**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**



**Quản Lý Tái Chế**

***Nhóm 08 - Sinh viên thực hiện***

1. Trần Đình Phong – 20036451
2. Ngô Đình Bảo Yến - 20051981

# MỤC LỤC

[MỤC LỤC 1](#_Toc164786500)

[DANH MỤC CÁC HÌNH VẼ 3](#_Toc164786501)

[DANH MỤC CÁC BẢNG BIỂU 4](#_Toc164786502)

[CHƯƠNG 1 : GIỚI THIỆU 5](#_Toc164786503)

[1.1 Tổng quan 5](#_Toc164786504)

[1.2 Mục tiêu đề tài 5](#_Toc164786505)

[1.3 Phạm vi đề tài 6](#_Toc164786506)

[1.3.1.1 Chức năng dành cho người dùng: 6](#_Toc164786507)

[1.3.1.2 *Chức năng dành cho người quản trị*: 6](#_Toc164786508)

[1.4 Mô tả yêu cầu chức năng 6](#_Toc164786509)

[1.4.1.1 Phân tích yêu cầu của hệ thống 6](#_Toc164786510)

[1.4.1.2 *Mô tả hoạt động của hệ thống* 7](#_Toc164786511)

[CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT 9](#_Toc164786512)

[2.1 Node.js 9](#_Toc164786513)

[2.2 React.js 10](#_Toc164786514)

[2.3 MongoDB 11](#_Toc164786515)

[2.4 Kiến trúc 12](#_Toc164786516)

[CHƯƠNG 3 : PHÂN TÍCH VÀ THIẾT KẾ 18](#_Toc164786517)

[3.1 Phân tích yêu cầu bằng UML 18](#_Toc164786518)

[3.1.1 Usecase tổng quát 18](#_Toc164786519)

[3.1.2 Danh sách tác nhân và mô tả 18](#_Toc164786520)

[3.1.3 Danh sách các tình huống hoạt động (Use cases) 19](#_Toc164786521)

[3.1.4 Tình huống hoạt động 21](#_Toc164786522)

[3.1.4.1 UC01\_Đăng nhập 21](#_Toc164786523)

[3.2 Class diagram 23](#_Toc164786524)

[3.3 Kiến trúc layer 23](#_Toc164786525)

[CHƯƠNG 4 : HIỆN THỰC 24](#_Toc164786526)

[4.1 Giao diện của hệ thống 24](#_Toc164786527)

[4.1.1 Giao diện web 24](#_Toc164786528)

[CHƯƠNG 5 : KẾT LUẬN 25](#_Toc164786529)

[5.1 Kết quả đạt được 25](#_Toc164786530)

[5.2 Hạn chế của đồ án 25](#_Toc164786531)

[5.3 Hướng phát triển 25](#_Toc164786532)

[TÀI LIỆU THAM KHẢO 25](#_Toc164786533)

# DANH MỤC CÁC HÌNH VẼ

[Hình 2‑1 Node.js development 1](#_Toc14693413)

# DANH MỤC CÁC BẢNG BIỂU

[Bảng 3‑1 Danh sách tác nhân và mô tả 1](#_Toc14682192)

[Bảng 3‑2 Danh sách các tình huống trong hệ thống 1](#_Toc14682193)

# : GIỚI THIỆU

## Tổng quan

Hệ thống quản lý tái chế được thiết kế để cung cấp một cách tiếp cận toàn diện và hiệu quả trong việc quản lý quy trình tái chế, từ việc nhận và đánh giá các vật liệu tái chế cho đến xử lý và báo cáo. Hệ thống này không chỉ giúp tối ưu hóa các quy trình nội bộ của các trung tâm tái chế mà còn tạo ra một môi trường thân thiện và minh bạch cho khách hàng.

## Mục tiêu đề tài

Mục tiêu của hệ thống quản lý tái chế là tạo ra một quy trình tái chế hiệu quả, minh bạch và thân thiện với người dùng. Hệ thống nhằm tối ưu hóa các quy trình từ việc nhận vật liệu, đánh giá, xử lý đến báo cáo, đảm bảo tất cả các bước đều diễn ra một cách suôn sẻ và chính xác

* Các Tính Năng Chính của Hệ Thống:

Giao Diện Người Dùng Nhận Hàng:

Xem và chỉnh sửa thông tin cá nhân.

Theo dõi trạng thái và tiến trình xử lý của vật liệu tái chế.

Giao Diện Người Dùng Tái Chế và Kế Toán:

Quản lý và tối ưu hóa các quy trình tái chế.

Quản lý tài chính và báo cáo chi tiết về các hoạt động tái chế.

Giao Diện Người Dùng Khách Hàng:

Cung cấp báo giá và thông tin trạng thái vật liệu tái chế.

Hỗ trợ khách hàng trong việc theo dõi và quản lý quy trình tái chế.c.

## Phạm vi đề tài

#### Chức năng dành cho người dùng:

* . Mua và Bán Trực Tuyến: Người dùng có thể mua và bán nhựa và chất thải điện tử thông qua cổng thông tin tái chế.
* Người dùng có thể mua và bán nhựa và chất thải điện tử thông qua cổng thông tin tái chMua và Bán Trực Tuyến: Người dùng có thể mua và bán nhựa và chất thải điện tử thông qua cổng thông tin tái chế..

#### **Chức năng dành cho người quản trị**:

* Quản lý tài khoản người dung
* Xử lý các yêu cầu của khách hàng
* Cập nhật tình trạng đơn hàng
* Báo giá sản phẩm
* Xem báo cáo thống kê

## Mô tả yêu cầu chức năng

#### Phân tích yêu cầu của hệ thống

- Người dùng có thể thay đổi thông tin cá nhân như,tên, giới tính…

- Người dùng thực hiện tải lên các loại rác thải

- Người dùng có vai trò thu gom sẽ thu lại chúng.

- Từ mô tả trên, có thể đưa ra yêu cầu của hệ thống với hai đối tượng chính tương tác với hệ thống như sau:

• Đối với hệ thống:

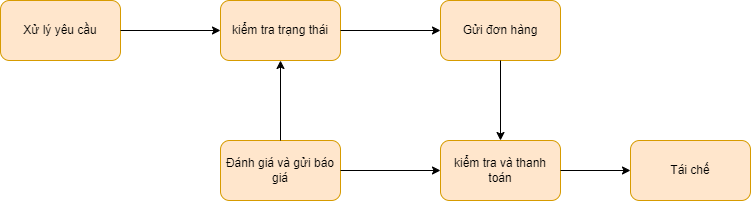
- Thay đổi trạng thái của các đơn hang.

- Quản lý thông tin của người dùng

• Đối với người dùng:

- Tải lên các loại rác thải, xem trạng thái đơn hàng, chỉnh sửa trang cá nhân

#### Mô tả hoạt động của hệ thống



*Bảng mô tả quy trình hoạt động hệ thống*

1. **Tạo Yêu Cầu:**

* Người dùng truy cập trang web và tạo yêu cầu gửi đến đơn vị bằng cách điền vào biểu mẫu trực tuyến. gồm các thông tin như loại rác thải, hình ảnh,…

1. Đánh giá sơ bộ

* Người dùng có vai trò là nhà thu gom nhận yêu cầu từ người dùng và tiến hành đánh giá sơ bộ. Quá trình này bao gồm kiểm tra thông tin yêu cầu và tình trạng thiết bị. Sau khi đánh giá, đơn vị gửi lại báo giá kèm nhận xét về yêu cầu đó.

1. **Kiểm Tra Trạng Thái:**

* Người dùng có thể kiểm tra trạng thái của yêu cầu bằng cách theo dõi trạng thái đơn hàng trên trang web

1. **Nhận Thiết Bị và Thanh Toán:**

* Đơn vị nhận thiết bị từ người dùng, tiến hành kiểm tra lại để xác nhận tình trạng và các thông tin liên quan. Sau khi kiểm tra xong, đơn vị sẽ thực hiện thanh toán cho người dùng.

# : CƠ SỞ LÝ THUYẾTTop of Form

## Node.js



Hình 2‑1 Node.js development

**Tổng quan về Node.js:**

Node.js là một nền tảng phát triển độc lập, được xây dựng dựa trên môi trường thực thi JavaScript của Google Chrome. Nền tảng này cho phép phát triển các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.

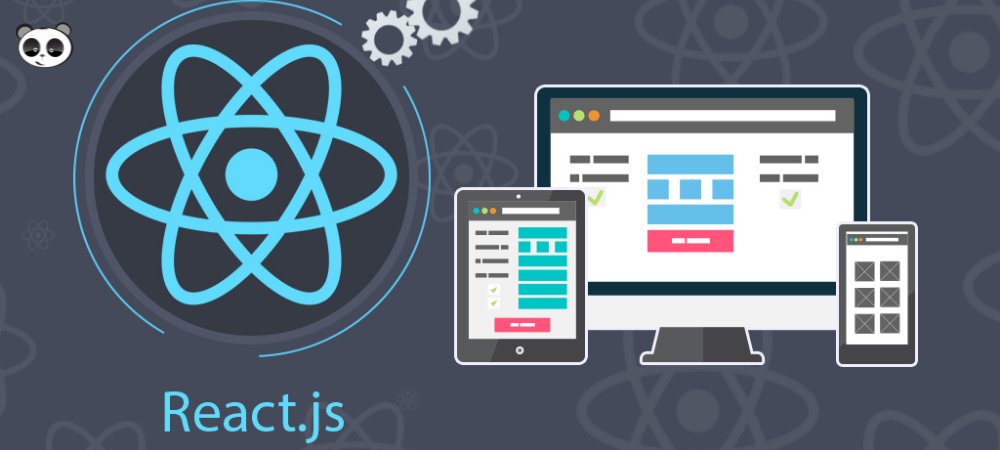
**Ưu điểm của Node.js:**

1. **Đơn ngôn ngữ:** Node.js sử dụng JavaScript, một trong những ngôn ngữ lập trình phổ biến nhất, cho phép phát triển ứng dụng đồng nhất từ client đến server.
2. **Kiến trúc hiệu quả:** Với kiến trúc hướng sự kiện và cơ chế non-blocking I/O, Node.js mang lại hiệu suất cao và khả năng xử lý đồng thời lớn, làm cho nó trở thành lựa chọn lý tưởng cho các ứng dụng cần xử lý nhiều kết nối cùng một lúc.
3. **Đa nền tảng:** Node.js có thể chạy trên nhiều hệ điều hành khác nhau như macOS, Windows và Linux, đảm bảo tính linh hoạt và dễ dàng triển khai trong môi trường phát triển đa dạng.
4. **Cộng đồng mạnh mẽ:** Có một cộng đồng lớn và tích cực hỗ trợ Node.js, với hàng ngàn mô-đun miễn phí có sẵn trên npm (Node Package Manager), giúp tăng tốc độ phát triển và cung cấp các giải pháp sẵn có cho nhiều vấn đề phức tạp.
5. **Miễn phí:** Node.js là một nền tảng mã nguồn mở, cung cấp miễn phí cho các nhà phát triển, giúp giảm chi phí phát triển phần mềm.

Nhờ vào những ưu điểm này, Node.js đã trở thành một trong những lựa chọn hàng đầu cho các nhà phát triển web và ứng dụng di động trên toàn thế giới.

Top of Form

## React.js



**Tổng quan về React:**

React là một thư viện JavaScript được sử dụng để xây dựng giao diện người dùng tương tác. Nó khuyến khích việc tạo ra các thành phần giao diện có thể tái sử dụng, và cho phép hiển thị dữ liệu thay đổi theo thời gian. Rất nhiều người sử dụng React làm phần giao diện người dùng (View) trong mô hình MVC (Model-View-Controller).

**Ưu điểm của React:**

1. **Sử dụng DOM ảo:** React sử dụng một cấu trúc dữ liệu gọi là DOM ảo (Virtual DOM) là một đối tượng JavaScript. Điều này cải thiện hiệu suất ứng dụng, vì DOM ảo JavaScript thường nhanh hơn DOM thực sự và giúp tránh trạng thái đồng bộ hóa tốn kém.
2. **Có thể sử dụng trên máy khách và máy chủ:** React có thể được sử dụng không chỉ trên phía máy khách (client-side) mà còn trên phía máy chủ (server-side), cung cấp khả năng rendering đa nền tảng và tối ưu hóa SEO.
3. **Các mẫu thành phần và dữ liệu cải thiện khả năng đọc:** React thúc đẩy việc sử dụng các mẫu thành phần (component-based) và quản lý dữ liệu một cách hiệu quả, giúp duy trì và phát triển các ứng dụng lớn một cách dễ dàng và hợp lý hơn.

Nhờ vào những ưu điểm này, React đã trở thành một trong những công cụ phát triển giao diện người dùng phổ biến nhất hiện nay, được sử dụng rộng rãi trong cả ứng dụng web và di động.

Top of Form

## MongoDB



**Tổng quan về MongoDB:**

MongoDB là một hệ thống quản lý cơ sở dữ liệu (DBMS) mã nguồn mở, được thiết kế với mô hình cơ sở dữ liệu hướng đối tượng. Đặc điểm nổi bật của MongoDB là khả năng lưu trữ dữ liệu một cách linh hoạt mà không yêu cầu phải tuân theo một cấu trúc nhất định.

**Ưu điểm của MongoDB:**

1. **Hiệu suất cao:** MongoDB được tối ưu hóa để đạt hiệu suất cao trong việc truy xuất và xử lý dữ liệu.
2. **Tính sẵn sàng cao - Nhân rộng:** MongoDB hỗ trợ tính năng nhân rộng tự động, giúp cân bằng tải và tăng khả năng chịu tải của hệ thống một cách linh hoạt.
3. **Khả năng mở rộng cao - Sharding:** MongoDB cho phép phân chia dữ liệu thành các phân đoạn nhỏ hơn (shards), giúp phân tải dữ liệu và cải thiện hiệu suất của hệ thống.
4. **Năng động:** MongoDB không yêu cầu một lược đồ cứng nhắc, cho phép thêm, xóa trường dữ liệu một cách linh hoạt mà không ảnh hưởng đến ứng dụng.
5. **Linh hoạt:** MongoDB cho phép dữ liệu được biểu diễn dưới dạng JSON hoặc BSON, đồng thời hỗ trợ các truy vấn phong phú và phức tạp.
6. **Dữ liệu không đồng nhất:** MongoDB cho phép lưu trữ dữ liệu không đồng nhất, tức là các bản ghi trong cùng một bảng không cần phải có cấu trúc giống nhau.
7. **Không joins:** MongoDB hạn chế việc sử dụng joins trong truy vấn, thay vào đó, nó sử dụng mô hình nhúng dữ liệu để tối ưu hóa hiệu suất.
8. **Phân phối được:** MongoDB có thể được triển khai trên nhiều nền tảng và môi trường, từ máy tính cá nhân đến các môi trường điện toán đám mây.

Nhờ vào những ưu điểm này, MongoDB trở thành một lựa chọn phổ biến cho các ứng dụng cần xử lý dữ liệu lớn và có tính linh hoạt cao.

## Kiến trúc MVC( Áp dụng vào project Nodejs)

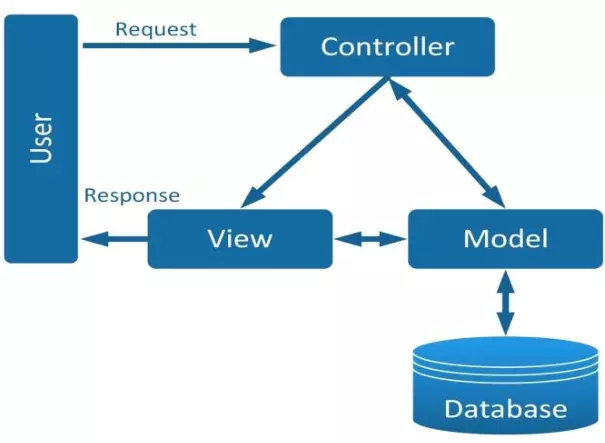
1. **Kiến trúc mvc là gì?**

**Mô hình mvc**  Kiến trúc MVC, viết tắt của Model-View-Controller, là một mô hình thiết kế phần mềm thường được sử dụng để phát triển giao diện người dùng, phân chia logic chương trình liên quan thành ba thành phần tương tác với nhau. Dưới đây là định nghĩa chi tiết của từng thành phần:Mô hình này phát huy hiệu quả nhất khi bạn xây dựng một hệ thống lớn, việc quản lý code và xử lý dữ liệu lỗi dễ dàng hơn.

* Model (Mô hình): Đại diện cho dữ liệu nội bộ và quy tắc kinh doanh của ứng dụng. Model chứa các thông tin cần thiết để biểu diễn dữ liệu đến người dùng và xử lý dữ liệu, bao gồm cả việc truy xuất và lưu trữ dữ liệu từ cơ sở dữ liệu.
* Model (Mô hình): Đại diện cho dữ liệu nội bộ và quy tắc kinh doanh của ứng dụng. Model chứa các thông tin cần thiết để biểu diễn dữ liệu đến người dùng và xử lý dữ liệu, bao gồm cả việc truy xuất và lưu trữ dữ liệu từ cơ sở dữ liệu.
* Model (Mô hình): Đại diện cho dữ liệu nội bộ và quy tắc kinh doanh của ứng dụng. Model chứa các thông tin cần thiết để biểu diễn dữ liệu đến người dùng và xử lý dữ liệu, bao gồm cả việc truy xuất và lưu trữ dữ liệu từ cơ sở dữ liệu.

1. Ưu điểm:

* Phân Tách Rõ Ràng: MVC cung cấp sự phân tách rõ ràng giữa logic nghiệp vụ, giao diện người dùng (UI), và logic đầu vào, giúp dễ dàng quản lý và bảo trì mã nguồn
* Dễ Dàng Quản Lý và Mở Rộng: Các thành phần Model, View, và Controller có thể được phát triển và thử nghiệm một cách độc lập, làm cho việc quản lý và mở rộng ứng dụng trở nên dễ dàng hơn
* Tối Ưu Băng Thông: MVC không tiêu tốn nhiều viewstate, giúp tiết kiệm băng thông và tăng tốc độ truy cập
* Hỗ Trợ Phát Triển Dựa Trên Thử Nghiệm: Kiến trúc MVC hỗ trợ phương pháp phát triển dựa trên thử nghiệm (Test-driven Development), giúp phát hiện và sửa lỗi dễ dàng hơn
* Kiểm Soát HTML và URL: MVC cung cấp toàn quyền kiểm soát HTML và URL, giúp thiết kế kiến trúc ứng dụng web dễ dàng và có thể tạo ra các URL dễ hiểu và có thể tìm kiếm được
* Tương Thích Đa Nền Tảng: MVC tối ưu hiệu quả trên đa dạng nền tảng ngôn ngữ lập trình, từ web đến di động



1. **Áp dụng kiến trúc mvc Architecture vào project NodeJS**
   1. **Các vấn đề gặp phải khi sử dụng nền tảng NodeJS**
2. Các framework nodejs phổ biết như Express cho phép chúng ta dễ dàng tạo ra Resful API xử lí các request từ phía client một cách nhanh chóng và linh hoạt. Tuy nhiên, Express lại không có một kiến trúc hay một quy tắc tổ chức codde một cách cụ thể. Điều này sẽ là một vấn đề lớn khi dự án của bạn được scale với nhiều chức năng cũng như logic nghiệp vụ ngày các phức tạp. Tình trạng này sẽ gây ra một số vấn đề:

* Rối code và trùng lặp code
* Khó quản lý và maintain
* Khó khăn trong việc mở rộng cũng như tính ổn định
* Cấu trúc tệ khiến khó khăn hoặc không thể viết unit-test

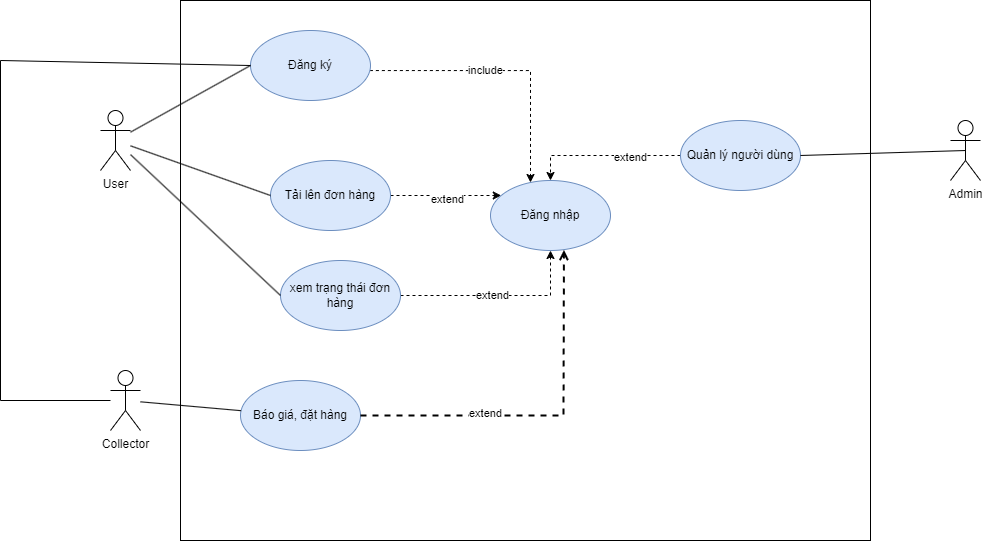
1. Việc Không tổ chức tốt cấu trúc dự án sau này sẽ dẫn đến phát sinh chi phí cũng như rủi do trong quá trình update cũng như maintain.
   1. **Áp dụng kiến trúc mvc**

* **Model:** Chứa dữ liệu và logic nghiệp vụ, quản lý thông tin về các yêu cầu tái chế và trạng thái của chúng.
* **View:** Cung cấp giao diện người dùng cho cả người dùng và người quản lý, hiển thị thông tin về yêu cầu tái chế và trạng thái của chúng.
* **Controller:** Xử lý các yêu cầu từ người dùng và người quản lý, điều phối giữa View và Model để cập nhật và truy xuất dữ liệu
* **Trường hợp nên dùng kiến trúc 3 lớp**
* Ứng dụng có nhiều lập trình viên: MVC hỗ trợ tốt cho việc phát triển ứng dụng bởi các đội ngũ lập trình viên lớn, giúp quản lý tính năng và sự phức tạp của ứng dụng
* Ứng dụng cần mô hình phát triển hướng kiểm thử (TDD): MVC tạo điều kiện thuận lợi cho việc lập trình dựa trên thử nghiệm, giúp kiểm tra và bảo đảm chất lượng mã nguồn
* Ứng dụng web và di động: MVC phổ biến trong thiết kế ứng dụng web và di động nhờ khả năng mở rộng và dễ bảo trì
* Cần sự phân tách rõ ràng: Khi ứng dụng cần sự tách biệt giữa logic nghiệp vụ, giao diện người dùng, và xử lý dữ liệu, MVC cung cấp sự phân tách này, giúp dễ dàng quản lý và nâng cấp
* Ứng dụng có URL dễ hiểu: MVC hỗ trợ xây dựng các ứng dụng có URL dễ hiểu và có thể tìm kiếm được, tối ưu hóa cho SEO
* Dễ kiểm tra và điều chỉnh: MVC giúp việc kiểm tra và điều chỉnh theo yêu cầu của khách hàng trở nên dễ dàng hơn
  1. **Trường hợp không nên dùng kiến trúc mvc**
* **Ứng dụng đơn giản:** Đối với các ứng dụng nhỏ và đơn giản, việc sử dụng MVC có thể làm tăng độ phức tạp không cần thiết và gây khó khăn trong việc bảo trì
* Dữ liệu liên kết chặt chẽ: Nếu ứng dụng của bạn có dữ liệu được liên kết chặt chẽ và không thể chia dữ liệu ra thành các database chạy độc lập, MVC có thể không phù hợp
* Ứng dụng cần tương tác phức tạp: MVC có thể không phù hợp với các ứng dụng có yêu cầu về trạng thái hoặc tương tác phức tạp, vì bộ điều khiển có thể trở thành nút thắt cổ chai
* Hiệu suất: Trong một số trường hợp, MVC có thể không đáp ứng được yêu cầu về hiệu suất, đặc biệt là với các ứng dụng cần xử lý nhanh chóng và hiệu quả
  1. **Khả năng kiểm thử**
* Khả năng kiểm thử của mô hình MVC trong phát triển phần mềm được đánh giá cao vì nó hỗ trợ việc phân chia rõ ràng các phần của ứng dụng, giúp việc kiểm thử trở nên dễ dàng và hiệu quả hơn. Dưới đây là một số điểm nổi bật về khả năng kiểm thử trong mô hình MVC:
* Kiểm thử độc lập: Các thành phần trong MVC có thể được kiểm thử một cách độc lập, giúp tìm ra lỗi một cách nhanh chóng và chính xác
* Kiểm thử dễ dàng: Tầng View và Controller có thể được mock hoặc stub để kiểm thử logic nghiệp vụ trong Model mà không cần phụ thuộc vào giao diện người dùng hoặc cơ sở dữ liệu
* Kiểm thử tự động: MVC phù hợp với việc áp dụng các công cụ và kỹ thuật kiểm thử tự động, giúp tiết kiệm thời gian và nguồn lực
* Kiểm thử tích hợp: Kiểm thử tích hợp giữa các thành phần trong MVC giúp đảm bảo rằng các thành phần làm việc cùng nhau một cách chính xác2
  1. **Hiệu năng**
* **Kiến trúc MVC (Model-View-Controller)** được đánh giá cao về khả năng cung cấp hiệu năng tốt cho ứng dụng, nhất là trong các ứng dụng web và di động. Dưới đây là một số lý do tại sao MVC có thể cải thiện hiệu năng của ứng dụng:
* Phân tách rõ ràng: MVC tạo ra sự phân tách rõ ràng giữa logic nghiệp vụ, giao diện người dùng và xử lý dữ liệu, giúp dễ dàng quản lý và tối ưu hóa từng phần
  1. **Khả năng mở rộng**
* . iến trúc MVC (Model-View-Controller) được đánh giá cao về khả năng mở rộng. Dưới đây là một số lý do tại sao MVC hỗ trợ khả năng mở rộng tốt:
* Phân tách rõ ràng: MVC phân chia ứng dụng thành ba thành phần chính (Model, View, và Controller), giúp dễ dàng quản lý và mở rộng từng phần một cách độc lập1.
* Tái sử dụng và mở rộng dễ dàng: Các thành phần trong MVC có thể được tái sử dụng và mở rộng, giúp tiết kiệm thời gian và công sức khi phát triển các tính năng mới hoặc cải thiện các tính năng hiện có2.
* Độc lập giữa các thành phần: Do sự độc lập giữa các thành phần, việc thay đổi hoặc nâng cấp một phần của hệ thống không ảnh hưởng đến các phần khác

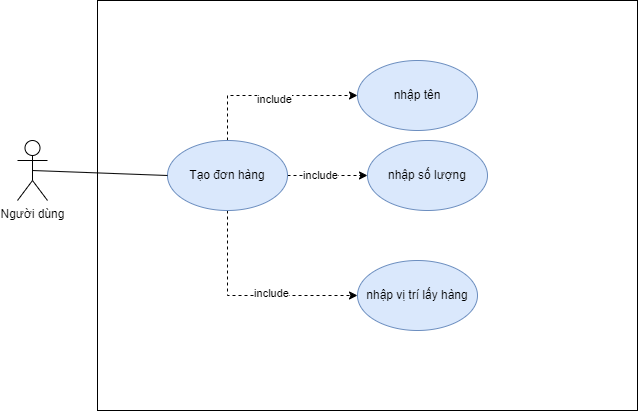
# : PHÂN TÍCH VÀ THIẾT KẾ

## Phân tích yêu cầu bằng UML

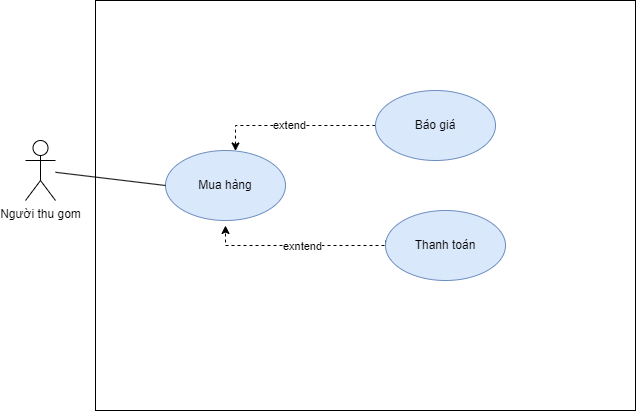
### Usecase tổng quát



Hình 3.1 Usecase tổng quát



Hình 3.2 Usecase Tạo đơn hàng



Hình 3.3 Usecase Đặt hàng

### Danh sách tác nhân và mô tả

Danh sách tác nhân và mô tả xem ở Bảng 3-1.

|  |  |
| --- | --- |
| Tác nhân | Mô tả tác nhân |
| User người bán | * Đăng nhập * Đăng kí * Tạo đơn bán * Xem trạng thái đơn hàng |
|  |
|  |
|  |
| User người mua | * Đăng nhập * Đăng kí * Báo giá * Thanh toán |

Bảng 3‑1 Danh sách tác nhân và mô tả

### Mô tả UseCase và hoạt động

|  |  |
| --- | --- |
| * **Tên use case**: UC\_Đăng nhập | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: User | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đã có tài khoản | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn chức năng đăng nhập | 1. Hệ thống hiển thị giao diện nhập thông tin tài khoản |
| 1. Người dùng nhập tài khoản gồm email và mật khẩu |  |
| 1. Chọn nút đăng nhập | 1. Hệ thống kiểm tra thông tin nhập vào |
|  | 1. Hệ thống chuyển hướng người dùng vào giao diện trang chủ. |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 6.1. Hệ thống thông báo thông tin tài khoản đăng nhập không đúng |
|  | 6.2. Hệ thống quay lại bước 3 |

Bảng 3‑2 Danh sách các tình huống trong hệ thống

Activity diagram:

A diagram of a diagram

Description automatically generated with medium confidence

Sequence diagram:

A diagram of a computer program

Description automatically generated

#### 3.1.4.2 UC02\_Đăng xuất

|  |  |
| --- | --- |
| * **Tên use case**: UC02\_Đăng xuất | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: người dùng | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đã đăng nhập vào tài khoản | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn chức năng đăng xuất | 2. Hệ thống hiển thị xác nhận muốn thoát |
| 1. Người dùng ấn xác nhận | 1. Hệ thống trả về giao diện đăng nhập |
| * **Luồng sự kiện thay thế (alternate flow):** | |
| **Không có** | |

A diagram of a system

Description automatically generated

#### 3.1.4.10 UC10 \_Đăng kí tài khoản

|  |  |
| --- | --- |
| * **Tên use case**: UC10 \_Đăng kí tài khoản | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: Người dùng | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng vào website trường | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn chức năng đăng kí tài khoản | 1. Hệ thống hiển thị form đăng kí tài khoản |
| 1. Người dùng nhập vào form đăng kí | 1. Hệ thống kiểm tra thông tin người dung điền vào |
|  | 1. Hệ thống thông báo đăng kí tài khoản thành công |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 4.a.1Hệ thống thông báo nhập sai tên đăng nhập( phải bắt đầu bằng chữ và kêt thúc là @gmail.com)  4.a.2.Quay lại form đăng kí |
|  | 4.a.1.Hệ thống thông báo mật khẩu không được nhập kí tự đặc biệt  4.a.2. Quay lại form đăng kí |
|  |  |

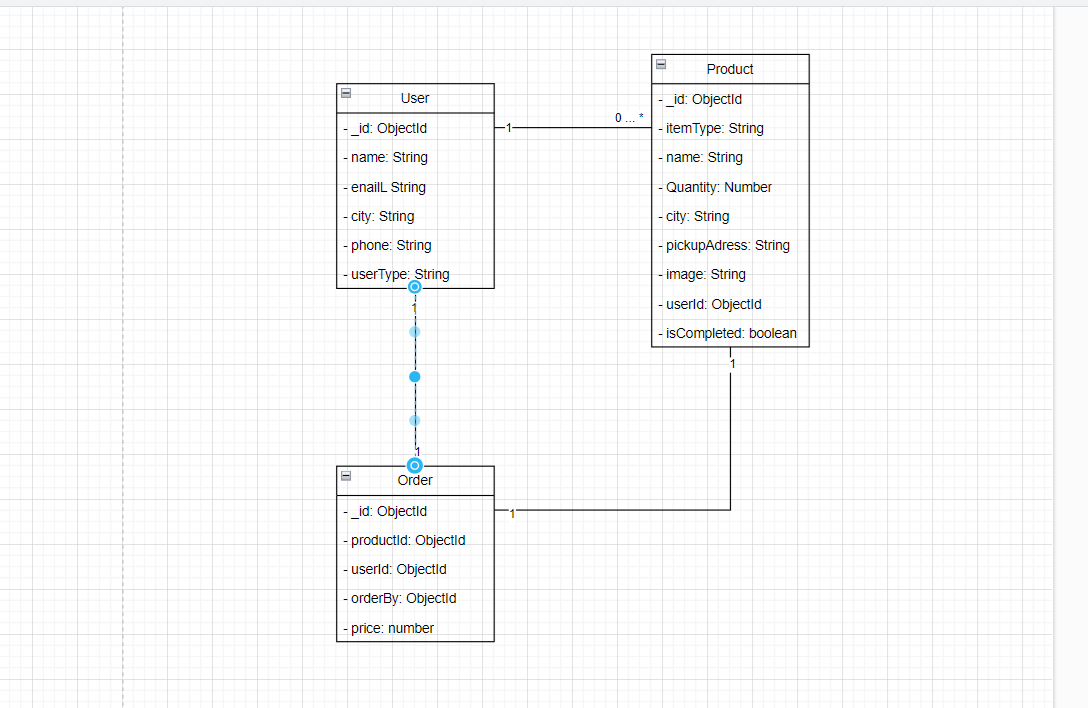
A diagram of a system

Description automatically generated

A diagram of a diagram

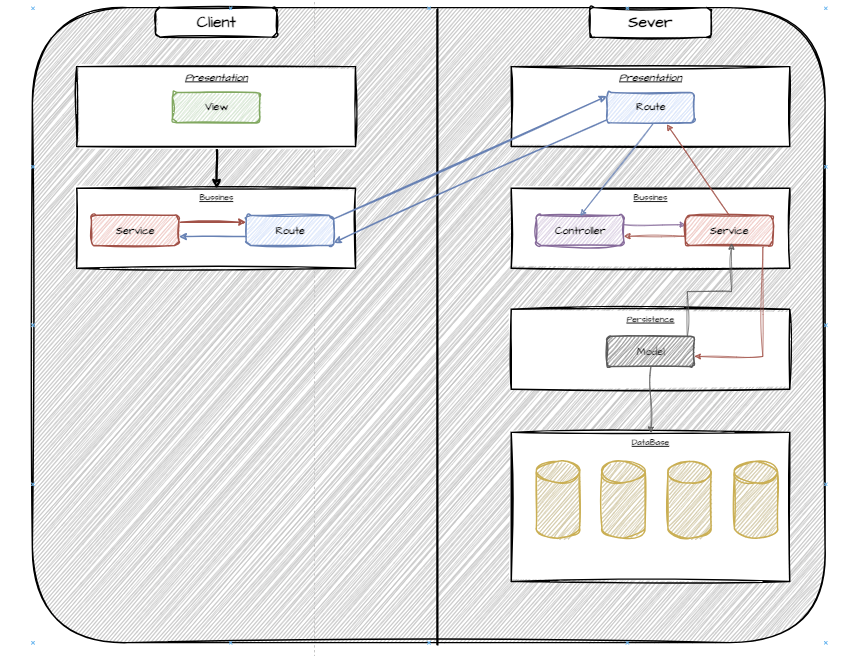
Description automatically generated

### Class diagram



## Kiến trúc *mvc architecture*

Kiến trúc sẽ có 2 phần là Client và Sever, ở trong mỗi phần sẽ có sự phân tầng khác nhau



* **Presentation Layer**

Chức năng:

* Giao diện người dùng: Hiển thị dữ liệu và cung cấp giao diện cho người dùng tương tác với ứng dụng.
* Xử lý đầu vào người dùng: Nhận và xử lý các thao tác từ người dùng như nhấp chuột, nhập liệu, và các sự kiện giao diện khác.
* Chuyển đổi dữ liệu: Chuyển đổi dữ liệu từ tầng business thành dạng có thể hiển thị và ngược lại.

Thành phần chính:

* **Business Layer (Tầng nghiệp vụ)**

Chức năng:

* Xử lý logic nghiệp vụ: Thực hiện các quy tắc và xử lý logic nghiệp vụ của ứng dụng.
* Điều phối dữ liệu: Điều phối và xử lý dữ liệu giữa presentation layer và persistence layer.
* Thực hiện các tác vụ nghiệp vụ: Thực hiện các tác vụ cụ thể của ứng dụng như tính toán, kiểm tra điều kiện, và các quy trình nghiệp vụ khác.

Thành phần chính:

* Client Side (Service): Xử lý logic nghiệp vụ trên client và gọi API tới server.
* Server Side (Controller, Service): Xử lý yêu cầu từ router, thực hiện logic nghiệp vụ và tương tác với tầng persistence.
* **Persistence Layer (Tầng lưu trữ)**

Chức năng:

* + - Quản lý dữ liệu: Xử lý lưu trữ và truy xuất dữ liệu từ cơ sở dữ liệu.
    - Thao tác CRUD: Thực hiện các thao tác tạo, đọc, cập nhật, và xóa dữ liệu.
    - Quản lý kết nối dữ liệu: Quản lý kết nối tới cơ sở dữ liệu và tối ưu hóa truy vấn.

Thành phần chính:

* + - Model: Tương tác với cơ sở dữ liệu và quản lý các thao tác dữ liệu.
* **Database Layer (Tầng cơ sở dữ liệu)**

Chức năng:

* Lưu trữ dữ liệu: Lưu trữ tất cả dữ liệu của ứng dụng.
* Quản lý dữ liệu: Cung cấp các chức năng quản lý và truy xuất dữ liệu một cách hiệu quả.

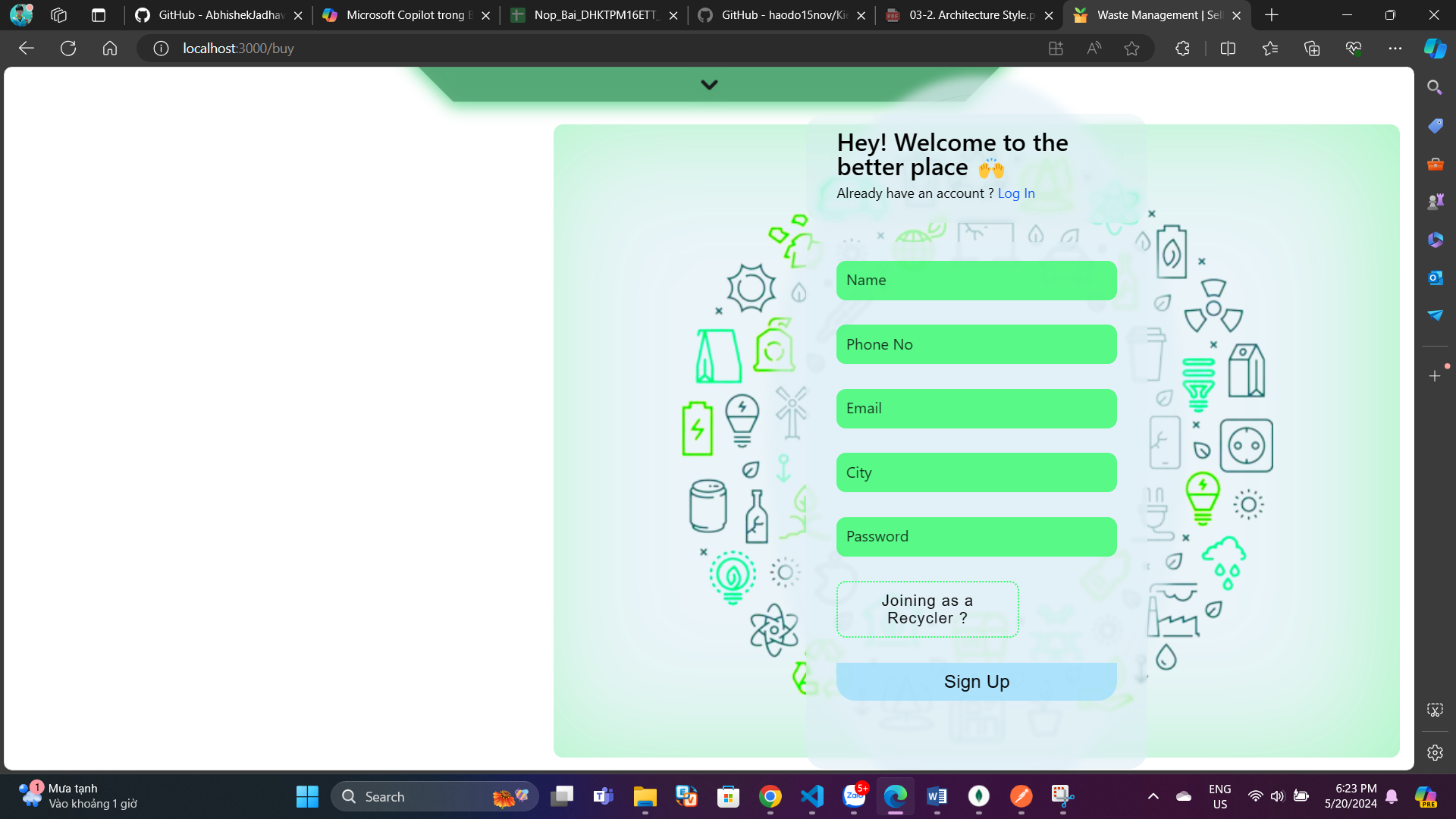
Thành phần chính:

Database: Cơ sở dữ liệu MongoDB.

# : HIỆN THỰC

## Giao diện của hệ thống

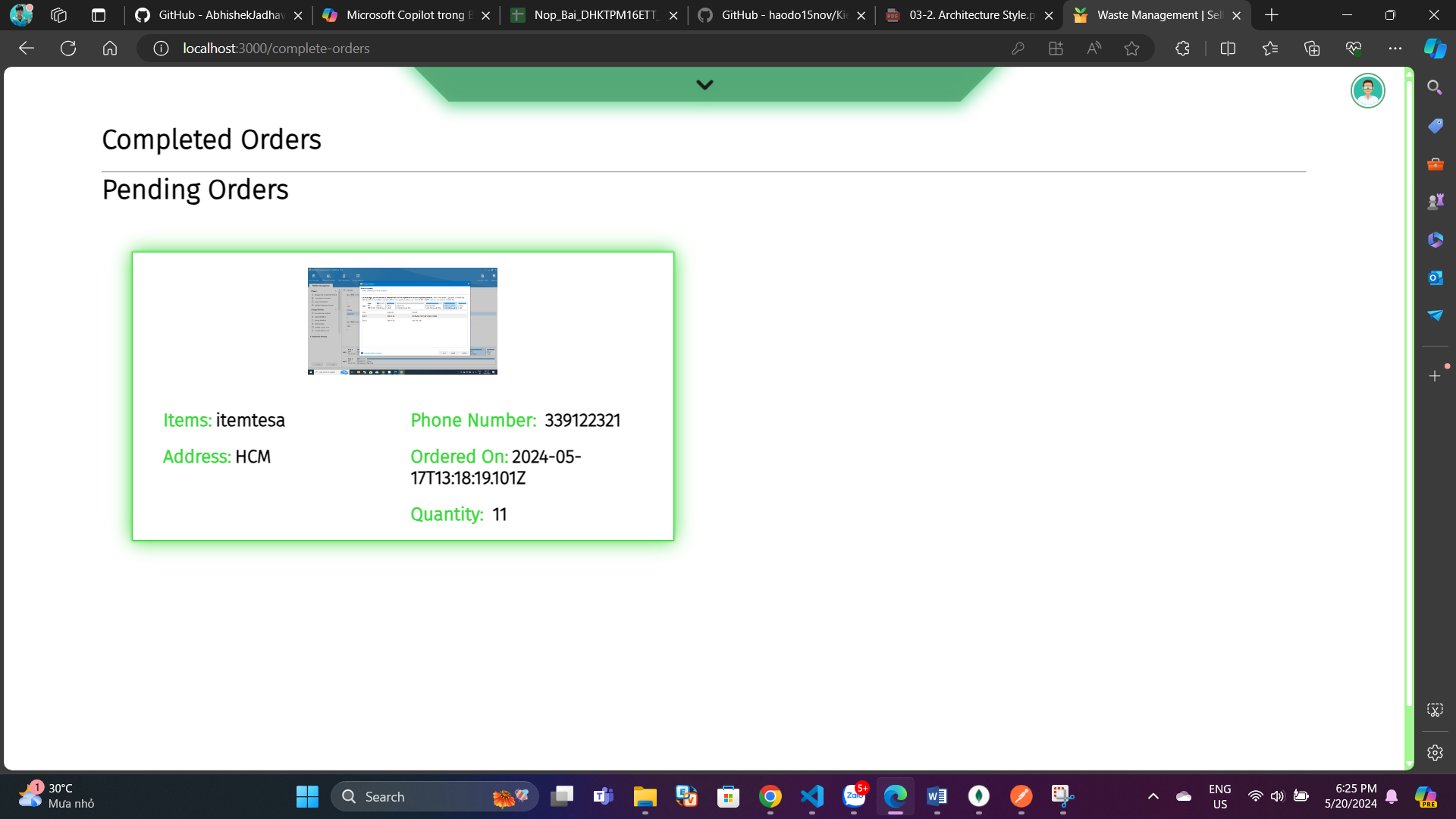
### Giao diện web



Hình 4.1 giao diện đăng Ký



Hình 4.3 giao diện Trang Chủ



### : KẾT LUẬN

## Kết quả đạt được

* Người dùng:
* Hệ thống dịch vụ tái chế: Xây dựng thành công một hệ thống cho phép người dùng gửi yêu cầu tái chế thiết bị điện tử cũ.
* Chức năng gửi yêu cầu: Người dùng có thể gửi yêu cầu tái chế thông qua giao diện trực tuyến.
* Xem yêu cầu: Người dùng có thể xem các yêu cầu họ đã gửi.

Xem trạng thái yêu cầu: Người dùng có thể theo dõi trạng thái của yêu cầu tái chế của họ.

* Hệ thống
* Xử lý yêu cầu: Người quản lý có thể xem và xử lý các yêu cầu tái chế từ người dùng.
* Báo giá: Người thu gom có thể đưa ra báo giá cho việc tái chế dựa trên thông tin của yêu cầu.
* Xác nhận tiêu huỷ hoặc tái chế: Người quản lý có thể xác nhận việc tiêu huỷ hoặc tái chế các thiết bị dựa trên yêu cầu đã được xử lý.

## Hạn chế của đồ án

* Một số chức năng vẫn chưa thực sự hoàn thiện như

## Hướng phát triển

* Cố gắng khắc phục nhữn hạn chế và sửa đổi hệ thống sao cho phù hợp

## TÀI LIỆU THAM KHẢO

Các tài liệu Tiếng Anh

1. Fundamentals of Software Architecture - Mark Richards
2. Software Architecture Monday - Mark Richards

Các tài liệu từ Internet

1. <https://nodejs.org/en/docs/>
2. <https://expressjs.com/en/guide/routing.html>
3. <https://www.mongodb.com/docs/manual/reference/operator/aggregation/graphLookup/>
4. <https://topdev.vn/blog/kien-truc-phan-lop-layered-architecture/?gidzl=9GbQV5FuxoSW5bv-LkBj9o4HMdb8vBnnQqK4B4-oioSnHWHw5-2y8ZKP3Yr2whvoOn8FA3D7m81kKFZY90>
6. [Software Architecture Patterns](Software%20Architecture%20Patterns)