phép người dùng đã đăng ký đăng nhập.
* **Header**: Header "THAM GIA JOBBER".
* **Content**:
  + Các trường nhập liệu:
  + Tên người dùng hoặc Email
  + Mật khẩu
  + Nút "ĐĂNG NHẬP" để xác thực người dùng.
* **Thiết kế**: Đơn giản và trực quan, ưu tiên truy cập nhanh.

A screenshot of a login screen

AI-generated content may be incorrect.

*Hình 7.6 – Wireframe popup đăng nhập*

**7.7, Giao diện chi tiết sản phẩm**

* **Purpose**: Hiển thị thông tin chi tiết về một sản phẩm cụ thể.
* **Header**: Thương hiệu "Jobber" với điều hướng ("Đơn hàng", "Trở thành Người bán").
* **Content**:
  + Thông tin dịch vụ:
  + Header dịch vụ
  + Giá
  + Giới thiệu dự án
  + Thông tin về người bán
* **Footer**: Footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.7 – Wireframe chi tiết sản phẩm*

**7.8, Giao diện Thanh toán**

* **Purpose**: Hỗ trợ việc mua dịch vụ.
* **Header**: Nhất quán với các Giao diện trước ("Jobber", "Đơn hàng", "Trở thành Người bán").
* **Content**:
  + Lựa chọn danh mục dịch vụ.
  + Biểu mẫu thanh toán để nhập chi tiết thanh toán.
  + Tóm tắt chi tiết dự án.
  + Nút "XÁC NHẬN & THANH TOÁN" để hoàn tất giao dịch.
* **Footer**: Footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.8 – Wireframe thanh toán*

**7.9, Giao diện trò chuyện - Tùy chọn 1**

* **Purpose**: Hiển thị các cuộc hội thoại của người dùng với người bán.
* **Header**: Header "Jobber" tiêu chuẩn với điều hướng.
* **Content**:
  + Ngữ cảnh danh mục dịch vụ.
  + Phần "Tất cả Cuộc hội thoại" với văn bản giữ chỗ cho tin nhắn.
* **Footer**: Footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.9 – Wireframe trò chuyện(1)*

**10. Giao diện thanh toán thành công**

* **Purpose**: Xác nhận thanh toán thành công và cung cấp các tùy chọn sau khi mua.
* **Header**: Header "Jobber" tiêu chuẩn.
* **Content**:
  + Ngữ cảnh danh mục dịch vụ.
  + Xác nhận thanh toán thành công.
  + Tóm tắt chi tiết dự án.
  + Tính năng "Hỏi Người bán Điều gì đó" để liên lạc trực tiếp.
* **Footer**: Footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.10 – Wireframe thanh toán thành công*

**11. Giao diện quản lý đơn hàng**

* **Purpose**: Cho phép người dùng quản lý các đơn hàng của họ.
* **Header**: Header "Jobber" tiêu chuẩn.
* **Content**:
  + Ngữ cảnh danh mục dịch vụ.
  + Các tab để lọc đơn hàng: Đang hoạt động, Hoàn tất, Đã hủy.
  + Một đơn hàng mẫu (Đơn hàng 1) được hiển thị.
* **Footer**: Footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.11 – Wireframe quản lý đơn hàng*

**12. Giao diện “Trở thành người bán”**

* **Purpose**: Cho phép người dùng đăng ký làm người bán.
* **Header**: Header "Jobber" tiêu chuẩn.
* **Content**:
  + Ngữ cảnh danh mục dịch vụ.
  + Biểu mẫu có Header "FORM ĐĂNG KÝ THÀNH SELLER" (Biểu mẫu Đăng ký Người bán).
  + Nút "TẠO HỒ SƠ" để gửi biểu mẫu.
* **Footer**: Footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.12 – Wireframe “Trở thành người bán”*

**7.13, Giao diện hồ sơ người bán**

* **Purpose**: Trưng bày hồ sơ và số liệu hiệu suất của người bán.
* **Header**: Header "Jobber" tiêu chuẩn.
* **Content**:
  + Thông tin người bán:
  + Tên
  + Tổng số công việc
  + Công việc đã hoàn thành
  + Đơn hàng đang thực hiện
  + Đánh giá và nhận xét
  + Thông tin chi tiết về người bán
* **Footer**: Footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.13 – Wireframe hồ sơ người bán*

**7.14, Giao diện trò chuyện - Tùy chọn 2**

* **Purpose**: Giao diện hội thoại thay thế, tương tự Trang 9.
* **Header**: Header "Jobber" tiêu chuẩn.
* **Content**:
  + Ngữ cảnh danh mục dịch vụ.
  + Phần "Tất cả Cuộc hội thoại" với văn bản giữ chỗ cho tin nhắn.
* **Footer**: Footer tiêu chuẩn.

A screenshot of a web page

AI-generated content may be incorrect.

*Hình 7.14 – Wirefame trò chuyện(2)*

**15. Giao diện tạo sản phẩm**

* **Purpose**: Cho phép người bán tạo danh sách dịch vụ.
* **Header**: Header "Jobber" tiêu chuẩn.
* **Content**:
  + Ngữ cảnh danh mục dịch vụ.
  + Biểu mẫu có Header "FORM TẠO SẢN PHẨM" (Biểu mẫu Tạo Sản phẩm).
  + Nút "TẠO SẢN PHẨM" để gửi biểu mẫu.
* **Footer**: Footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.15 – Wireframe tạo sản phẩm*

**7.16, Giao diện trạng thái dự án**

**Purpose**: Hiển thị trạng thái của một dự án.

**Header**: Header "Jobber" tiêu chuẩn.

**Content**:

Ngữ cảnh danh mục dịch vụ.

Thông tin về trạng thái dự án.

**Footer**: Footer tiêu chuẩn.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 7.16 – Wireframe trạng thái dự án*