 **BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM  
KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**MÔN HỌC: NHẬP MÔN LẬP TRÌNH PYTHON CHO PHÂN TÍCH**

**ĐỀ TÀI:**

**XÂY DỰNG WEBSITE QUẢN LÝ QUÁN CÀ PHÊ SỬ DỤNG FRAMEWORK DJANGO**

**Mã môn học: 212IPPA233277**

**GVHD: Ths. Nguyễn Văn Thành**

**Nhóm thực hiện: Nhóm 05**

***TP. Hồ Hồ Chí Minh, tháng 05, năm 2022***

DANH SÁCH THÀNH VIÊN THỰC HIỆN ĐỒ ÁN CUỐI KỲHỌC KỲ II, NĂM HỌC 2021-2022Nhóm 05. Lớp Python 03 - Chiều thứ sáu - tiết 7-10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Phân công nhiệm vụ** | **Mức độ hoàn thành** |
| 1 | Huỳnh Tiến Dĩ | 20110246 | * Xây dựng app orders: hiển thị sản phẩm; lưu, thanh toán hóa đơn | 100% |
| 2 | Trần Văn Quảng | 20110160 | * Tạo khung dự án * Thiết kế giao diện * Hỗ trợ JavaScript * Xây dựng app core: đăng nhập, đăng xuất * Xây dựng app tables: thêm, sửa, xóa, gộp/tách bàn * Hỗ trợ app foods: lọc danh sách món * Hỗ trợ xây dựng app manager: vẽ bảng thống kê, bán hàng | 100% |
| 3 | Nguyễn Đình Thiện | 20110263 | * Xây dựng app foods: thêm sửa xóa danh mục/món | 100% |
| 4 | Hoàng Hạ Vũ | 20110750 | * Xây dựng app manager: doanh số, thông tin cửa hàng, quản lý tài khoản | 100% |

*Ghi chú:*

* Nhóm trưởng: Trần Văn Quảng

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NHẬN XÉT CỦA GIÁO VIÊN**

*Tháng 05 năm 2022*

***Giáo viên chấm điểm***

**LỜI CẢM ƠN**

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành đến ***Ths.*** ***Nguyễn Văn Thành*** - giảng viên bộ môn ***Nhập môn Lập trình Python cho phân tích*** của chúng em. Trong quá trình học tập và nghiên cứu về bộ môn, thầy đã trực tiếp hướng dẫn chúng em từ cách chọn đề tài phù hợp, đến việc hỗ trợ, giúp đỡ tận tình để khắc phục những khó khăn, thiếu sót trong lúc hoàn thành đồ án cuối kỳ, thầy đã tạo điều kiện thuận lợi để nhóm hoàn thành đồ án một cách hoàn thiện và đạt hiệu quả tốt nhất. Xin gửi lời cảm ơn đến các bạn trong lớp đã tận tình trao đổi, hỗ trợ, chia sẻ nhóm những thông tin cần thiết mà nhóm thiếu sót trong quá trình thực hiện.

Với những kiến thức tiếp thu từ bài giảng, sự chỉ dẫn tận tình của giảng viên cùng với quá trình tự trau dồi hiểu biết của bản thân, nhóm chúng em đã cố gắng hoàn thành đồ án một cách tốt nhất. Quá trình thực hiện không thể tránh những thiếu sót, kính mong thầy đóng góp ý kiến để nhóm chúng em hoàn thiện, nâng cao hiểu biết của bản thân, củng cố lại đồ án với kết quả tốt hơn.

Kính chúc thầy sức khỏe và hạnh phúc, thành công trong con đường giảng dạy của mình!

Chúng em xin chân thành cảm ơn!

*Thành phố Hồ Chí Minh, ngày 20 tháng 05 năm 2022*

*Đại diện nhóm 05*

***Trần Văn Quảng***

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN 1](#_Toc104883809)

[1.1. Tình hình nghiên cứu 1](#_Toc104883810)

[1.2. Lý do chọn đề tài 1](#_Toc104883811)

[1.3. Mục tiêu nghiên cứu 1](#_Toc104883812)

[1.4. Phạm vi nghiên cứu 1](#_Toc104883813)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 2](#_Toc104883814)

[2.1. Framework DJANGO 2](#_Toc104883815)

[2.1.1. Django là gì? 2](#_Toc104883816)

[2.1.2. Tại sao nên sử dụng Django 2](#_Toc104883817)

[2.1.3. Mô hình MVT trong Django 2](#_Toc104883818)

[2.1.4. Phát triển ứng dụng Web với Django 3](#_Toc104883819)

[2.2. HTML, CSS 9](#_Toc104883820)

[2.2.1. HTML 9](#_Toc104883821)

[2.2.2. CSS 10](#_Toc104883822)

[2.3. JavaScript 11](#_Toc104883823)

[2.4. SQLite 11](#_Toc104883824)

[CHƯƠNG 3: GIỚI THIỆU DỰ ÁN XÂY DỰNG WEBSITE QUẢN LÝ QUÁN CÀ PHÊ 12](#_Toc104883825)

[3.1. Cơ sở dữ liệu 12](#_Toc104883826)

[3.2. Demo và phân tích các chức năng chính của dự án: 12](#_Toc104883827)

[3.2.1. Chức năng chính 12](#_Toc104883828)

[3.2.2. Cấu trúc dự án và demo từng chức năng 13](#_Toc104883829)

[CHƯƠNG 4: KẾT LUẬN 43](#_Toc104883830)

[4.1. Kết quả đạt được 43](#_Toc104883831)

[4.2. Hạn chế của dự án 43](#_Toc104883832)

[4.3. Định hướng phát triển của đề tài 43](#_Toc104883833)

[4.3.1. Hình thức và chức năng 43](#_Toc104883834)

[4.3.2. Quy mô 43](#_Toc104883835)

[TÀI LIỆU THAM KHẢO 44](#_Toc104883836)

# CHƯƠNG 1: TỔNG QUAN

## Tình hình nghiên cứu

Trong cuộc sống hiện đại ngày nay, không khó để có thể tìm được một quán nước, quán cà phê để phục vụ không chỉ cho nhu cầu giải khác mà còn là nơi để gặp gỡ cùng gia đình, bạn bè, người thân, hay là việc học nhóm tại những quán cà phê yên tĩnh, thoáng mát cũng không còn xa lạ đối với giới trẻ ngày nay. Để đáp ứng được những nhu cầu đó, việc những quán cà phê dần dần mọc lên ở khắp các phố phường, với đủ các thể loại, đủ hình thức để thu hút khách hàng.

## Lý do chọn đề tài

Việc kinh doanh một quán cà phê đòi hỏi cần phải có nhiều yếu tố như: mặt bằng, thiết kế, nguồn nguyên liệu, thuê nhân viên, ... Bên cạnh đó thì việc quản lý quán cà phê của mình như thế nào cũng là một bài toán nan giải. Chúng ta sẽ quản lý quản lý quán cà phê như thế nào cho hiệu quả, tiện lợi, ít tốn công sức nhất?

Cùng với sự phát triển của công nghệ ngày nay, việc ra đời những phần mềm giúp quản lý việc bàn hàng để đáp ứng cho việc kinh doanh và giúp tiết kiệm thời gian, công sức, làm cho việc quản lý hiệu quả hơn là một điều thiết yếu.

Với những kiến thức đã có cùng với những kiến thức được tiếp thu trong môn học này, nhóm chúng em quyết định chọn đề tài ***Xây dựng Website Quản lý quán Cà phê.***

## Mục tiêu nghiên cứu

Website sẽ hỗ trợ quản lý các nghiệp vụ như gọi món, thanh toán, quản lý thực đơn, hóa đơn, thống kê, … một cách dễ dàng hơn. Nó giúp cho người dùng tiết kiệm được thời gian, thao tác dễ dàng, nhanh chóng.

## Phạm vi nghiên cứu

* Website được được xây dựng để sử dụng cho một quán cà phê.
* Website được đáp ứng cho người quản lý và nhân viên thao tác.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## Framework DJANGO

### Django là gì?

Django là một framework cấp cao được viết bằng Python, được thiết kế để đáp ứng nhu cầu phát triển các website an toàn và dễ bảo trì.

Bởi vì Framework này được xây dựng bởi các nhà phát triển có kinh nghiệm nên nó có khả năng xử lý hầu hết những phần phức tạp của việc phát triển web.

Do đó, bạn có thể tập trung vào việc viết app mà không cần phải dừng lại để kiểm tra. Django miễn phí và là open-source.[[1]](#footnote-1)

### Tại sao nên sử dụng Django

Một số ưu điểm khi sử dụng Django:

* ***Đơn giản và nhanh chóng:*** Django được thiết kế để giúp các nhà phát triển đưa ứng dụng từ ý tưởng đến hoàn thiện nhanh nhất có thể.
* ***Có đầy đủ thư viện/module cần thiết:*** Django có sẵn các thư viện về user authentication, content admin, site maps, …
* ***Bảo mật:*** Django rất coi trọng vấn đề bảo mật và giúp các nhà phát triển tránh được các vấn đề bảo mật phổ biến hiện này.
* ***Dễ dàng mở rộng:*** Django có thể đáp ứng được lượng traffics lớn.

Ngoài ra Django có một cộng đồng người dùng vô cùng lớn với nhiều diễn đàn, kênh và các trang web để bạn có thể dễ dàng nhận được sự giúp đỡ khi gặp bất cứ vấn đề gì liên quan đến Django.

### Mô hình MVT trong Django

MVT (*Model View Template*): Django sử dụng 3 khái niệm chính đó là Model, View và Template (sẽ được đề cập ở phần sau)

Mô hình này hoạt động theo cách: browser sẽ gửi những request lên server, django căn cứ vào url của những request để gọi những function tương ứng trong views lấy dữ liệu từ trong POST hay GET để xử lý. Tùy thuộc vào request mà nó có thể đọc hoặc ghi thông tin từ cơ sở dữ liệu hoặc thực hiện những tác vụ khác để đáp ứng những yêu cầu. Sau đó sẽ trả về phản hồi cho browser dưới dạng HTML, HttpResponse, JsonResponse, …

Diagram

Description automatically generated

Mô phỏng cách thức hoạt động của mô hình MVT

*(Nguồn ảnh:* [*https://ironhackvietnam.edu.vn/django-la-gi/#Tinh\_chat\_cua\_Django*](https://ironhackvietnam.edu.vn/django-la-gi/#Tinh_chat_cua_Django)*)*

### Phát triển ứng dụng Web với Django

#### Cài đặt Django

Để có thể sử dụng Django thì chúng ta nhất định phải cài Python trong máy mình rồi, và khi cài thì Python có kèm theo một chương trình có tên là pip, đây là một phần mềm quản lý các gói mở rộng dành cho Python. Để cài đặt Django thì chúng mở Command Prompt (cmd) lên và gõ lệnh:

pip install django

#### Tạo Project

Để tạo project Django, vào IDE (hay Code Editor) bạn sử dụng Python và đến thư mục muốn tạo project bật terminal lên gõ lệnh sau:

django-admin startproject myproject

Ở đây chúng ta tạo project có tên là myproject. Lệnh startproject sẽ tạo một thư mục có tên là myproject, cấu trúc bên trong thư mục sẽ thư thế này:

Text

Description automatically generated

Lưu ý: khi đặt tên project bạn cần tránh đặt những tên trùng với các từ khóa có sẵn trong Python như: sys, os, django, ... để tránh bị xung đột.

#### Chạy Server

Để chạy server, trỏ đến thư mục myproject gõ câu lệnh:

python manage.py runserver

Text

Description automatically generated

Sau đó sẽ hiện đường dẫn, ta gõ đường dẫn đó trên trình duyệt thì thành công chạy được server:

Graphical user interface, text, application

Description automatically generated

#### Tạo Web App

Bây giờ chúng ta sẽ tạo ứng dụng web: ở terminal trỏ đến thư mục myproject gõ câu lệnh:

python manage.py startapp webapp

Ở đây chúng ta tạo một ứng dụng web tên là webapp, một thư mục webapp được tạo ra và có cấu trúc như sau:

Graphical user interface

Description automatically generated

\* Một project sẽ gồm nhiều nhiều app, trong đó mỗi app thực hiện một công việc riêng biệt.

Sau khi tạo một ứng dụng web thì vào myproject/settings.py khai báo ở phần INSTALLED\_APPS có một web app mới có tên là webapp như hình bên dưới:

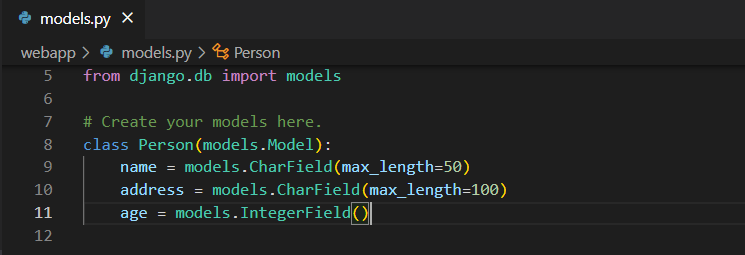
Text

Description automatically generated

#### Model

Model là nguồn định nghĩa thông tin từ cơ sở dữ liệu. Nó bao gồm các lớp và các thuộc tính của dữ liệu. Thông thường, mỗi class model tương ứng với một bảng cơ sở dữ liệu. Mỗi thuộc tính trong class sẽ tương ứng với một trường cơ sở dữ liệu.

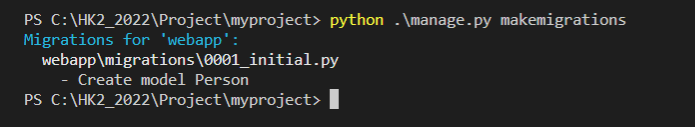
Để tạo một model, vào webapp/models.py:



Person chính là một bảng cơ sở dữ liệu gồm có 3 trường là tên, địa chỉ, tuổi có kiểu dữ liệu là ký tự hoặc số phù hợp với từng trường.

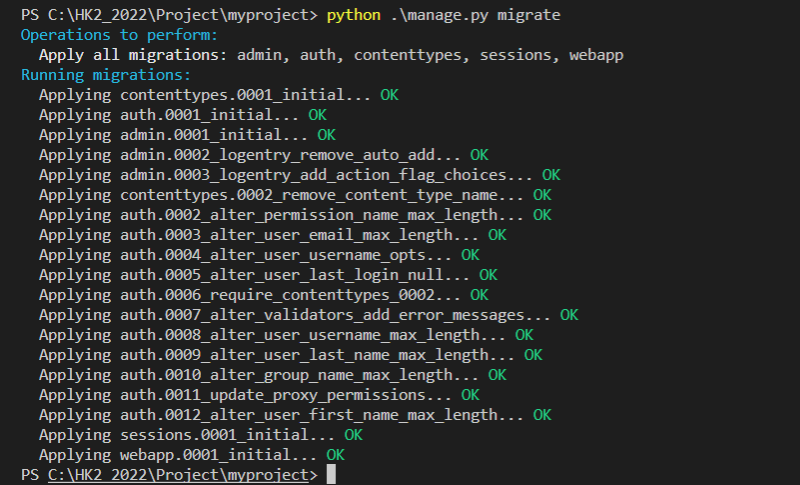
Sau khi tạo ra các model mà mình mong muốn, để thông báo cho Django biết đã có sự thay đổi với các model thì ở terminal trỏ đến thư mục myproject gõ câu lệnh:

python manager.py makemigrations



Thông báo thành công việc tạo mới model tên là Person và các thay đổi về model được lưu như một migration. Tiếp theo chạy câu lệnh:

python manage.py migrate



Câu lệnh migrate sẽ lấy tất cả các thay đổi model của bạn(migration) mà chưa được apply và chạy trên database – đồng bộ tất cả những thay đổi bạn đã tạo cho các model với schema trên database.

Migration rất mạnh mẽ và cho phép bạn thay đổi liên tục mà không cần phải xoá bảng, database hay tạo mới chúng, cũng như không làm mất dữ liệu.

#### Hệ thống admin

Thường ở mỗi website, đặc biệt là web bán hàng, blog hay tin tức thì ta cần một hệ thống admin để quản lý những vấn đề như thêm sửa xoá dữ liệu. Công việc xây dựng admin khá là mất nhiều thời gian và gây ra sự nhàm chán. Ở trong Django đã cung cấp cho ta một hệ thống admin rất là tiện lợi mà ta không cần phải viết ra nó.[[2]](#footnote-2)

Hệ thống Admin của Django hỗ trợ hầu hết các chức năng để ta tương tác với cơ sở dữ liệu như thêm sửa xóa các trường trong các models, phân quyền cho Users, tạo các Groups để quản lý, …

Bây giờ chúng ta sẽ tiến hành tạo một tài khoản admin để truy cập vào hệ thống, ta sẽ mở **Command Prompt lên**và gõ lệnh:

python manage.py createsuperuser

Tiếp theo ta sẽ cung cấp username, password và email sẽ tạo thành công một tài khoản với đặc quyền là superuser (admin).

Text

Description automatically generated

Sau khi đăng nhập thành công, **Admin**mặc định sẽ quản lý 2 bảng là **User**(bảng lưu các user trong hệ thống), và bảng **Group**nhằm tạo các nhóm trong user với những quyền có thể thực thi trong hệ thống.

Graphical user interface

Description automatically generated

Để quản lý thêm những model khác của dự án, ta cần vào file admin.py trong app của dự án đó, và gõ dòng lệnh sau:

admin.site.register(<model\_name>)

#### Views và Templates

Chế độ xem là một “loại” trang web trong ứng dụng Django của bạn thường phục vụ một chức năng cụ thể và có một mẫu cụ thể.

Trong Django, các trang web và nội dung khác được phân phối bởi views. Mỗi view được biểu diễn bằng một python function (hoặc method, trong trường hợp sử dụng class-based views). Django sẽ chọn một view bằng cách kiểm tra URL request (chính xác là một phần của URL sau tên miền)

To get from a URL to a view, Django uses what are known as ‘URLconfs’. A URLconf maps URL patterns to views.

Để chuyển từ một đường dẫn URL đến một View thì Django sử dụng khái niệm URLconfs, URLconfs giúp ánh xạ các URL đến các Views.

#### Upload file

Khi submit file lên server, data file sẽ nằm trong request.FILES. Bắt buộc cho thẻ HTML form phải có thuộc tính enctype="multipart/form-data"  nếu không request.FILES sẽ trống.

Form cần được submit với **POST** methods.

## 2.2. HTML, CSS

### HTML

HTML là viết tắt của Hyper Text Markup Language (ngôn ngữ đánh dấu siêu văn bản). HTML cho phép người dùng tạo và cấu trúc hóa các thành phần trên một trang web như đoạn văn, tiêu đề, liên kết, trích dẫn, bảng biểu, ...

* Các phần tử trong HTML là các khối của trang web HTML, được đại diện bằng những thẻ đánh dấu (tag)
* Thẻ đánh dấu HTML chứa các nội dung như ‘paragraph’, ‘heading’, ‘table’, ...
* Trình duyệt không hiển thị thẻ HTML nhưng dùng chúng để hiển thị nội dung của trang.[[3]](#footnote-3)

HTML không phải là ngôn ngữ lập trình, mà chỉ là ngôn ngữ đánh dấu.

Ví dụ về một văn bản HTML đơn giản:

A screenshot of a computer

Description automatically generated with medium confidence

Trong ví dụ này:

* <!DOCTYPE html> cho biết văn bản này là HTML5
* <html> là phần tử gốc của trang HTML
* <head> chứa thông tin mô tả về văn bản
* <title> là phần tử cho biết tiêu đề văn bản
* <body> là phần tử chứa các nội dung trang sẽ hiển thị
* <h1> là phần tử cho biết sẽ hiển thị tiêu đề lớn
* <p> là phần tử của đoạn văn bản

Ngoài ra, HTML còn chứa nhiều tag khác phục vụ cho nhiều mục đích khác nhau.

### CSS

**CSS** là chữ viết tắt của Cascading Style Sheets, nó là một ngôn ngữ được sử dụng để **tìm và định dạng** lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu ([HTML](https://topdev.vn/blog/html-la-gi/)). Nói ngắn gọn hơn là ngôn ngữ tạo phong cách cho trang web. Bạn có thể hiểu đơn giản rằng, nếu HTML đóng vai trò định dạng các phần tử trên website như việc tạo ra các đoạn văn bản, các tiêu đề, bảng,…thì CSS sẽ giúp chúng ta có thể thêm style vào các phần tử HTML đó như đổi bố cục, màu sắc trang, đổi màu chữ, font chữ, thay đổi cấu trúc, …

Phương thức hoạt động của CSS là nó sẽ tìm dựa vào các vùng chọn, vùng chọn có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác. Sau đó là nó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.

*Mối tương quan giữa HTML và CSS rất mật thiết. HTML là ngôn ngữ markup (nền tảng của site) và CSS định hình phong cách (tất cả những gì tạo nên giao diện website), chúng là không thể tách rời.[[4]](#footnote-4)*

## JavaScript

JavaScript (JS) là một ngôn ngữ lập trình client side, được dùng trong trang web bên cạnh HTML và CSS. HTML dùng tạo bộ xương cho trang web, CSS trang trí và tạo layout cho trang web lung linh hơn, thì JS lập trình để có những hành vi, tương tác với người dùng.

Hiện nay, JS không chỉ có thể viết client side mà còn mở rộng ra nhiều lĩnh vực khác, có thể kể tới như viết app với Electron hoặc web service với Node.js. Các thư viện, framework cho JS phát triển rất mạnh, như jquery, React, Angular, Vue, ...

Trong khuôn khổ dự án, nhóm sẽ sử dụng JS để:

* Tương tác với HTML DOM, lắng nghe và xử lý các Event.
* Tương tác với server để gửi và nhận dữ liệu, dùng dữ liệu đó để hiển thị ra giao diện.

## SQLite

SQLite là một hệ quản trị cơ sở dữ liệu hay còn gọi là hệ thống cơ sở dữ liệu quan hệ nhỏ gọn, khác với các hệ quản trị khác như MySQL, SQL Server, Ocracle, PostgreSQL… SQLite là một thư viện phần mềm mà triển khai một SQL Database Engine truyền thống, không cần mô hình client-server nên rất nhỏ gọn. SQLite được sử dụng vào rất nhiều chương trình từ desktop đến mobile hay là website.[[5]](#footnote-5)

Trong dự án, mặc định thì Django đã sử dụng cơ sở dữ liệu là SQLite. Tuy nhiên, khi thực hiện dự án Django, ta chỉ cần tương tác với cơ sở dữ liệu thông qua các model class cho nên không cần sử dụng các câu query phức tạp.

# CHƯƠNG 3: GIỚI THIỆU DỰ ÁN XÂY DỰNG WEBSITE QUẢN LÝ QUÁN CÀ PHÊ

## Cơ sở dữ liệu

**Graphical user interface, application

Description automatically generated**

Cơ sở dữ liệu gồm các models (tương ứng với các bảng trong SQLite):

* User: với các trường thuộc tính như trên, user sẽ quản lý một tài khoản người dùng truy cập vào web. Tài khoản này có thể là quản lý hay một nhân viên, được xác định ở thuộc tính type.
* TableFood: tương ứng với một bàn trong quán.
* FoodCategory và Food: FoodCategory thể hiện cho một danh mục món, mỗi danh mục sẽ có nhiều món, mỗi món được tượng trưng bằng một model Food.
* Order và OrderInfo: mỗi một Order tương ứng với một hóa đơn của bàn, trong mỗi Order sẽ gồm nhiều OrderInfo nhỏ ứng với một món trong Order đó.
* Ngoài ra sẽ có thêm một model Shop để lưu thông tin của quán như logo của quán, tên quán, địa chỉ, số địa thoại.

## Demo và phân tích các chức năng chính của dự án:

### Chức năng chính

Dự án sẽ bao gồm các chức năng sau đây:

* Đăng nhập/đăng xuất
* Quản lý bàn
* Quản lý việc order/thanh toán
* Quản lý đơn hàng
* Quản lý menu
* Quản lý account
* Thống kê doanh số theo ngày, theo tháng
* Phân quyền quản lý và nhân viên

### Cấu trúc dự án và demo từng chức năng

Web gồm 1 app chính: coffee và 5 apps nhỏ: core, foods, manager, orders, tables. Từng app sẽ đảm nhiệm những chức năng riêng.

Text

Description automatically generated

Sau đây sẽ đi vào cụ thể từng app và demo chức năng mà mỗi app thực hiện:

#### coffee

* Đây là app chính, chỉ lưu những thiết lập cho trang web như settings (file settings.py), đường dẫn gốc cho website , …

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('core.urls')),

    path('foods/', include('foods.urls')),

    path('tables/', include('tables.urls')),

    path('orders/', include('orders.urls')),

    path('manager/', include('manager.urls')),

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

#### core

* Chứa template base.html làm gốc cho các template của các app khác sử dụng.
* Chứa model User được custom lại từ model User của Django (thêm các thuộc tính như hình):

from django.db import models

from django.contrib.auth.models import AbstractUser

type\_choices = ((1, 'Quản lý'), (2, 'Nhân viên'))

class User(AbstractUser):

    first\_name = models.CharField(max\_length=150)

    last\_name = models.CharField(max\_length=150)

    type = models.IntegerField(choices=type\_choices, default=2)

    phone = models.CharField(max\_length=12)

    def full\_name(self):

        return self.last\_name + ' ' + self.first\_name

* Chức năng đăng nhập, đăng xuất:
  + Đăng nhập:

Khi đăng nhập sẽ request url /login và thực hiện:

def loginPage(request):

    if request.user.is\_authenticated:

        return redirect('tables:index')

    if request.method == 'POST':

        username = request.POST.get('username')

        password = request.POST.get('password')

        type = request.POST.get('type')

        user = authenticate(username=username, password=password)

        if user is None:

            try:

                user = User.objects.get(username=username, password=password)

            except Exception:

                pass

        if user is not None:

            if str(user.type) == str(type):

                login(request, user)

                return redirect('tables:index')

        messages.error(request, 'Tài khoản hoặc mật khẩu chưa chính xác')

        messages.error(request, 'Vui lòng kiểm tra lại hoặc liên hệ với quản lý của bạn')

    context = {}

    return render(request, 'core/login.html', context)

Nhập đúng tên và mật khẩu thì vào được trang web, nếu không sẽ báo lỗi.

* + Đăng xuất:

Graphical user interface, text, application, chat or text message

Description automatically generated

Ấn đăng xuất sẽ request với url /logout/ và tiến hành đăng xuất tài khoản hiện tại

def logoutUser(request):

    logout(request)

    return redirect('core:login')

#### foods

* Chứa model FoodCategory và Food:

from django.db import models

class FoodCategory(models.Model):

    name = models.CharField(max\_length=200)

class Food(models.Model):

    name = models.CharField(max\_length=200)

    food\_category = models.ForeignKey(FoodCategory, on\_delete=models.CASCADE)

    price = models.IntegerField(default=0)

    out\_of\_stock = models.BooleanField(default=False) # true: hết món

    image = models.ImageField(blank=True, null=True, upload\_to='images/%Y/%m/%D/')

* Chức năng chính
  + Hiển thị danh sách các món:

Graphical user interface, application

Description automatically generated

Khi request với url /foods, views index sẽ lấy ra danh sách các món từ model FoodCategory và truyền vào context để render ra template index.html hiển thị danh sách món:

def index(request):

    shop = Shop.objects.get(id=1)

    categories = FoodCategory.objects.order\_by('name')

    food\_form = FoodForm()

    edit\_food\_form = EditFoodForm()

    context = {

        'categories': categories,

        'edit\_food\_form': edit\_food\_form,

        'food\_form': food\_form,

        'shop': shop,

    }

    return render(request, 'foods/index.html', context)

* + Thêm danh mục:

Graphical user interface, application

Description automatically generated

Sau khi nhập tên mục và ấn thêm thì url sẽ chuyển tới foods/add\_category/ để gọi hàm add\_category trong view thực hiện bắt dữ liệu trong form và lưu vào cơ sở dữ liệu.

def add\_category(request):

    if request.method == 'POST':

        try:

            category = FoodCategory(name=request.POST['category\_name'])

            category.save()

            return redirect('foods:index')

        except Exception:

            pass

    return HttpResponse('Thêm không thành công')

* + Sửa tên danh mục:

Graphical user interface, text, application, chat or text message

Description automatically generated

Khi bấm vào biểu tượng sửa để đổi tên một danh mục bất kì, nhập tên mới và bấm lưu thì sẽ request với url /foods/edit\_category/ để gọi hàm edit\_category trong views:

def edit\_category(request):

    if request.method == 'POST':

        data = loads(request.body)

        category\_id = data['categoryId']

        new\_name = data['categoryName']

        category = FoodCategory.objects.get(id=category\_id)

        if category:

            category.name = new\_name

            category.save()

            return JsonResponse('Thao tác thành công', safe=False)

    return JsonResponse('Thao tác không thành công', safe=False)

* + Xóa danh mục:

Khi bấm vào biểu tượng xóa danh mục và ấn OK thì sẽ request với url /foods/delete\_category/<category\_id> để gọi hàm delete\_category trong views:

def delete\_category(request, category\_id):

    try:

        category = FoodCategory.objects.get(id=category\_id)

        category.delete()

    except Exception:

        return HttpResponse(Exception)

    return redirect('foods:index')

* + Thêm món:

Graphical user interface, text, application, chat or text message

Description automatically generated

Sau khi nhập thông tin của món và bấm thêm thì sẽ request với url foods/add\_food/ để gọi hàm add\_food trong views:

def add\_food(request):

    if request.method == 'POST':

        food = FoodForm(request.POST, request.FILES)

        if food.is\_valid():

            food.save()

    return redirect('foods:index')

* + Sửa món:

Graphical user interface, text, application, chat or text message

Description automatically generated

Khi bấm lưu sẽ request với url /foods/edit\_food/ để gọi hàm edit\_food trong views:

def edit\_food(request):

    if request.method == 'POST':

        food\_id = request.POST.get('food\_id')

        is\_out\_of\_stock = True if request.POST.get('out\_of\_stock') == 'on' else False

        instance = Food.objects.get(id=food\_id)

        instance.out\_of\_stock = is\_out\_of\_stock

        instance.save()

        food = FoodForm(request.POST, request.FILES, instance = instance)

        if food.is\_valid():

            food.save()

    return redirect('foods:index')

* + Xóa món:

Khi bấm vào biểu tượng xóa món:

Graphical user interface, application

Description automatically generated

Ấn OK sẽ request với url /foods/delete\_food/<food\_id> để gọi hàm delete\_food ở views:

def delete\_food(request, food\_id):

    food = Food.objects.get(id=food\_id)

    food.delete()

    return redirect('foods:index')

Hàm này thực hiện bắt lấy food có id là món mà ta bấm vào, và thực hiện xóa nó trong cơ sở dữ liệu sau đó trả về hàm index (url chuyển về <url>/foods/).

#### manager

* Chứa model Shop: chứa thông tin của quán

class Shop(models.Model):

    name = models.CharField(max\_length=150, default='Tên quán')

    address = models.CharField(max\_length=300, default='Địa chỉ quán')

    phone = models.CharField(max\_length=12, default='SĐT')

    logo = models.ImageField(blank=True, null=True, upload\_to='images/%Y/%m/%D/')

* Giao diện khi chọn vào button Quản lý

Graphical user interface, text, application, chat or text message

Description automatically generated

* Các chức năng trong phần Quản lý:
* Doanh số: Hiển thị doanh số của quán trong những ngày hoặc tháng khác.

Graphical user interface, text, application

Description automatically generated

Khi vào tab doanh số, client sẽ dùng javascript để request lên server và gửi thông tin gồm ngày, tháng, năm hoặc tháng và năm lên server để lấy dữ liệu và hiển thị ra trang web.

var dateObj = new Date()

var todayMonth = dateObj.getUTCMonth() + 1 // months object from 0 - 11

var todayDay = dateObj.getUTCDate()

var todayYear = dateObj.getUTCFullYear()

udpateDayOverview(url, todayDay, todayMonth, todayYear)

updateMonthOverview(url, todayMonth, todayYear)

function udpateDayOverview(url, day, month, year) {

    fetch(url + 'day\_overview/', {

        method: 'POST',

        headers: {

            'Content-Type': 'application/json',

            'X-CSRFToken': csrftoken

        },

        body: JSON.stringify({

            'day': day,

            'month': month,

            'year': year

        })

    })

    .then(response => response.json())

    .then(data => {

        renderDayOverview(day, month, year, JSON.parse(data))

    })

}

function updateMonthOverview(url, month, year) {

    fetch(url + 'month\_overview/', {

        method: 'POST',

        headers: {

            'Content-Type': 'application/json',

            'X-CSRFToken': csrftoken

        },

        body: JSON.stringify({

            'month': month,

            'year': year

        })

    })

    .then(response => response.json())

    .then(data => {

        renderMonthOverview(month, year, JSON.parse(data))

    })

}

Ở phía server, ta sẽ bắt được url và gọi view tương ứng để xử lý, khi nhận được dữ liệu sẽ tiến hành lấy ra thông tin doanh thu liên quan đến ngày tháng năm nhận được và gửi thông tin về cho client

Xử lý khi request lấy thông tin doanh thu ngày:

def day\_overview(request):

    data = json.loads(request.body) # convert => dictionary

    day = data['day']

    month = data['month']

    year = data['year']

    total = day\_total(day, month, year)

    paid\_total = day\_total(day, month, year, status=True)

    unpaid\_total = day\_total(day, month, year, status=False)

    orders = count\_day\_orders(day, month, year)

    paid\_orders = count\_day\_orders(day, month, year, status=True)

    unpaid\_orders = count\_day\_orders(day, month, year, status=False)

    paid\_orderinfos = count\_day\_orderinfos(day, month, year, status=True)

    unpaid\_orderinfos = count\_day\_orderinfos(day, month, year, status=False)

    send\_data = {

        'total': total,

        'paidTotal': paid\_total,

        'unpaidTotal': unpaid\_total,

        'orders': orders,

        'paidOrders': paid\_orders,

        'unpaidOrders': unpaid\_orders,

        'paidOrderinfos': paid\_orderinfos,

        'unpaidOrderinfos': unpaid\_orderinfos,

    }

    return JsonResponse(json.dumps(send\_data), safe=False)

Xử lý khi request lấy thông tin doanh thu tháng:

def month\_overview(request):

    data = json.loads(request.body)

    month = data['month']

    year = data['year']

    total = month\_total(month, year)

    total\_days = []

    n\_days = monthrange(year, month)[1]

    for day in range(1, n\_days + 1):

        total\_days.append(day\_total(day, month, year))

    send\_data = {

        'total': total,

        'totalDays': total\_days,

    }

    return JsonResponse(json.dumps(send\_data), safe=False)

* + - Ta có thể chọn ngày bất kì để xem doanh thu của ngày đó:

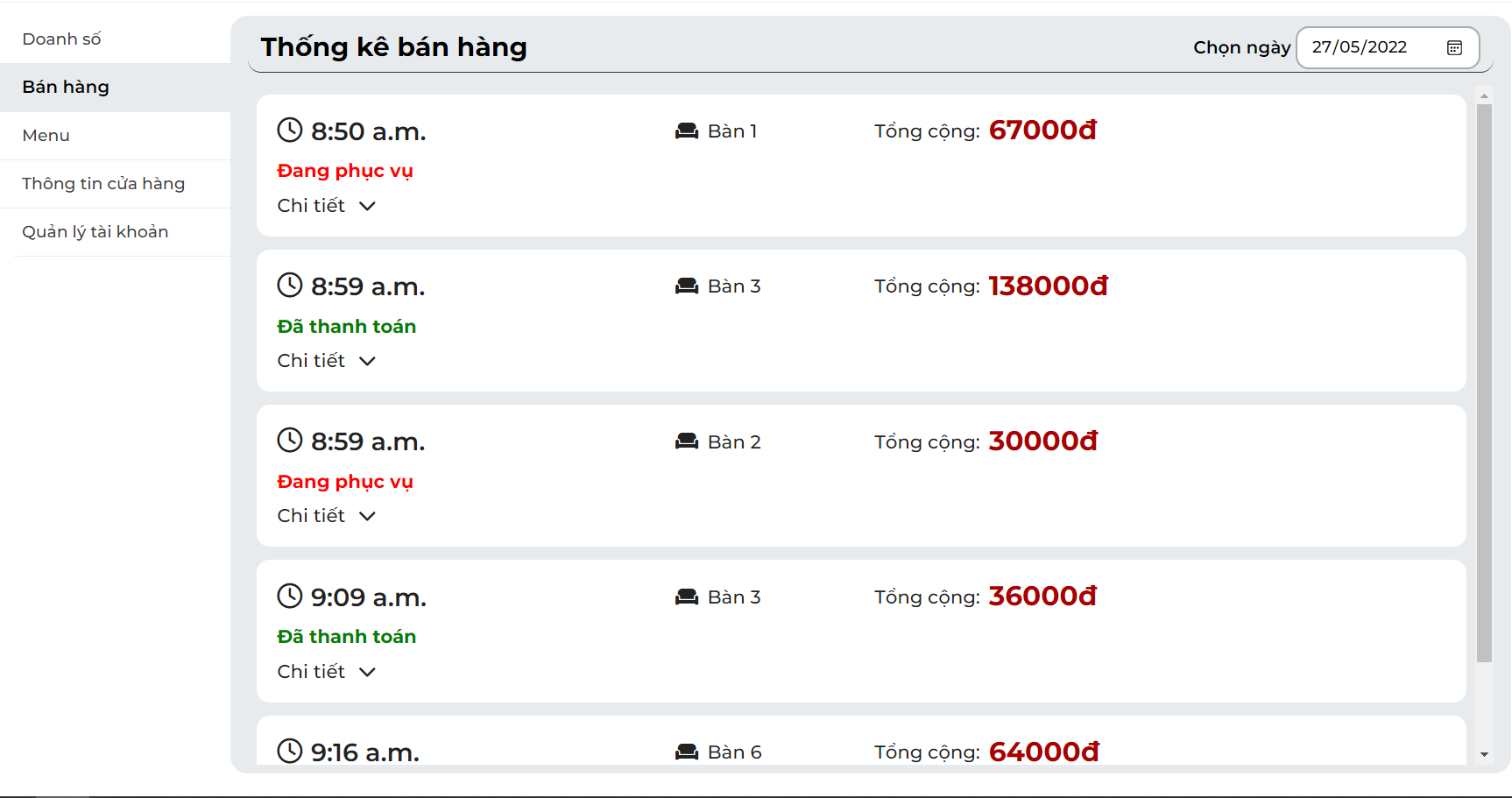
Graphical user interface, text, application, email

Description automatically generated

Khi chọn ngày bất kì, ta cũng request tương tự lúc đầu và với ngày tháng năm là ngày tháng năm mà ta đã chọn.

Tương tự ta có thể chọn tháng tùy ý như chọn ngày.

* + Xem thông tin hóa đơn theo ngày:



Khi vào url orders/<dd>/<mm>/yyyy server sẽ lấy ra thông tin hóa đơn của ngày đó và render ra ngoài giao diện:

def orders(request, day, month, year):

    shop = Shop.objects.get(id=1)

    try:

        orders = Order.objects.filter(created\_\_day=day, created\_\_month=month, created\_\_year=year)

        context = {

            'orders': orders,

            'shop': shop

        }

        return render(request, 'manager/orders.html', context)

    except Exception:

        return HttpResponse(Exception)

* + Thông tin cửa hàng: hiển thị Tên cửa hàng, Địa chỉ, Số điện thoại, Logo và có thể chỉnh sửa phù hợp trong các trường hợp.

Graphical user interface, text, application

Description automatically generated

Ta chỉnh sửa trực tiếp sau khi bấm nút lưu thì thông tin sẽ được cập nhật.



def shop\_info(request):

    shop = Shop.objects.get(id=1)

    fshop = ShopForm()

    context = {

        'shop': shop,

        'fshop': fshop

    }

    return render(request, 'manager/shop\_info.html', context)

def update\_shop\_info(request):

    shop = Shop.objects.get(id=1)

    fshop = ShopForm(request.POST, request.FILES, instance=shop)

    if fshop.is\_valid():

        fshop.save()

    return shop\_info(request)

* + Quản lý tài khoản: tài khoản sẽ được chia thành Quản lý và Nhân viên. Đối với quản lý sẽ được xem, thêm, sửa, xóa các thông tin của quán, còn những tài khoản Nhân Viên chỉ được thực hiện các việc trên ở mục Menu và Thông tin cửa hàng.

Đối với tài khoản Quản lý:

Table

Description automatically generated

Được xem thông tin, có thể thêm mới, sửa, xóa đối với những tài khoản nhân viên hoặc của chính bản thân không thể chỉnh sửa thông tin của những quản lý khác.

Để thêm mới một tài khoản click vào button dấu cộng vào xanh 🡪 Điền đầy đủ thông tin 🡪 Lưu để hoàn thành hoặc Hủy.

Graphical user interface, text, application, email

Description automatically generated

Để sửa thông tin click vào button có hình bút chì 🡪 tùy chỉnh thông tin 🡪 Lưu để hoàn thành hoặc nhấn Hủy

Graphical user interface, text, application, email

Description automatically generated

Đối với tài khoản nhân viên: Có thể chỉnh sửa thông tin của Menu (báo hết món, thay đổi giá,..), Thông tin cửa hàng (chỉ được xem)

Graphical user interface, application

Description automatically generated

Ta sẽ xử lý việc phân quyền này bằng cách chỉ hiển thị ra những chức năng tương ứng ứng với từng loại tài khoản:

{% if request.user.type < user.type or request.user.username == user.username %}

<div class="users-actions">

    <div class="users-action edit" data-user-id="{{ user.id }}" data-username="{{ user.username }}" data-password="{{ user.password }}" data-type="{{ user.type }}" data-first-name="{{ user.first\_name }}" data-last-name="{{ user.last\_name }}" data-phone="{{ user.phone }}">

        <i class="fa-solid fa-pen-to-square"></i>

    </div>

    <a class="delete-user" href="{% url 'manager:delete\_user' user.id %}">

        <div class="users-action delete">

            <i class="fa-solid fa-xmark"></i>

        </div>

    </a>

</div>

{% endif %}

#### orders

* Chứa model Order và OrderInfo:

class Order(models.Model):

    created = models.DateTimeField(auto\_now\_add=True)

    table = models.ForeignKey('tables.TableFood', on\_delete=models.CASCADE)

    status = models.BooleanField(default=False)#true: paid - false: not paid

    note = models.TextField(blank=True, default='')

    def total(self):

        total = 0

        order\_infos = self.orderinfo\_set.all()

        for order\_info in order\_infos:

            total += order\_info.total()

        return total

class OrderInfo(models.Model):

    order = models.ForeignKey(Order, on\_delete=models.CASCADE)

    food = models.ForeignKey('foods.Food', null=True, on\_delete=models.CASCADE)

    quantity = models.IntegerField(default=0)

    def total(self):

        return self.food.price \* self.quantity

* Chức năng chính:
* Hiển thị thực đơn của quán theo từng danh mục. Ở danh mục “Tất cả”, hiện thị ra toàn bộ sản phẩm của quán.
* Hiển thị thông tin hóa đơn của bàn được chọn.

Graphical user interface, application

Description automatically generated

Khi request với url /orders/<table\_id>/, views index sẽ lấy ra danh mục sản phẩm từ model FoodCategory, các sản phẩm trong danh mục đó từ model Food, danh sách bàn từ model TableFood và truyền vào context để template index.html render ra các thành phần nêu trên:

def index(request, table\_id):

    shop = Shop.objects.get(id=1)

    # List categories and foods

    categories = FoodCategory.objects.order\_by('name')

    # List tables

    tables = sorted(TableFood.objects.order\_by('name'), key=lambda o:len(o.name), reverse=False)

    # Order and OrderInfos of table\_id

    table = None

    order = None

    order\_infos = None

    if table\_id != 0:

        table = TableFood.objects.get(id=table\_id)

        try:

            order = table.order\_set.get(status=False)

        except:

            pass

        if order:

            order\_infos = order.orderinfo\_set.all()

    context = {

        'categories': categories,

        'order': order,

        'order\_infos': order\_infos,

        'shop': shop,

        'tables': tables,

        'table': table,

    }

    return render(request, 'orders/index.html', context)

* Thêm món vào bàn (chỉ ở phần giao diện): ấn vào sản phẩm ở danh sách món, món sẽ được thêm vào và thông tin món sẽ hiển thị đầy đủ ở mục “Thông tin hóa đơn”:

Graphical user interface, text, application, chat or text message

Description automatically generated

Nội dung này sẽ được xử lý bằng javascript:

// Handle add item to order (just interface)

const addFoodBtns = document.querySelectorAll('div.add-food:not(.food--out-of-stock)')

var billFoods = document.querySelector('.bill-food-list')

addFoodBtns.forEach((btn) => {

    btn.onclick = (e) => {

        var id = btn.dataset.foodId

        var name = btn.dataset.name

        var price = btn.dataset.price

        // Create a new order item of this food

        var orderItem = billFoods.querySelector(`.bill-food[data-food-id="${id}"]`)

        // If this food is already in order, increase the quantity

        if (orderItem) {

            var quantityInput = orderItem.querySelector('input')

            if (orderItem.style.display == 'none') {

                orderItem.style.display = 'flex'

                quantityInput.setAttribute('value', '1')

            } else {

                quantityInput.setAttribute('value', quantityInput.getAttribute('value') \* 1 + 1)

            }

            quantityInput.value = quantityInput.getAttribute('value')

        } else { // if not in order, create a new one

            var orderItemHtml = createBillInfoHtml(id, name, price)

            billFoods.innerHTML += orderItemHtml

            updateChangeQuantityEvent()

            updateRemoveEvent()

        }

        calcOrderTotal()

    }

})

Khi thay đổi số lượng sản phẩm thì cột “Thành tiền” và “Tổng cộng” sẽ cập nhật theo, khi thay đổi về 0 hay ấn nút X thì món sẽ được xóa khỏi bàn.

* Lưu thông tin hóa đơn: những thay đổi phía trên chỉ cập nhật ở phần giao diện, reload trang web lại thì sẽ bị mất, để lưu những thay đổi này ta sẽ ấn nút lưu, thông tin sẽ được cập nhật lên cơ sở dữ liệu:

Graphical user interface, application

Description automatically generated

Ở phía client: dùng javascript để gửi request với url /orders/update\_order/, dữ liệu sẽ được chuyển lên server dưới dạng JSON. Dữ liệu gửi đi gồm: tableId, note, orderItems.

btnUpdate = document.querySelector('.bill-save')

btnUpdate.onclick = () => {

    var tableId = document.querySelector('.bill-foods').dataset.tableId

    const listFoodInfo = document.querySelectorAll('.bill-food\_\_quantity-input')

    var note = document.querySelector('textarea').value

    var orderItems = []

    listFoodInfo.forEach((food) => {

        orderItems.push(

            {

                'foodId': food.dataset.foodId,

                'quantity': food.getAttribute('value')

            }

        )

    })

    updateOrder(tableId, orderItems, note)

}

function updateOrder(tableId, orderItems, note) {

    const url = getPreviousHref(window.location.href) + '/update\_order/'

    fetch(url, {

        method: 'POST',

        headers: {

            'Content-Type': 'application/json',

            'X-CSRFToken': csrftoken

        },

        body: JSON.stringify({

            'tableId': tableId,

            'note': note,

            'orderItems': orderItems

        })

    })

    .then(response => response.json())

    .then(data => {

        alert(data)

        location.reload()

    })

}

function getPreviousHref(href) {

    href = href.substring(0, href.lastIndexOf('/'))

    return href.substring(0, href.lastIndexOf('/'))

}

Ở server sẽ nhận dữ liệu từ client và tiến hành lưu thông tin bàn, sau đó sẽ trả về một JsonResponse thông báo kết quả:

def update\_order(request):

    data = json.loads(request.body) # convert json => dictionary

    table\_id = data['tableId']

    note = data['note']

    order\_items = data['orderItems']

    table = TableFood.objects.get(id=table\_id)

    order, \_ = Order.objects.get\_or\_create(table=table, status=False)

    order.note = note

    order.save()

    if order\_items == []:

        order.delete()

        order.save()

        table.is\_empty = True

        table.save()

    else:

        for order\_item in order\_items:

            food\_id = int(order\_item['foodId'])

            food = Food.objects.get(id=food\_id)

            quantity = int(order\_item['quantity'])

            order\_info, \_ = OrderInfo.objects.get\_or\_create(order=order, food=food)

            if quantity == 0:

                order\_info.delete()

            else:

                order\_info.quantity = quantity

                order\_info.save()

                table.is\_empty = False

                table.save()

    return JsonResponse('Cập nhật thành công!', safe=False)

* Chức năng thanh toán: thực hiện thanh toán hóa đơn của bàn được chọn

Ở phía client: dùng javascript để gửi request với url /orders/pay\_order/, dữ liệu sẽ được chuyển lên server dưới dạng JSON:

window.addEventListener('load', () => {

    btnPay = document.querySelector('.bill-pay')

    btnPay.onclick = () => {

        if (confirm('Thanh toán bàn này?')) {

            var tableId = document.querySelector('.bill-foods').dataset.tableId

            payOrder(tableId)

        }

    }

})

function payOrder(tableId) {

    const url = getPreviousHref(window.location.href) + '/pay\_order/'

    fetch(url, {

        method: 'POST',

        headers: {

            'Content-Type': 'application/json',

            'X-CSRFToken': csrftoken

        },

        body: JSON.stringify({

            'tableId': tableId

        })

    })

    .then(response => response.json())

    .then(data => {

        alert(data)

        location.reload()

    })

}

Ở server sẽ nhận dữ liệu từ client và tiến hành lưu thông tin bàn, sau đó sẽ trả về một JsonResponse thông báo kết quả:

def pay\_order(request):

    data = json.loads(request.body)

    table\_id = data['tableId']

    table = TableFood.objects.get(id=table\_id)

    try:

        order = Order.objects.get(table=table, status=False)

        if order:

            order.status = True # order đã thanh toán

            order.save()

            table.is\_empty = True # Bàn trống

            table.save()

            return JsonResponse('Thanh toán thành công!', safe=False)

    except Exception:

        pass

    return JsonResponse('Thanh toán không thành công!', safe=False)

Graphical user interface, application

Description automatically generated

Sau khi nhấn OK, trạng thái bàn sẽ được thay đổi thành rỗng (thanh toán thành công).

Graphical user interface, website

Description automatically generated

#### tables

* Chứa model TableFood:

class TableFood(models.Model):

    name = models.CharField(max\_length=100, default="Chưa đặt tên")

    is\_empty = models.BooleanField(default=True)

* Chức năng chính:
  + Hiển thị danh sách bàn:

Graphical user interface, application

Description automatically generated

Khi request với url /tables, views index sẽ lấy ra danh sách bàn từ model TableFood và truyền vào context để template index.html render ra danh sách bàn:

def index(request):

    tables = sorted(TableFood.objects.order\_by('name'), key=lambda o:len(o.name), reverse=False)

    shop = Shop.objects.get(id=1)

    context = {

        'tables': tables,

        'shop': shop

    }

    return render(request, 'tables/index.html', context)

* + Thêm bàn:

Graphical user interface, application

Description automatically generated

Khi nhập tên bàn và ấn thêm, sẽ request với url /tables/add-table/, tiến hành thêm bàn mới vào cơ sở dữ liệu và chuyển hướng trang web về /tables/

def add\_table(request):

    if request.method == 'POST':

        try:

            table = TableFood(name=request.POST['table\_name'])

            table.save()

            return redirect('tables:index')

        except Exception:

            return HttpResponse('Thêm không thành công')

    else: return HttpResponse('Không phải POST')

* + Sửa tên bàn:

Ấn vào nút sửa và chọn vào bàn bất kì:

Graphical user interface, application, chat or text message

Description automatically generated

Khi ấn lưu, sẽ request với url /tables/edit-table/ và tiến hành thay đổi tên bàn:

def edit\_table(request):

    if request.method == 'POST':

        try:

            id = request.POST['table\_id']

            name = request.POST['table\_name']

            table = TableFood.objects.get(pk=id)

            table.name = name

            table.save()

            return redirect('tables:index')

        except:

            return HttpResponse('Sửa không thành công')

    return HttpResponse('Lỗi')

* + Xóa bàn:

Ấn vào nút xóa và chọn bàn bất kì:

Graphical user interface, application

Description automatically generated

Khi ấn OK sẽ request lên url /tables/delete-table/ để thực hiện xóa bàn.

def delete\_table(request, table\_id):

    try:

        table = TableFood.objects.get(pk=table\_id)

        table.delete()

        return redirect('tables:index')

    except:

        return HttpResponse('Xóa không thành công')

* + Chức năng gộp/tách/chuyển bàn:

Khi ấn nút Gộp, tách, chuyển bàn sẽ hiện lên giao diện gộp/tách chuyển bàn:

Graphical user interface, application

Description automatically generated

Chọn bàn cần chuyển/tách và chọn món cần chuyển, sau đó chọn bàn muốn chuyển đến:

Graphical user interface

Description automatically generated

Khi ấn lưu thì các món được chọn ở bàn bên trái sẽ chuyển sang bàn bên phải bằng cách sau:

Ở phía client: dùng javascript để gửi request url /tables/merge-table/ và gửi dữ liệu về bàn cần chuyển, danh sách món cần chuyển và bàn muốn chuyển đến dưới dạng JSON:

// Khi ấn nút Lưu

modal.querySelector('button[type="submit"]').onclick = (e) => {

    // Lấy ra id bàn cần chuyển và id bàn muốn chuyển đến

    var srcTableId = srcTable.dataset.tableId

    var desTableId = desTable.dataset.tableId

    if (srcTableId && desTableId) {

        var checkeds = Array.from(srcTable.querySelectorAll('input[type="checkbox"]:checked'))

        if (checkeds.length > 0) {

            // Lấy ra danh sách món cần chuyển đến gồm là một mảng các Object chứa {tên món, số lượng}

            var orderItems = []

            checkeds.forEach(item => {

                var foodItem = item.closest('.food-item')

                var foodId = foodItem.dataset.foodId

                var quantity = foodItem.querySelector('input[name="food\_quantity"]').value

                orderItems.push({

                    foodId: foodId,

                    quantity: quantity,

                })

            })

            // Request lên server với url tables/merge-table/ để gửi dữ liệu

            const mergeUrl = window.location.href + 'merge-table/'

            if (mergeUrl.includes('/results')) mergeUrl = mergeUrl.replace('/results', '')

            console.log('srcTableId: ', srcTableId,

            ',desTableId: ', desTableId,)

            fetch(mergeUrl, {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json',

                    'X-CSRFToken': csrftoken

                },

                // Dữ liệu gửi đi gồm ID bàn cần chuyển, ID bàn chuyển đến và danh sách món

                body: JSON.stringify({

                    'srcTableId': srcTableId,

                    'desTableId': desTableId,

                    'orderItems': orderItems

                })

            })

            // Bắt dữ liệu mà server gửi về

            .then(response => response.json())

            .then(data => {

                alert(data)

                location.reload()

            })

        } else {

            alert('Chưa chọn món nào')

        }

    }

    else alert('Bàn chuyển đến không hợp lệ')

}

Ở server sẽ nhận dữ liệu từ client và tiến hành việc chuyển bàn, sau đó sẽ trả về một JsonResponse thông báo kết quả:

def merge\_table(request):

    data = loads(request.body) # Nhận dữ liệu từ Client ở dạng JSON và convert sang Dictionary

    # Đọc dữ liệu

    src\_table\_id = data['srcTableId']

    des\_table\_id = data['desTableId']

    order\_items = data['orderItems']

    # Tiến hành việc chuyển bàn

    src\_table = TableFood.objects.get(id=src\_table\_id)

    src\_order = Order.objects.get(table=src\_table, status=False)

    des\_table = TableFood.objects.get(id=des\_table\_id)

    des\_order, created = Order.objects.get\_or\_create(table=des\_table, status=False)

    for order\_item in order\_items:

        food\_id = int(order\_item['foodId'])

        quantity = int(order\_item['quantity'])

        food = Food.objects.get(id=food\_id)

        src\_order\_info = OrderInfo.objects.get(order=src\_order, food=food)

        src\_order\_info.quantity -= quantity

        src\_order\_info.save()

        if src\_order\_info.quantity <= 0:

            src\_order\_info.delete()

        des\_order\_info, created = OrderInfo.objects.get\_or\_create(order=des\_order, food=food)

        if created:

            des\_order\_info.quantity = quantity

        else:

            des\_order\_info.quantity += quantity

        des\_order\_info.save()

    des\_table.is\_empty = False

    des\_table.save()

    if not src\_order.orderinfo\_set.all().exists():

        src\_order.delete()

        src\_table.is\_empty = True

        src\_table.save()

    return JsonResponse('Cập nhật thành công!', safe=False)

# CHƯƠNG 4: KẾT LUẬN

## Kết quả đạt được

* Nắm vững lý thuyết cơ bản phục vụ tốt cho việc thiết kế chương trình.
* Thiết kế được cỡ sở dữ liệu tương đối hoàn chỉnh, đáp ứng tốt cho việc quản lý quán Cà phê.
* Xây dựng được một ứng dụng với đầy đủ những chức năng cơ bản cần thiết cho việc quản lý bán hàng.
* Cùng cố lại kiến thức đã được học, ứng dụng lý thuyết vào thực tế.

## Hạn chế của dự án

* Do thời gian nghiên cứu còn hạn chế, nên việc nghiên cứu, thực hiện và giải quyết các vấn đề còn chưa thực sự tốt.
* Kiến thức lập trình cùng như kiến thức thực tế của các thành viên trong nhóm còn chưa quá vững, nên tính chuyên nghiệp của chương trình chưa cao, chưa đầy đủ các tính năng như mong muốn.

## Định hướng phát triển của đề tài

### Hình thức và chức năng

*Về hình thức:* Cải thiện giao diện, responsive để sử dụng trên nhiều loại thiết bị (PC, tablet, …)

*Về chức năng:*

* Cho phép điều chỉnh danh sách bàn phù hợp với vị trí thực tế trong quán: chia thành từng khu, từng tầng, …
* Thêm chức năng in bill cho hóa đơn.
* Tạo thêm danh sách khách hàng để thực hiện việc tích điểm, …
* Chức năng giảm giá, phụ thu.
* Quản lý nhân viên: lương, thời gian làm việc, …

### Quy mô

Cho phép quản lý một cửa hàng với quy mô lớn hơn: chuỗi cửa hàng với nhiều chi nhanh thay vì một cửa hàng duy nhất, đồng bộ dữ liệu của từng chi nhánh.

# TÀI LIỆU THAM KHẢO

* 1. *Django là gì? Cách hiệu quả nhất để học Django là gì?* (2021) – Fiex Marketing, truy cập tại: <https://ironhackvietnam.edu.vn/django-la-gi/>
  2. *Hệ thống Admin trong Python Django (2021)* – Kteam, truy cập tại: <https://howkteam.vn/course/he-thong-admin-trong-python-django/he-thong-admin-trong-python-django-1523>
  3. *HTML là gì? Cấu trúc HTML cơ bản (2021)* - Vũ An, truy cập tại: <https://quantrimang.com/gioi-thieu-ve-html-149822#:~:text=HTML%20l%C3%A0%20vi%E1%BA%BFt%20t%E1%BA%AFt%20c%E1%BB%A7a,th%E1%BA%BB%20%C4%91%C3%A1nh%20d%E1%BA%A5u%20(tag).>
  4. *CSS là gì?* – TopDev Blog, truy cập tại: <https://topdev.vn/blog/css-la-gi/#css-la-gi>
  5. *Bài 1: SQLite là gì? Ưu và nhược điểm của SQLite – Học SQLite cơ bản* – VNCoder, truy cập tại <https://vncoder.vn/bai-hoc/sqlite-la-gi-uu-va-nhuoc-diem-cua-sqlite-182>
  6. *Đề tài: Xây dựng phần mềm quản lý quán cà phê (2019)* – Slideshare, truy cập tại:<https://www.slideshare.net/trongthuy3/luan-van-xay-dung-phan-mem-quan-li-quan-ca-phe-hot-9d>

1. *Django là gì? Cách hiệu quả nhất để học Django là gì?* (2021) – Fiex Marketing, truy cập tại: <https://ironhackvietnam.edu.vn/django-la-gi/> [↑](#footnote-ref-1)
2. *Hệ thống Admin trong Python Django* (2021) – Kteam, truy cập tại: <https://howkteam.vn/course/he-thong-admin-trong-python-django/he-thong-admin-trong-python-django-1523> [↑](#footnote-ref-2)
3. *HTML là gì? Cấu trúc HTML cơ bản (2021)* - Vũ An, truy cập tại: [HTML là gì? Cấu trúc HTML cơ bản - QuanTriMang.com](https://quantrimang.com/gioi-thieu-ve-html-149822#:~:text=HTML%20l%C3%A0%20vi%E1%BA%BFt%20t%E1%BA%AFt%20c%E1%BB%A7a,th%E1%BA%BB%20%C4%91%C3%A1nh%20d%E1%BA%A5u%20(tag).) [↑](#footnote-ref-3)
4. *CSS là gì? –* TopDev Blog, truy cập tại: <https://topdev.vn/blog/css-la-gi/#css-la-gi> [↑](#footnote-ref-4)
5. *Bài 1: SQLite là gì? Ưu và nhược điểm của SQLite – Học SQLite cơ bản* – VNCoder, truy cập tại <https://vncoder.vn/bai-hoc/sqlite-la-gi-uu-va-nhuoc-diem-cua-sqlite-182> [↑](#footnote-ref-5)