BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH

ĐỖ CAO THÀNH

NGHIÊN CỨU DJANGO, REACT JS XÂY DỰNG HỆ THỐNG QUẢN LÝ NHÀ HÀNG TIỆC CƯỚI

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

TP. HỒ CHÍ MINH, 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH

ĐỖ CAO THÀNH

NGHIÊN CỨU DJANGO, REACT JS XÂY DỰNG HỆ THỐNG QUẢN LÝ NHÀ HÀNG TIỆC CƯỚI

Mã số sinh viên: 1851050131

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

Giảng viên hướng dẫn: TS. TRƯƠNG HOÀNG VINH

TP. HỒ CHÍ MINH, 2021

TRƯỜNG ĐẠI HỌC MỞ **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

THÀNH PHỐ HỒ CHÍ MINH **Độc lập – Tự do – Hạnh phúc**

KHOA CÔNG NGHỆ THÔNG TIN

———

GIẤY XÁC NHẬN

Tôi tên là: Đỗ Cao Thành .....

Ngày sinh: 05-05-2000 ..... Nơi sinh: Tp. Hồ Chí Minh .....

Chuyên ngành: CNTT ..... Mã sinh viên: 1851050131 .....

Tôi đồng ý cung cấp toàn văn thông tin đồ án/ khóa luận tốt nghiệp hợp lệ về bản quyền cho Thư viện Trường Đại học Mở Thành phố Hồ Chí Minh. Thư viện Trường Đại học Mở Thành phố Hồ Chí Minh sẽ kết nối toàn văn thông tin đồ án/ khóa luận tốt nghiệp vào hệ thống thông tin khoa học của Sở Khoa học và Công nghệ Thành phố Hồ Chí Minh.

Ký tên

(Ghi rõ họ và tên)

Thành

……Đỗ Cao Thành……

Ý KIẾN CHO PHÉP BẢO VỆ ĐỒ ÁN/ KHÓA LUẬN TỐT NGHIỆP

CỦA GIẢNG VIÊN HƯỚNG DẪN

Giảng viên hướng dẫn: TS. Trương Hoàng Vinh

Sinh viên thực hiện: Đỗ Cao Thành ..... Lớp: IT81

Ngày sinh: 05/05/2000 .....Nơi sinh: TP Hồ Chí Minh

Tên đề tài: Nghiên cứu Django, React JS xây dựng hệ thống Quản lý nhà hàng tiệc cưới.....

.....

.....

.....

.....

Ý kiến của giảng viên hướng dẫn về việc cho phép sinh viên được bảo vệ đồ án/ khóa luận trước Hội đồng:

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thành phố Hồ Chí Minh, ngày ... tháng ... năm ……

Người nhận xét

……

LỜI CẢM ƠN

Trong khoảng thời gian thực hiện đồ án đã giúp em ôn lại, củng cố những phần kiến thức đã được học, đồng thời trong quá trình thực hiện đồ án cũng giúp em học hỏi thêm được nhiều kiến thức mới nâng cao thêm hiểu biết chuyên môn. Em rất cảm ơn các quý thầy cô bộ môn của khoa Công nghệ thông tin đã truyền đạt những kiến thức nền tảng để em tự tin thực hiện đồ án, đây cũng là những kiến thức sẽ đi theo em trong công việc tương lai.

Đặc biệt, em xin gửi lời biết ơn đến quý thầy TS. Trương Hoàng Vinh, thầy đã tạo điều kiện, chia sẻ những kinh nghiệm, kiến thức và những lưu ý để việc thực hiện đồ án của em được đầy đủ, phù hợp quy định.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TÓM TẮT ĐỒ ÁN

Ứng dụng web với tên gọi Quản lý nhà hàng tiệc cưới là một hệ thống được tạo ra trên nền tảng web nhằm hỗ trợ các doanh nghiệp là về các lĩnh vực nhà hàng tiệc cưới, trang web có hỗ trợ chức năng đặt tiệc online và thanh toán qua ví điện tử, so với các trang web cùng lĩnh vực nhà hàng tiệc cưới thì việc có thêm chức năng đặt tiệc online và thanh toán qua ví điện tử giúp cho khách hàng cảm thấy mới mẻ, thuận tiện và nhanh chóng. Trang web có các chức năng cơ bản như xem thông tin về sảnh cưới, món ăn, đồ uống và các dịch vụ đang được phục vụ. Bên cạnh đó khách hàng có thể đặt tiệc cưới hoặc tiệc thông thường, có thể phản hồi về món ăn, đồ uống, và các dịch vụ nhưng trước tiên phải đăng kí tài khoản. Song hành với các chức năng trên, trang web còn có hệ thống quản trị để quản lý tài khoản và thống kê thông tin.

Trong đồ án này, em tập trung nghiên cứu các nội dung như sau thứ nhất là xây dựng các API phía server để cung cấp cho client, thứ hai là trang quản trị Admin, thứ ba xây dựng trang client gồm các chức năng xem thông tin, tìm kiếm, đặt tiệc, thanh toán, đăng kí người dùng, đăng nhập. Đồ án trình bày những lý thuyết bao gồm Django, Django Rest Framework, ReactJS để xây dựng hệ thống Quản lý nhà hàng tiệc cưới cũng như tổng quan về các chức năng mà hệ thống có thể thực hiện.

TÓM TẮT KHÓA LUẬN
(PHIÊN BẢN TIẾNG ANH NẾU CÓ)

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT VÀ THUẬT NGỮ TIẾNG ANH

STT	Từ viết tắt	Tên đầy đủ	Ý nghĩa
1	Python	Python Programming Language	Ngôn ngữ lập trình Python
2	CSDL	Cơ sở dữ liệu	Cơ sở dữ liệu
3	DRF	Django Rest Framwork	Công cụ phát triển API web
4	MVT	Model-View-Template	Một loại kiến trúc phần mềm
5	MVC	Model – View – Controller	Một loại kiến trúc phần mềm
6	PyCharm	PyCharm	IDE đặc biệt dùng cho Python
7	IDE	Integrated Development Environmen	Công cụ để viết code và phát triển ứng dụng
8	URL	Uniform Resource Locator	Đường dẫn tham chiếu đến nguồn thông tin
9	HTTP	Hyper Text Transfer Protocol	Giao thức Truyền tải Siêu Văn Bản

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

MỞ ĐẦU

Hiện nay, cuộc cách mạng công nghệ 4.0 đã là xu hướng toàn cầu. Vì vậy, việc chuyển đổi để bắt kịp xu hướng diễn ra mạnh mẽ tại các doanh nghiệp vừa và nhỏ tại Việt Nam. Các doanh nghiệp trong các lĩnh vực như kinh doanh bán hàng, dịch vụ, bất động sản đều có những phương pháp áp dụng công nghệ vào kinh doanh để tăng hiệu quả trải nghiệm khách hàng, đồng thời giúp tăng hiệu quả kinh doanh. Trong phân khúc kinh doanh dịch vụ thì kinh doanh dịch vụ ăn uống (Food and Beverage Service) với một lượng nhu cầu rất lớn từ người Việt Nam. Theo thống kê tại Việt Nam người dân thường dành khoảng chi tiêu trung bình cho dịch vụ ăn uống khoảng $361/tháng [1]. Vì vậy, khách hàng sử dụng dịch vụ này cần được quan tâm hơn, tư vấn nhiều hơn và có thể sử dụng dịch vụ thông qua internet, cụ thể hơn là từ website của nhà cung cấp dịch vụ. Đặc biệt hơn, với tình hình đại dịch vừa qua từ nhà hàng cho đến các quán ăn nhỏ đều phải đóng cửa. Tính đến tháng 9/2021, cả nước có 2704 doanh nghiệp kinh doanh lĩnh vực ăn uống tạm đóng cửa và có 728 doanh nghiệp kinh doanh ăn uống phải giải thể [2]. Đây là một con số rất lớn, cho thấy doanh nghiệp chưa có giải pháp đối phó cho biến cố dịch bệnh xảy ra dẫn đến khi dịch bệnh đến mô hình kinh doanh truyền thống bị ngưng hoạt động, khiến doanh nghiệp rơi vào khó khăn, thua lỗ và phá sản. Vậy việc có một website để khách hàng đưa ra nhu cầu về các dịch vụ trong lĩnh vực ăn uống và doanh nghiệp đáp ứng, phục vụ để giải quyết nhu cầu đó là rất cần thiết và phải được tiến hành ngay.

Chương 1. **TỔNG QUAN VỀ ĐỀ TÀI**

1.1. Giới thiệu đề tài

Doanh nghiệp kinh doanh lĩnh vực nhà hàng tiệc cưới là nhà cung cấp dịch vụ ăn uống với chất lượng tốt, sự phục vụ nhiệt tình và đa phần khách hàng hài lòng khi được sử dụng các dịch vụ tại nhà hàng. Đồng thời, các doanh nghiệp này cũng đem đến công việc cho nhiều lao động với thu nhập trung bình khá. Vì vậy, việc sử dụng công nghệ giải quyết khó khăn cho các doanh nghiệp nhà hàng tiệc cưới giúp các doanh nghiệp tồn tại và phát triển là đang góp phần giúp đỡ rất nhiều người từ khách hàng sử dụng dịch vụ cho đến lực lượng lao động trong lĩnh vực này.

Trong đề tài này, em sẽ sử dụng những công nghệ đã được học nhằm tạo ra một hệ thống hoạt động trên nền tảng web để giải quyết các vấn đề trong thực tế của nhà hàng tiệc cưới.

1.2. Lý do chọn đề tài

Với nhu cầu để có giải pháp giúp doanh nghiệp có thể hoạt động hiệu quả khi mô hình kinh doanh truyền thống bị ngưng hoạt động thì việc có một website và có thể đặt hàng và thanh toán trực tiếp tại website sẽ đem đến sự nhanh chóng, thuận tiện cho phần lớn khách hàng có nhu cầu về các dịch vụ nhà hàng tiệc cưới, đồng thời cũng giúp doanh nghiệp vừa và nhỏ trong lĩnh vực nhà hàng tiệc cưới có thể phục vụ khách hàng thông qua nền tảng website, khắc phục được những khó khăn đang gặp phải của cách thức phục vụ truyền thống trong thời điểm dịch Covid. Với những lợi ích tốt đẹp trên nên em đã quyết định chọn "Nghiên cứu Django, Django Rest Framework, React JS để xây dựng hệ thống Quản lý nhà hàng tiệc cưới" làm đề tài.

1.3. Mục tiêu

Mục tiêu của em khi thực hiện đồ án này là nhằm sử dụng những kiến thức đã được học, sau đó tìm hiểu, nghiên cứu những kiến thức mới như Django, Django Rest Framework để xây dựng api phía server kết hợp với React JS để xây dựng single page phía client. Từ kiến thức từ phía server và client sẽ góp phần xây dựng ra một hệ thống Quản lý nhà hàng tiệc cưới trên nền tảng web hoàn chỉnh với nhiều chức năng, có thể ứng dụng trong môi trường thực tế đem lại những lợi ích cho cộng đồng trong mùa dịch Covid hiện tại và cả trong tương lai gần.

1.4. Phạm vi sử dụng và đối tượng nghiên cứu

Phạm vi sử dụng: những cá nhân, tập thể có nhu cầu về các dịch vụ nhà hàng tiệc cưới.

Đối tượng nghiên cứu:

- Công nghệ Django
- Công nghệ Django Rest Framework
- Công nghệ React JS
- Cơ sở dữ liệu MySQL
- Các website nhà hàng tiệc cưới

1.5. Bố cục đề tài

Ngoài phần mở đầu và lời cảm ơn, bố cục đề tài bao gồm các chương sau:

Chương 1: Giới thiệu đề tài, lý do chọn, mục tiêu, bố cục đề tài.

Chương 2: Trình bày cơ sở lý thuyết phía server bao gồm các khái niệm cơ bản về Django, Django Rest Framework.

Chương 3: Trình bày cơ sở lý thuyết phía client bao gồm khái niệm cơ bản về React JS.

Chương 4: Hệ thống quản lý nhà hàng tiệc cưới, trình bày các chức năng và hệ thống.

Chương 5: Kết luận quá trình thực hiện đồ án, trình bày những điều làm được, chưa làm được, đã làm được nhưng chưa tối ưu, hoàn thiện trong đồ án và hướng phát triển trong tương lai.

Chương 2. **CƠ SỞ LÝ THUYẾT PHÍA SERVER**

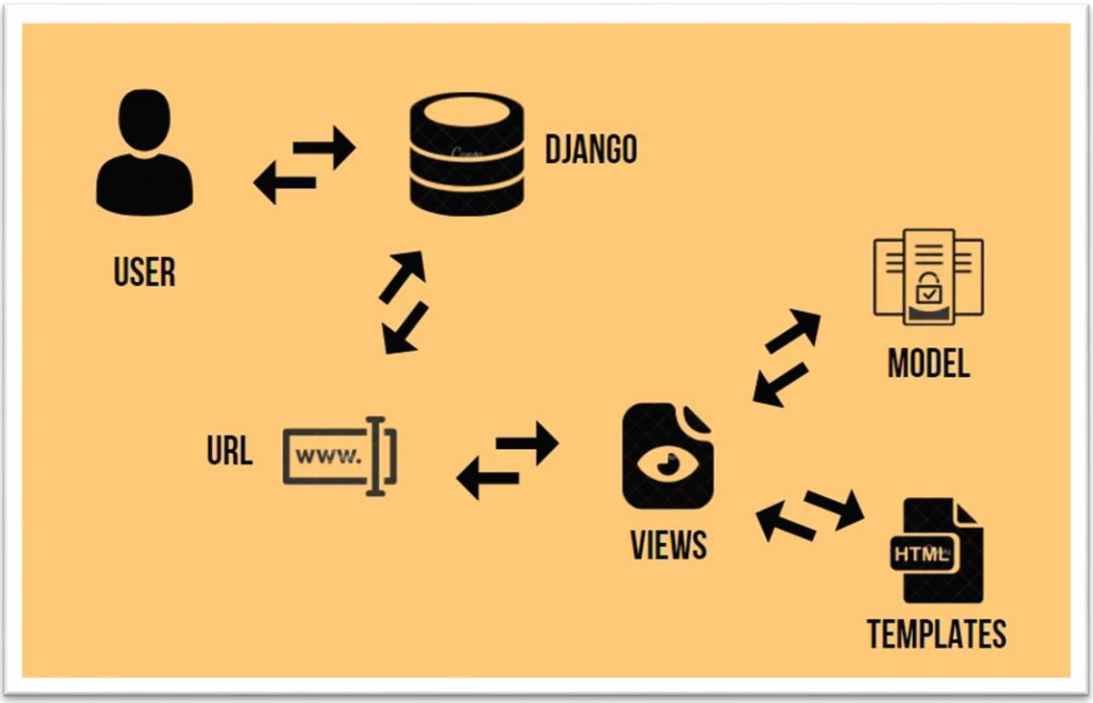
2.1. Giới thiệu Django

Django là một framework mã nguồn mở mạnh mẽ, linh hoạt được viết bằng ngôn ngữ lập trình Python. Các ứng dụng web phức tạp đều có được phát triển một cách dễ dàng, nhanh chóng bởi Django. Đồng thời, Django hỗ trợ rất nhiều tính năng giúp xử lý các tác vụ thông thường, hay lặp lại [3].

Đặc biệt, cả nhà phát triển và cá nhân em thích ở Django là yếu tố bảo mật. Django đã được phát triển tích hợp để phòng tránh các lỗi bảo mật phổ biến như SQL injection, cross-site scripting, cross-site request forgery and clickjacking giúp chúng ta yên tâm phát triển ứng dụng của mình. Bên cạnh đó, Django còn cung cấp hệ thống quản lý và xác thực tài khoản người dùng mạnh mẽ, tin cậy [3].

2.2. Kiến trúc Django

Kiến trúc được sử dụng trong Django là mẫu kiến trúc MVT (Model – View – Template). Cơ bản MVT (Model – View – Template) khá tương đồng với mẫu kiến trúc MVC (Model – View – Controller), nhưng kiến trúc MVT không có bộ điều khiển riêng biệt như MVC, với kiến trúc MVT mọi thứ hoạt động dựa trên Model – View – Template.



Hình 2.1: Kiến trúc của Django

(Nguồn: https://www.askpython.com/django/django-mvt-architecture)

Model: cung cấp các Interface cho dữ liệu được lưu trữ trong CSDL, trong toàn bộ ứng dụng web các tác vụ liên quan đến dữ liệu được xử lý bởi Model.

Template: hiển thị giao diện cho người dùng, lấy thông tin người dùng truyền vào.

Views: xử lý các nghiệp vụ logic đằng sau ứng dụng web. Có thể hiểu đơn giản khi người dùng yêu cầu dữ liệu, View sẽ lấy dữ liệu phù hợp với yêu cầu của người dùng từ CSDL, sau đó dữ liệu truy xuất được hiển thị lại trên template. View có thể được hình dung như chiếc cầu nối giữa Template và Model.

2.3. Model

Trong Django nơi duy nhất tương tác với CSDL là Model. Mỗi model gồm các trường (field) và các hành vi (behavior) dữ liệu được lưu trữ. Mỗi model là một lớp con kế thừa lớp cha là “django.db.models.Model”. Hiểu một cách đơn giản hơn, mỗi model sẽ ánh xạ đến một bảng trong CSDL và mỗi thuộc tính trong model sẽ đại diện cho một trường của bảng trong CSDL [4].

Ví dụ: tạo một model là Lobby với các thuộc tính name, capacity, description.

```
from django.db import models

class Lobby(models.Model):
    name = models.CharField(max_length=150)
    capacity = models.IntegerField(null=False)
    description = models.TextField(null=True)
```

Lobby trong ví dụ sẽ ánh xạ xuống một bảng trong CSDL, với tên bảng được tạo như sau: tênapp_tênmodel. Mỗi thuộc tính name, capacity, description sẽ ánh xạ xuống là một trường trong bảng.

2.3.1. Sử dụng model

Khi định nghĩa một model, ta cần thông báo cho Django biết ta sẽ sử dụng những model mà mình tự định nghĩa bằng cách vào file cài đặt và chỉnh sửa biến INSTALLED_APPS để thêm tên của module chứa các model mà ta định nghĩa.

Ví dụ: Ta có module NhaHangTiecCuoiApp/models.py, file cài đặt setting.py (NhaHangTiecCuoi/settings.py) và một app có tên NhaHangTiecCuoiApp. App được tạo ra khi thực thi lệnh:

```
django-admin startapp <app-name>
```

Đồng thời file INSTALLED_APPS được cài đặt như sau:

```
INSTALLED_APPS = [

    ...

    'NhaHangTiecCuoiApp.apps.NhahangtieccuoiappConfig',

]
```

Lưu ý: sau chỉ định một app được thêm mới vào biến INSTALLED_APPS, ta phải chắc chắn đã thực hiện hai lệnh sau:

```
python manage.py makemigrations
```

Lệnh này thông báo trong file models.py có sự thay đổi.

```
python manage.py migrate
```

Lệnh này tiến hành migration để áp dụng những thay đổi trong models.py và ánh xạ xuống CSDL.

2.3.2. Field

Trong một model thành phần quan trọng và bắt buộc phải có là các trường dữ liệu (field). Các trường được chỉ định bằng các thuộc tính trong một lớp. Mỗi trường tương ứng là một thể hiện của lớp Field phù hợp như IntegerField, BooleanField, CharField, DatetimeField, JsonField…

2.3.3. Các field thông dụng

CharField: dùng để lưu chuỗi nhỏ.

TextField: dùng dể lưu đoạn văn bản lớn.

CharField và TextField có một thuộc tính quan trọng là max_length (chiều dài tối đa).

BooleanField: dùng để lưu giá trị luận lý true hoặc false, giá trị mặc định là None.

IntegerField: dùng lưu các giá trị số nguyên từ - 2147483648 đến 2147483647.

FloatField: dùng lưu các giá trị số dấu chấm động (floating-point).

DecimalField: lưu số dấu chấm tĩnh (fixed-precision decimal number). Thuộc tính quan trọng: max_digits: số chữ số tối đa; decimal_places: số chữ thập phân, giá trị phải nhỏ hơn hoặc bằng max_digits [4].

DateField: dùng để lưu trữ dữ liệu ngày tháng (là thể hiện của datetime.date trong Python). Thuộc tính quan trọng bao gồm: auto_now: tự động thiết lập thời gian hiện tại khi đối tượng được lưu; auto_add_now: tự động thiết lập thời gian hiện tại ngay tại thời điểm đối tượng được tạo lần đầu tiên.

DateTimeField: dùng để lưu trữ data và time.

JSONField: dùng để lưu trữ dữ liệu dạng json.

FileField: dùng để lưu trữ đường dẫn của tập tin khi upload. Thuộc tính quan trọng bao gồm: upload_to: chỉ định thư mục chưa tập tin upload; max_length: chỉ định chiều dài tối đa của đường dẫn.

ImageField: dùng dể upload hình ảnh. Khi sử dụng trường này ta phải cài đặt thư viện pillow bằng lệnh.

```
pip install Pillow
```

Sau đó chỉ định biến MEDIA_ROOT trong file settings.py để Django xác định dường dẫn sẽ lưu ảnh.

```
MEDIA_ROOT/<giá trị thuộc tính upload_to>
```

2.3.4. Các thuộc tính quan trọng của lớp Fields (Field options)

primary_key: chỉ định trường dữ liệu khóa chính, mặc định Django sẽ tự tạo một khóa chính với tên là id có giá trị số nguyên tăng dần [4].

default: chỉ định giá trị mặc định cho trường.

unique: chỉ định giá trị duy nhất trong bảng.

Null: cho phép trường này null, giá trị mặc định là False.

Blank: cho phép trường này rỗng, giá trị mặc định là False.

choices: cho phép trường lưu giá trị là dãy các tuple, sử dụng trong các trường hợp trường có nhu cầu lưu giá trị lựa chọn. Mỗi tuple sẽ bao gồm hai phần.

- Giá trị đầu lưu giá trị thật cho model.
- Giá trị thứ hai là giá trị hiển thị cho người dùng.

Ví du:

```
from django.db import models

class LobbyPrice(models.Model):

    Morning = 'Morning'

    Afternoon = 'Afternoon'

    Evening = 'Evening'

    Weekend = 'Weekend'

    SESSIONS = [

        (Morning, 'Morning'),

        (Afternoon, 'Afternoon'),

        (Evening, 'Evening'),

        (Weekend, 'Weekend')
```

```
    ]
```

2.3.5. Meta options

Sử dụng model metadata để chỉ định những ràng buộc như sắp xếp, tìm kiếm, chỉ định một lớp là abstract. Meta options không bắt buộc phải sử dụng, việc sử dụng meta options giúp cho việc ràng buộc dữ liệu thêm chặt chẽ [4].

Ví dụ:

```
    class Lobby(models.Model):
        class Meta:
            db_table = 'lobby'
        name      =      models.CharField(max_length=150,
unique=True)
```

Các meta options quan trọng:

db_table: thiết lập tên bảng sẽ lưu trữ cho model dưới CSDL.

abstract: thiết lập lớp model là lớp trừu tượng.

app_label: chỉ định tên app chứa model, điều này là bắt buộc nếu app muốn sử dụng các model khai báo bên ngoài phạm vi của nó.

ordering: chỉ định sắp xếp theo trường nào trong một model.

unique_together: chỉ định ràng buộc duy nhất cho nhiều trường trong một model.

2.4. Các loại quan hệ trong Model (Relationships)

Giữa hai model sẽ có các mối quan hệ như many-to-one, many-to-many, one-to-one. Trong quá trình thiết lập CSDL ta sẽ phải chỉ định rõ ràng mối quan hệ giữa các model.

2.4.1. Quan hệ many-to-one

Để thiết lập mối quan hệ many-to-one giữa hai model, ta sử dụng lớp django.db.models.ForeignKey giống như một trường bình thường để chỉ định khóa ngoại. Các đối số mà lớp này yêu cầu là model tham chiếu tới và giá trị của đối số on_delete (cách thức xử lý khi đối tượng tham chiếu bởi ForeignKey bị xóa).

Ví dụ: ta có hai model ServiceType và Service. Một ServiceType sẽ có nhiều Service, một Service chỉ thuộc một ServiceType [4].

```
from django.db import models

class ServiceType(models.Model):
...
name = models.CharField(max_length=100, null=False,
unique=True)

class Service(models.Model):
...
service_type = models.ForeignKey(ServiceType,
on_delete=models.CASCADE)
```

Các giá trị cho đối số on_delete:

SET_NULL: thiết lập giá trị null cho trường ForeignKey, khi đó ta cần phải khai báo thuộc tính null=True trong trường ForeignKey.

SET_DEFAULT: thiết lập giá trị mặc định cho trường ForeignKey, ta cần khai báo thêm thuộc tính default bên trong ForeignKey.

CASCADE: cho biết các đối tượng có tham chiếu ForeignKey sẽ bị xóa theo.

PROTECT: thiết lập ngăn không cho đối tượng tham chiếu ForeignKey bị xóa bằng cách ném ra một ngoại lệ ProtectError.

RESTRICT: thiết lập ngăn không cho đối tượng được tham chiếu bị xóa bằng cách ném ra một ngoại lệ RestrictedError.

Các thuộc tính quan trọng khác:

related_name: chỉ định tên được dùng bởi đối tượng được tham chiếu khoá ngoại sử dụng khi đối tượng được tham chiếu khóa ngoại có nhu cầu truy vấn ngược.

related_query_name: tên dùng để lọc dữ liệu từ đối tượng được tham chiếu khoá ngoại (mặc định là giá trị từ related_name)

2.4.2. Quan hệ many-to-many

Để thiết lập mối quan hệ many-to-many giữa hai model, ta sử dụng lớp "django.db.models.ManyToManyField" và yêu cầu ta truyền vào đối số là model.

Ví dụ: ta có hai model MenuFood và Food. Một MenuFood có nhiều Food và một Food có thể thuộc nhiều MenuFood.

```
from django.db import models

class Food(BaseModel):

     ...

    image   =   models.ImageField(upload_to='images/food',
null=True)


class MenuFood(models.Model):

     ...

    foods = models.ManyToManyField(
Food, related_name="menu_foods", blank=True, null=True)
```

Bên cạnh hai thuộc tính related_name và related_query_name giống trong quan hệ many-to-one thì trong quan hệ many-to-many có những thuộc tính khác quan trọng như:

through: chỉ định bảng trung gian cho quan hệ many-to-many, mặc định Django sẽ tự động tạo một bảng trung gian.

symmetrical: chúng ta sử dụng thuộc tính này khi có nhu cầu chỉ định quan hệ many-to-many giữa một model đến chính model đó.

2.4.3. Quan hệ one-to-one

Khi một chương trình đã đưa vào thực tế, các model đã được tạo đầy đủ và không cho chỉnh sửa. Vì vậy, việc thêm một trường dữ liệu mới vào model (hay còn gọi là mở rộng model) sẽ được giải quyết bằng các sử dụng mối quan hệ one-to-one. Yêu cầu phải truyền vào là model được tham chiếu đến và khóa ngoại sẽ tham khảo trực tiếp đến chính model đó.

Ví dụ: ta có model User và muốn mở rộng model này bằng model UserDataPlus.

```
from django.db import models

class User(models.Model):

     ...

     Pass

Class UserDataPlus(models.Model):

     ...

     user_plus = models.OneToOneField(User,

on_delete=models.CASCADE, primary_key=True)
```

2.5. Truy vấn dữ liệu

Các phương thức truy vấn dữ liệu quan trọng:

create(): tạo đối tượng và lưu xuống CSDL

update(): cập nhật đối tượng và lưu xuống CSDL.

delete(): xóa đối tượng dưới CSDL.

save(): lưu đối tượng xuống CSDL.

count(): đếm số đối tượng trong một QuerySet.

order_by(): mặc định khi QuerySet được tạo sẽ sắp xếp theo thuộc tính ordering trong Meta options của model, tuy nhiên ta cũng có thể thay đổi nó bằng cách ghi đè.

first(): trả về đối tượng đầu tiên trong một QuerySet.

last():trả về đối tượng cuối trong một QuerySet.

latest(): trả về đối tượng cuối trong QuerySet tùy vào trường chỉ định.

earliest(): trả về đối tượng đầu tiên trong QuerySet tùy vào trường chỉ định.

exists(): kiểm tra QuerySet có tồn tại một kết quả nào đó.

aggregate(): thống kê cho QuerySet.

filter(): dùng để lọc dữ liệu theo một điều kiện cụ thể.

exclude(): tương tự filter() tuy nhiên phương thức này dùng để loại bỏ dữ liệu không cần theo điều kiện cụ thể.

get_or_create(defaults, **kwargs): tìm kiếm một đối tượng theo điều kiện, nếu không tìm thấy trong CSDL mới tạo đối tượng [4].

updated_or_create(defaults, **kwargs): cập nhật đối tượng với thông tin tương ứng thoả điều kiện yêu cầu.

2.6. Hệ thống trang quản trị Admin

Trang quản trị admin trong Django chỉ cần đọc dữ liệu từ model để hiển thị ra giao diện một cách dễ dàng, có thể nói đây là một trong những phần hỗ trợ mạnh nhất, tuyệt vời nhất của Django.

Khi cài đặt trang quản trị admin yêu cầu cần có một vài cấu hình trong biến INSTALLED_APPS và TEMPLATES trong file setting.py như sau.

```
    INSTALLED_APPS = [
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
    ]

    TEMPLATES = [
        {
            'BACKEND':
'django.template.backends.django.DjangoTemplates',
            'DIRS': [],
            'APP_DIRS': True,
            'OPTIONS': {
                'context_processors': [

'django.template.context_processors.debug',

'django.template.context_processors.request',
```

```
'django.contrib.auth.context_processors.auth',

'django.contrib.messages.context_processors.messages',
                ],
            },
        },
    ]
```

2.6.1. ModelAdmin class

Trong hệ thống quản trị Admin lớp đại diện là ModelAdmin class. ModelAdmin class được lưu trữ trong file admin.py.

Ví dụ:

```
from django.contrib import admin
from .model import LobbyPrice


class LobbyPriceAdmin(admin.ModelAdmin):
    list_display = ['id', 'lobby', 'price', 'lp_sessions']
    search_fields = ['lobby__name', 'lobby__capacity',
'price']
    list_filter = ['price']
```

Trong nhiều trường hợp ta chỉ có nhu cầu sử dụng giao diện Admin mặc định thì không cần thiết phải sử dụng lớp này. Và ta có thể đăng kí model mặc định như sau.

```
admin.site.register(LobbyPriceAdmin)
```

2.6.2. Superuser

Trong trang quản trị admin ngoài việc đăng kí các model thì ta cần phải có một tài khoản với vai trò admin có toàn quyền để quản trị trang. Đăng kí tài khoản với vai trò admin bằng lệnh sau.

```
    python manage.py createsuperuser
```

Sau đó ta phải nhập các thông tin như username, password, email và được xác nhận là đăng kí thành công thì có thể sử dụng tài khoản vừa đăng kí để đăng nhập vào trang admin.

2.6.3. InlineModelAdmin

Để các model có quan hệ với nhau hiển thị trên cùng một trang trên trang quản trị Admin thì InlineModelAdmin trong Django cung cấp hai dạng gồm TabularInline, StackedInline [4].

Ví dụ: ta có hai model Lobby và LobbyPrice. Để model LobbyPrice hiển thị trong cùng một trang với model Lobby để thuận lợi theo dõi.

```
class LobbyPriceInlineAdmin(admin.StackedInline):
    model = LobbyPrice
    fk_name = 'lobby'



class LobbyPriceAdmin(admin.ModelAdmin):
    list_display = ['id', 'lobby', 'price', 'lp_sessions']
    search_fields = ['lobby__name', 'lobby__capacity',
'price']
    list_filter = ['price']



class LobbyAdmin(admin.ModelAdmin):
     ...
    inlines = [LobbyPriceInlineAdmin, ]
```

2.7. URL Dispatcher

URL trong ứng dụng web là một phần rất quan trọng. Django cho phép ta xây dựng URL theo cách ta muốn, và không giới hạn khuôn mẫu. Trong ứng dụng URL

được thiết kế bằng cách tạo ra một module là URLconf (đây là module ánh xạ giữa các biểu thức đường dẫn URL đến các view trong Django.

Khi người dùng muốn vào một trang web. URLconf gốc dùng để sử dụng sẽ được xác định bởi Django. Giá trị của URLconf gốc sẽ được cài đặt ở biến ROOT_URLCONF trong file setting.py. Sau đó từ URLconf gốc Django sẽ tìm biến urlpatterns (là một chuỗi django.urls.path() hoặc django.urls.re_path()) và tìm từng mẫu URL theo thứ tự và sẽ dừng lại ở mẫu đầu tiên phù hợp với URL được yêu cầu. Mặc khác, Django sẽ ném ngoại lệ hoặc gọi view ngoại lệ nếu không tìm được URL yêu cầu [5].

2.8. Django Authentication

Hệ thống xác thực người dùng do Django cung cấp vô cùng mạnh mẽ, cho phép quản lý người dùng, phân quyền, tạo nhóm, phiên làm việc dựa trên cookie. Ngoài ra, hệ thống còn hỗ trợ xử lý cả authentication (xác thực user là ai) và authorization (xác thực user có quyền gì).

Bên cạnh đó, Django có hỗ trợ xác thực và được đóng gói trong module "django.contrib.auth". Mặc định, tại thời điểm tạo project các yêu cầu cấu hình trong file setting.py sẽ đồng thời được tạo cùng lúc. Hai mục sẽ được cấu hình nằm trong biến INSTALLED_APPS của file setting.py là 'django.contrib.auth' (chứa thông tin cốt lõi và cái model mặc định) và 'django.contrib.contenttypesngo' (hệ thống cho phép liên kết các quyền với các model).

Ngoài ra, còn cấu hình thêm các mục khác nằm trong biến MIDDLEWARE của file setting.py là "SessionMiddleware" (quản lý phiên qua những yêu cầu của người dùng) và "AuthenticationMiddleware" (liên kết quyền với những yêu cầu của người dùng). Sau khi cấu hình xong các mục, cần phải yêu cầu Django tạo các bảng dữ liệu cần thiết cho các model liên quan đến xác thực bằng cách thực hiện lệnh.

```
python manage.py migrate
```

2.9. Django Rest Framework

2.9.1. Giới thiệu Django Rest framework (DRF)

Ngày nay, để xây dựng một ứng dụng web sẽ có rất nhiều yêu cầu và tiêu chuẩn được đặt ra tùy thuộc vào nhu cầu sử dụng. Và RESTful API là một tiêu chuẩn. Việc

sử dụng tiêu chuẩn RESTful API giúp cho ứng dụng web tương tác với CSDL mà không cần kết nối trực tiếp xuống CSDL. Django REST framework là một bộ công cụ linh hoạt và mạnh mẽ hỗ trợ chúng ta điều đó.

Những ưu điểm quan trọng của Django REST API:

- Hỗ trợ giao diện duyệt API hiệu quả cho lập trình viên.
- Hỗ trợ nhiều bộ công cụ chứng thực tiêu biểu như Oauth2.
- Serialization hỗ trợ cả các data source theo ORM và không ORM (non-ORM).
- Có cộng động hỗ trợ lớn và tài liệu phong phú trên internet.

Cài đặt Django Rest Framework bằng lệnh.

```
pip install djangorestframework
```

Cập nhật Django Rest Framework vào biến INSTALLED_APPS trong file settings.py.

```
INSTALLED_APPS = [

 ...

 'rest_framework',

]
```

2.9.2. Requests

Trong Django, lớp Request của DRF là mở rộng của lớp HttpRequest tiêu chuẩn. Tuy nhiên, nó mạnh mẽ, linh hoạt hơn trong việc xác thực và phân tích request. Các thuộc tính quan trọng của request:

request.data: trả về nội dung của request body.

request.query_params: trả về các tham số truyền từ request.get.

request.method: trả về các phương thức dưới dạng chữ hoa của HTTP Request như get, post, put, delete, patch.

request.conten_type: trả về một đối tượng chuỗi đại diện cho media type của request body.

request.user: các user đã chứng thực thì nó trả về thể hiện của django.contrib.auth.models.User. Ngược lại, nó là thể hiện của django.contrib.auth.models.AnonymousUser.

request.auth: trả về các thông tin bổ sung chứng thực .

request.authenticators: trả về các chính sách chứng thực.

2.9.3. Responses

DRF hỗ trợ trả về thông tin cho Client bằng cách cung cấp một lớp Response cho phép chúng ta trả về thông tin dưới nhiều media type phụ thuộc vào yêu cầu của người dùng.

Những thuộc tính trọng yếu của Response:

data: dữ liệu đã được serialize.

status: trạng thái của request cho response, mặc định 200.

headers: từ điển HTTP headers cho response.

content_type: loại media type cho response.

template_name: tên template sử dụng nếu HTMLRenderer được chọn [6].

2.9.4. Generic Views

Để phát triển các API View nhanh hơn có thể sử dụng Generic View.

Một số Generic view có thể kể đến như: CreateAPIView, ListAPIView, RetrieveAPIView, DestroyAPIView, UpdateAPIView, …

Các thuộc tính quan trọng trong GenericAPIView:

queryset: Câu truy vấn trả về các đối tượng. Nó sẽ được thực thi một lần khi được sử dụng (evaluated) và được lưu đệm (cache) cho các lần sau. Ta có thể ghi đè phương thức get_queryset() để chỉnh sửa queryset khi cần.

serializer_class: Chỉ định lớp serializer cho dữ liệu. Ta có thể ghi đè phương thức

get_serializer_class(): để thay thế.

pagination_class: Để phân trang cho kết quả list, ta cần cho biết cụ thể giá trị của thuộc tính này là một lớp phân trang, mặc định sử dụng giá trị cấu hình của DEFAULT_PAGINATION_CLASS. Nếu paginate_class=None sẽ không phân trang.

get_object(): trả về thể hiện của đối tượng được sử dụng cho detail [6].

2.9.5. ViewSet

ViewSet cung cấp những phương thức như list(), create(), retrieve(), destroy(), update() và không cung cấp phương thức như get(), post(). Và ViewSet sẽ không hiện thực sẵn các phương thức [8].

2.9.6. Serializers

Serializer là chuyển dữ liệu phức tạp trong hệ thống thành dữ liệu ra bên ngoài thường là JSON cho phía client sử dụng. Ngược lại, khi hệ thống nhận vào dữ liệu dạng JSON thì chuyển nó về thành dữ liệu phức tạp ban đầu sau khi xác thực (validate) được gọi là deserializer [8].

2.9.7. Routers

Thiết lập Router cho viewsets tương tự việc ta cung cấp urlpattern. Khi thiết lập cần cung cấp hai đối số bắt buộc trong phương thức register().

prefix: phần tên trước trên URL được sử dụng cho tất cả router, dạng {prefix}/

viewset: lớp ViewSet.

2.9.8. OAuth2

OAuth là giao thức cho phép chứng thực người dùng giữa các dịch vụ liên quan. Trong OAuth cho phép người dùng ủy quyền cho một dịch vụ đại diện mà không cần cung cấp username, password, để truy cập tài nguyên từ một dịch vụ khác lưu giữ tài nguyên người dùng.

Để sử dụng chứng thực bằng OAuth2, yêu cầu người phát triển phải đăng kí một application với OAuth2 provider. Khi quá trình đăng kí hoàn tất ta sẽ nhận được các thông tin là client_id, client_secret,…

Trong OAuth2 quá trình giao tiếp giữa client và server gồm.

- Authorize: chứng thực yêu cầu quyền truy cập từ client.
- Yêu cầu access token.
- Truy cập các tài nguyên được bảo vệ.

Để giao thức OAuth2 được thực hiện ta sử dụng công cụ Django Toolkits.

Cài đặt Django Toolkit bằng lệnh.

```
pip install django-oauth-toolkit
```

Cập nhất biến INSTALLED_APP trong file settings.py.

```
INSTALLED_APPS = (

 ...

 'oauth2_provider',

)
```

Bổ sung thông tin cấu hình cho biến REST_FRAMEWORK trong file settings.py.

```
REST_FRAMEWORK = {

 ...

 'DEFAULT_AUTHENTICATION_CLASSES': (

 'oauth2_provider.contrib.rest_framework.

 OAuth2Authentication',)

}
```

Cập nhật urls cho URLConfig của project.

```
from django.urls import include, path

urlpatterns = [

 ...

 path('o/', include('oauth2_provider.urls',

 namespace='oauth2_provider')),

]
```

Sau đó thực thi migrate.

```
python manage.py migrate oauth2_provider
```

Yêu cầu lấy access token.

- Url: /o/token/
- Method: POST
- Body data

Body data như sau.

```
{
 "grant_type": "password",
 "username": "string",
 "password": "string",
 "client_id": "string",
 "client_serect": "string"
}
```

Sau khi lấy được access token, lúc này ta có thể truy cập tài nguyên yêu cầu chứng thực mà không cần username, password. Trong HTTP header của mỗi request phải gửi kèm Authorization, giá trị của nó có thể là Token hoặc Bearer phụ thuộc OAuth2 provider.

2.9.9. CORS

Để giới hạn các tài nguyên trên trang web được yêu cầu truy cập từ các domain khác ta sử dụng cơ chế CORS (Cross-origin resource sharing), đồng thời đây là cơ chế cho phép server và trình duyệt xác định request cross-origin có an toàn không.
Cài đặt CORS bằng lệnh.

```
pip install django-cors-middleware
```

Bổ sung biến INSTALLED_APP trong file settings.py.

```
INSTALLED_APP = [
 ...
 'corsheaders'
]
```

Cập nhật biến MIDDLEWARE của settings.py giá trị

corsheaders.middleware.CorsMiddleware. Nên đặt giá trị này trước các middleware có thể tạo ra response.

```
MIDDLEWARE = [
```

```
 'corsheaders.middleware.CorsMiddleware',
 ...
]
```

Thiết lập giá trị biến sau trong file settings.py.

```
CORS_ORIGIN_ALLOW_ALL = True
```

2.10. Thiết lập chương trình đầu tiên phía server

Trong đồ án, em dùng IDE là Pycharm để tạo một chương trình cơ bản phục vụ cho phía server bằng Django kết hợp với Django Rest Framework. Đồng thời, em sử dụng MySQL Workbench để tương tác với CSDL MySQL. Tất cả đều được thực thi trên hệ điều hành Window 10.

2.10.1. Cài đặt Django

2.10.1.1. Kiểm tra cài đặt Python

Vì Django là một framework phát triển dựa trên Python nên muốn sử dụng được ta phải đảm bảo Python đã được cài đặt trên máy tính.

Kiểm tra Python đã được cài Python bằng cách vào Command Prompt gõ lệnh.

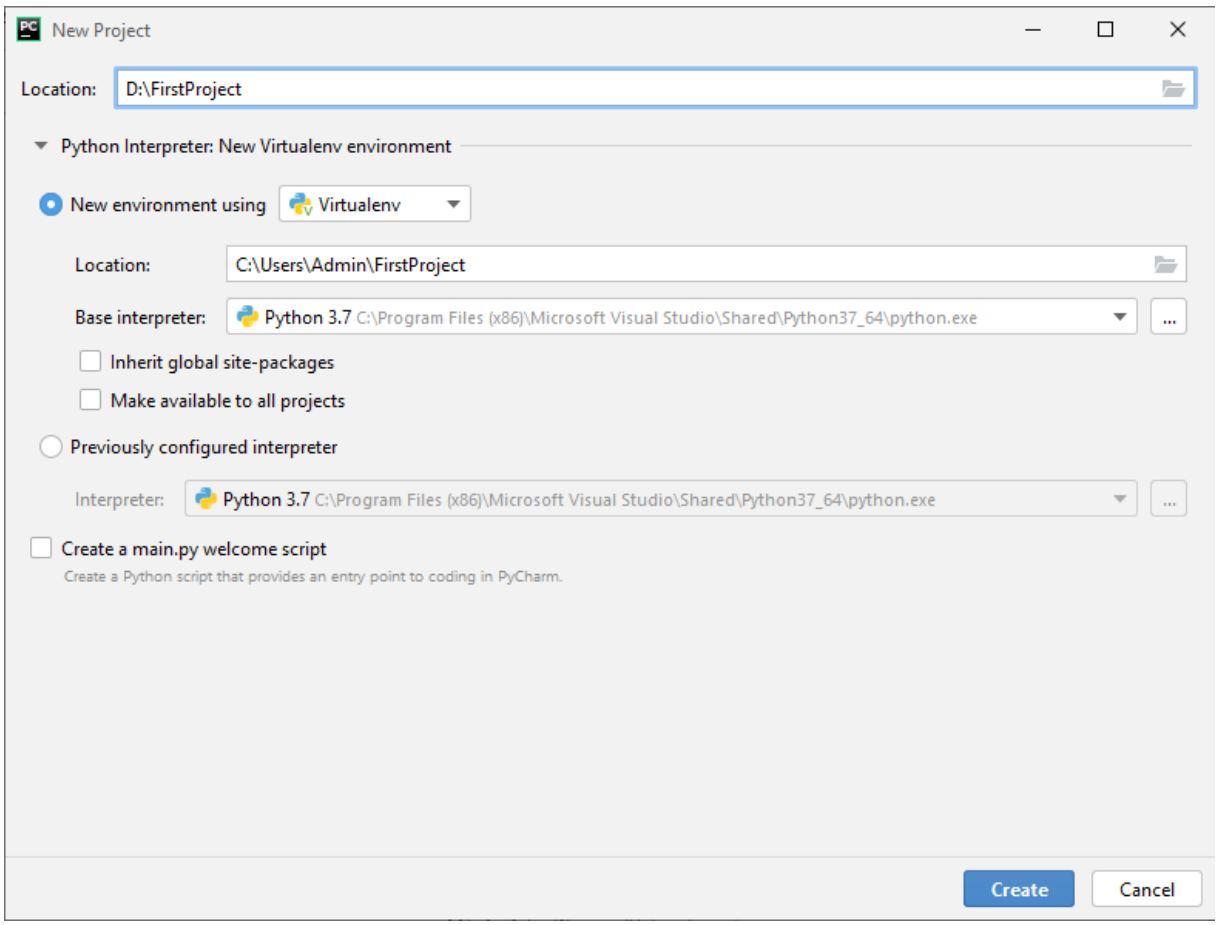
```
python --version
```

Sau khi lệnh được thực thi sẽ hiển thị kết quả phiên bản của Python hiện có trong máy tính.

```
C:\Users\Admin>Python --version
Python 3.7.8
```

2.10.1.2. Tạo môi trường ảo venv

Để việc phát triển ứng dụng hiệu quả em sử dụng môi trường ảo hóa cho Python.



Hình 2.2: Tạo môi trường ảo venv

2.10.1.3. Cài đặt môi trường

Để cài đặt Django ta vào Terminal của IDE, sau đó gõ lệnh.

```
pip install django
```

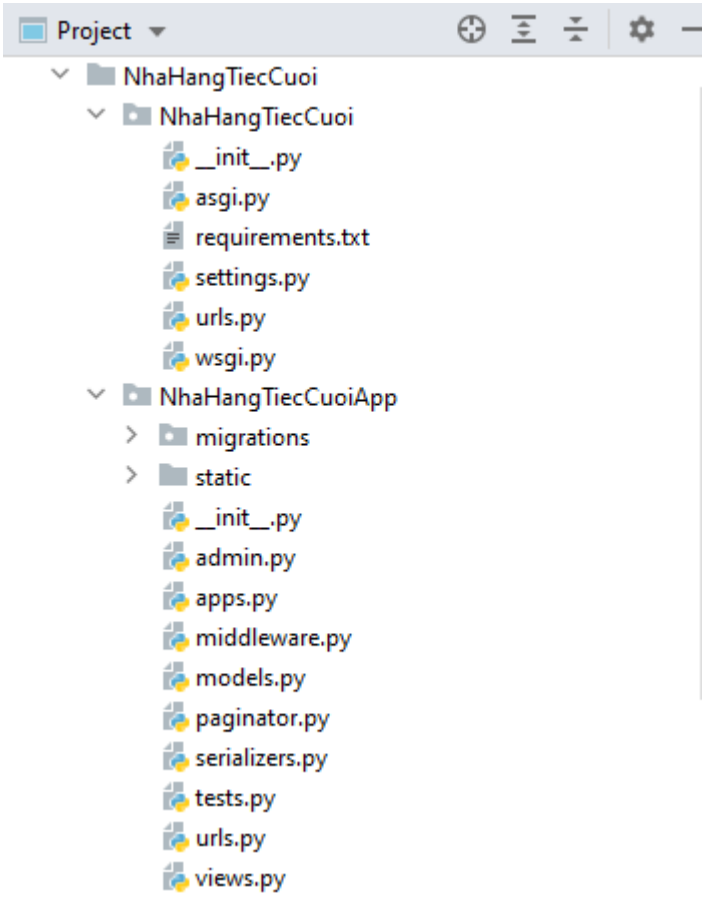
Sau khi cài đặt Django ta tiến hành tạo project Django bằng lệnh.

```
django-admin startproject NhaHangTiecCuoi
```

Tiếp theo, trong project Django ta tiến hành tạo app bằng lệnh.

```
django-admin startapp NhaHangTiecCuoiApp
```

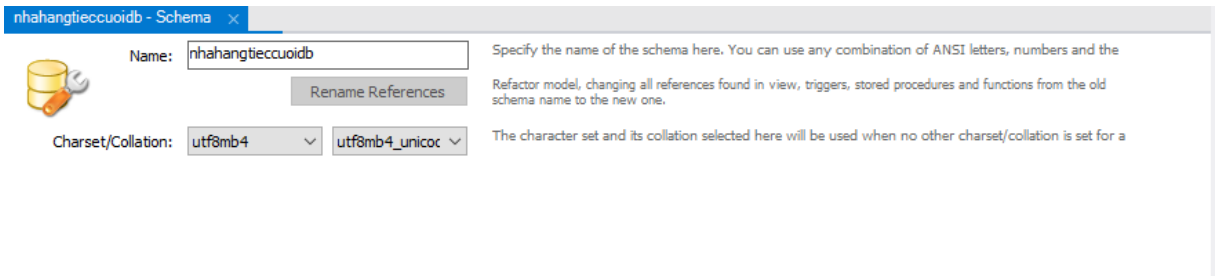
Cấu trúc project sau khi cài đặt xong sẽ như sau.



Hình 2.3: Cấu trúc project ban đầu của Django

2.10.2. Kết nối cơ sơ dữ liệu

Đầu tiên, tiến hành tạo một cơ sơ dữ liệu rỗng trong MySQL Workbench với tên shanedb.



Hình 2.4: Tạo cơ sở dữ liệu rỗng

Tiếp theo, ta cấu hình kết nối cơ sở dữ liệu trong biến DATABASES trong file setting.py.

```
DATABASES = {

 'default': {

      'ENGINE': 'django.db.backends.mysql',
```

```
        'NAME': 'shanedb',
        'USER': 'root',
        'PASSWORD': 'Thait123$',
        'HOST': '',
    }
}
```

Trong đó:

ENGINE: chỉ định engine hệ quản trị CSDL sẽ sử dụng.

NAME: tên cơ CSDL.

USER: username của CSDL.

PASSWORD: password của CSDL.

HOST: host chứa CSDL (mặc định là localhost).

Tại biến INSTALLED_APPS trong file settings.py ta cấu hình cho Django biết sự tồn tại của NhaHangTiecCuoiApp.

```
INSTALLED_APPS = [
'NhaHangTiecCuoiApp.apps.NhahangtieccuoiappConfig',
    ...
]
```

Sau khi hoàn thành cấu hình kết nối cơ sở dữ liệu, ta tiếp tục cài đặt MySQL driver bằng lệnh.

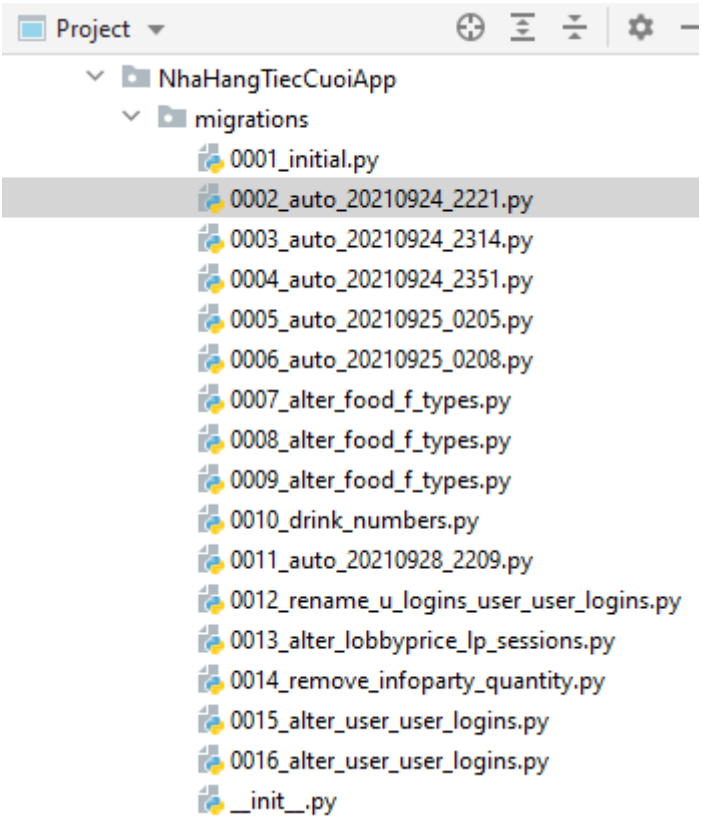
```
pip install mysqlclient
```

Trong NhaHangTiecCuoiApp/models.py tiến hành tạo các model cần thiết. Sau đó vào Terminal của IDE và thực thi lệnh.

```
python manage.py makemigrations NhaHangTiecCuoiApp
```

Lệnh makemigrations thông báo cho Django biết trong file models.py có sự thay đổi (nhưng chưa thật sự thay đổi dưới cơ sở dữ liệu cho đến khi lệnh migrate được thực thi) và mỗi lần lệnh makemigrations được thực thi thì trong thư mục NhaHangTiecCuoiApp/migrations sẽ có một file được tạo thành với mục đích lưu trữ

thông tin vào thời gian nào và cơ sở dữ liệu có những thay đổi. Thông tin các file trong thư mục migrations.



Hình 2.5: Các file được tạo ra qua nhiều lần migrations

Cập nhật những thay đổi của file models.py xuống cơ sở dữ liệu bằng lệnh.

```
python manage.py migrate
```

Lúc này phía dưới CSDL mới thật sự có sự thay đổi giống với những thay đổi trên file models.py. Lưu ý, chỉ các app được khai báo trong biến INSTALLED_APPS mới thực hiện được lệnh migrate.

2.10.3. Cấu hình Admin

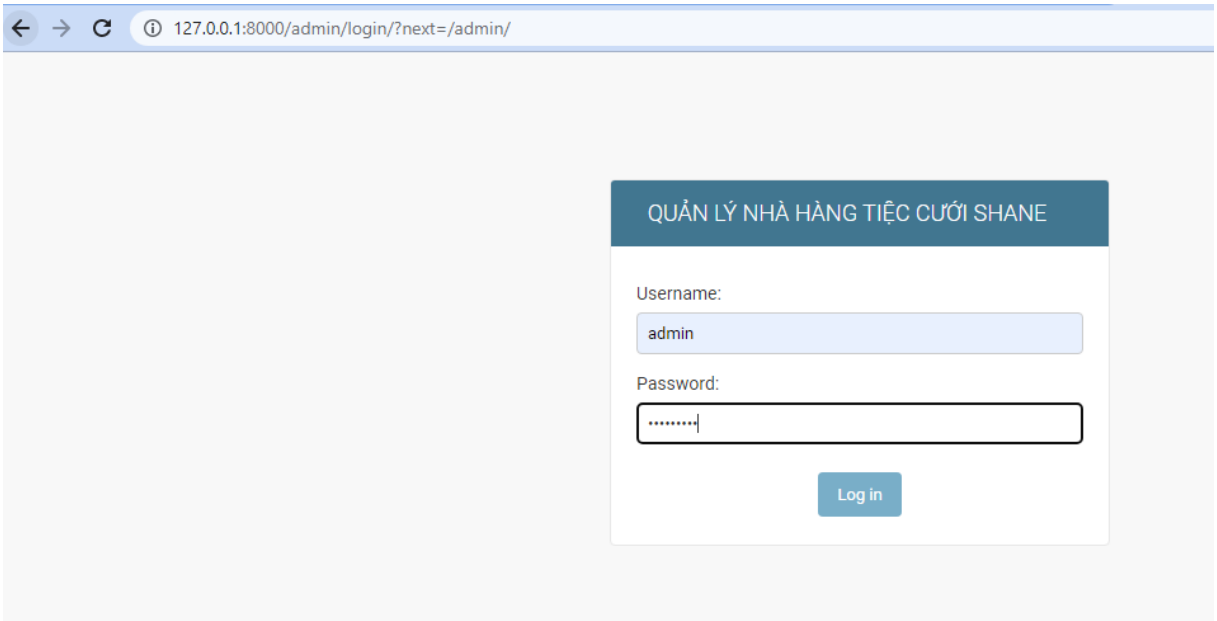
Tạo một tài khoản với vai trò admin trong Terminal bằng lệnh.

```
python manage.py createsuperuser
```

Vào terminal và chạy lệnh sau.

```
 python manage.py runserver
```

Sau đó truy cập vào "http://127.0.0.1:8000/admin" và đăng nhập bằng tài khoản Admin vừa tạo.



Hình 2.6: Trang đăng nhập hệ thống quản trị Admin

2.10.4. Cài đặt Django Rest Framework

Cài đặt môi trường bằng lệnh.

```
pip install djangorestframework
```

Cập nhật biến INSTALLED_APPS.

```
INSTALLED_APPS = [

 ...

 'rest_framework',

]
```

Tạo NhaHangTiecCuoiApp/serializers.py.

```
from rest_framework.serializers import *

from .models import *


class LobbyPriceSerializer(ModelSerializer):

    class Meta:

        model = LobbyPrice

        fields = ['id', 'lp_sessions', 'price', 'lobby']
```

Trong NhaHangTiecCuoiApp/views.py.

```
from rest_framework import viewsets, generics
from .serializers import *


class LobbyPriceViewSet(
viewsets.ViewSet, generics.ListAPIView):
    queryset = LobbyPrice.objects.all()
    serializer_class = LobbyPriceSerializer
```

Trong NhaHangTiecCuoiApp/urls.py.

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from . import views


router = DefaultRouter()
router.register('lobby_prices',   views.LobbyPriceViewSet,
basename='lobby_price')


urlpatterns = [
    path('', include(router.urls)),
]
```

Cập nhật biến urlpatterns trong CharitySocialNetwork/urls.py.

```
from django.urls import path, include


urlpatterns = [
    path('', include('NhaHangTiecCuoiApp.urls')),]
```

Chạy server và truy cập “http://127.0.0.1:8000/” và sau khi truy cập có giao diện như sau.

Hình 2.7: Giao diện API Root

Vậy là ta vừa cài đặt và thực thi một chương trình đơn giản nhất kết hợp giữa Django, Django Rest Framework, và cơ sở dữ liệu MySQL để viết api.

Chương 3. **Cơ sở lý thuyết phía client**

3.1. Giới thiệu React JS

React JS cung cấp cho người dùng cơ chế để tách giao diện UI (header, body, footer,…) thành nhiều phần gọi là (component), component nào cố định thì giữ nguyên, component nào cần thay đổi thì ghi đè lên. Mục tiêu của việc tách thành nhiều phần là để tái sử dụng hiểu quả cao. React JS hoạt động cả hai phía client và server nhưng tập trung giải quyết ở client là chủ yếu.

React JS là dạng phát triển single page tức là suốt quá trình làm ta chỉ thực hiện trên một file HTML duy nhất là index.html.

Đặc biệt, React JS sử dụng DOM ảo với đầy đủ đặc tính của DOM thật giúp cải thiện hiệu năng của ứng dụng web. Ta có thể hiểu như sau, khi một node trên DOM ảo có sự thay đổi thì React JS sẽ tìm sự thay đổi bằng cách so sánh DOM ảo với DOM thật, sau khi tìm thấy node có sự thay đổi thì tiến hành cập nhật thay đổi của node đó lên DOM thật và không ảnh hưởng đến các node khác trên DOM thật.

3.2. JSX, Component, Fragment, Props, State

3.2.1. JSX

JSX (Javascript XML) cho phép HTML được phép viết trong Javascript và tiến hành chuyển các thẻ HTML thành các component của React. Biểu thức khi đưa vào JSX phải đặt trong cặp dấu ngoặc nhọn {}. Trong JSX ta có thể trả ra nhiều thành phần nhưng chúng phải được bọc (wrap) trong một thành phần container. Việc sử dụng JSX giúp xây dựng component dễ dàng và thực thi nhanh hơn, đồng thời giúp chương trình an toàn hơn vì các lỗi đều được phát hiện trong quá trình biên dịch.

3.2.2. Component

React JS tập trung giải quyết phần View trong mô hình MVC, vì vậy việc chia view thành nhiều phần nhỏ khác nhau gọi là component giúp việc tái sử dụng hiệu quả và bảo trì, quản lý dễ dàng thực hiện hơn. Component hoạt động như một hàm trả về các thành phần HTML.

Về cơ bản, khi định nghĩa một component trong React phải kế thừa (extends) các thành phần của React. Component và yêu cầu bắt buộc có phương thức render() trả về HTML.

Trong component có phương thức constructor() là nơi khởi tạo các thuộc tính của component, đồng thời phương thức này sẽ được gọi đầu tiên khi sử dụng component.

3.2.3. Fragment

Khi định nghĩa một component việc trả ra nhiều thành phần HTML là bình thường và các thành phần HTML phải được bọc trong một container. Tuy nhiên, ở một số trường hợp có thể phá vỡ quy định CSS của các thành con trong container và một số bất lợi khác cho lập trình viên. Vì vậy, React cung cấp cơ chế React Fragment giúp gom nhóm các thành phần con mà không cần sử dụng container bọc lại. Việc sử dụng React Fragment giúp tiết kiệm bộ nhớ và chương trình được thực thi nhanh hơn. Các thành phần con sẽ được bọc trong cặp <> </>.

Cú pháp Fragment.

```
class Hello extends React.Component {

render() {
```

```
    return (
    <>
    <h1>Test 1</h1>
    <h2>Test 2</h2>
    </>
    )
    }
}
```

3.2.4. Props

Props (properties) các đối số được truyền vào component như các thuộc tính. Tại các component con giá trị của props không được thay đổi. Trong component nếu có phương thức constructor thì props phải được truyền vào constructor và truyền bằng phương thức super.

3.2.5. State

State là đối tượng lưu giá trị các thuộc tính của component, khi muốn thay đổi giá trị của các thuộc tính trong component ta dùng phương thức this.setState(), khi phương thức this.setState() được gọi thì sẽ reload toàn bộ trang, thì lúc này trên server sẽ so sánh giữa DOM ảo và DOM thật, tìm kiếm phần thay đổi và tiến hành cập nhật phần thay đổi đó trên DOM thật. Một state chỉ có hiệu lực trên một component.

3.3. React CSS, React Bootstrap, React Form, React Events

3.3.1. React CSS

CSS (Cascading Style Sheets) là ngôn ngữ dùng để mô tả cách trình bày của tài liệu dạng HTML, cụ thể là mô tả cách hiển thị của các phần tử trên màn hình. Trong React, ta có thể viết CSS bằng các cách như thứ nhất, tạo các đối tượng chứa thông tin thiết lập CSS và gán cho thuộc tính style của thành phần HTML, thứ hai, ta có tạo một file .css riêng chứa thông tin thiết lập CSS, sau đó import vào component cần sử dụng.

3.3.2. React Bootstrap

Trong React, Bootstrap javascript được thay thế bằng React Bootstrap, đồng thời component của Bootstrap được xây dựng thành các React component không cần các thư viện liên quan như jQuery.

Cài đặt React Bootstrap bằng lệnh.

```
npm install react-bootstrap bootstrap@4.6.0
```

Sau đó import các component của React Bootstrap để sử dụng như sau.

```
import { Container, Row, Col } from "react-bootstrap"
```

3.3.3. React Form

Trong React, dữ liệu form thường được xử lý bởi các component khi tất cả dữ liệu lưu trong state.

Chú ý cần khởi động state trong constructor trước khi sử dụng. Ta có thể truy cập giá trị các trường bằng lệnh event.target.value.

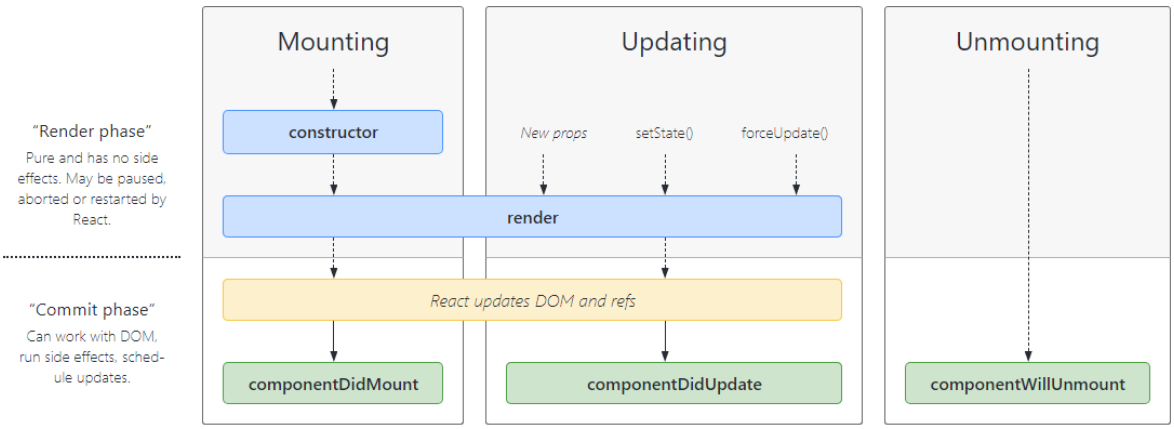
3.3.4. React Events

Trong React, khi một sự kiện xảy thì một phương thức xử lý sự kiện đó trong component sẽ được gọi. Một số event như onClick, onChange. Từ khóa this trong React đại diện cho component chứa phương thức, tuy nhiên khi định nghĩa phương thức bằng cách truyền thống thì từ khóa this đại diện cho bất kì đối tượng nào gọi nó. Vì vậy, khi định nghĩa phương thức bằng cách truyền thống ta phải dùng phương thức bind để kết buộc this vào component. Ngoài ra, nếu ta định nghĩa phương thức dạng arrow function thì không cần sử dụng bind [9].

3.4. Vòng đời của React

Mỗi component của React có 3 giai đoạn chính.

- Mounting: khi đặt một component vào DOM.
- Updating: khi component được update. Một component được update khi có thay đổi trong props hoặc state.
- Unmounting: khi một component được gỡ khỏi DOM.



Hình 3.1: Vòng đời của React

(Nguồn ảnh: https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/)

3.4.1. Mounting

Có 4 phương thức sẽ được gọi:

constructor(): gọi khi component được tạo.

render(): là phương thức bắt buộc và trả ra HTML cho DOM.

static getDerivedStateFromProps(): gọi ngay trước khi render các thành phần trong DOM.

componentDidMount(): gọi sau khi component được render.

3.4.2. Updating

Có 5 phương thức được gọi:

static getDerivedStateFromProps(): gọi khi component được cập nhật.

shouldComponentUpdate(): trả về giá trị boolean cho biết React có tiếp tục render hay không.

render(): gọi khi component cập nhật render lại HTML cho DOM.

getSnapshotBeforeUpdate(): được gọi trước khi props và state được cập nhật.

componentDidUpdate(): gọi sau khi được cập nhật.

3.4.3. Unmounting

React cung cấp một phương thức có sẵn khi component được unmounted là componentWillUnmount().

3.5. React Routers

Trong React, React Router được gọi là thư viện định tuyến tiêu chuẩn, nó giữ cho giao diện của ứng dụng web đồng bộ với URL trên trình duyệt. Đồng thời, giúp ta chuyển qua lại giữa các component mà không cần nạp lại trang. React có ba gói cho routing.

- react-router: cung cấp các thành phần routing chính cho ứng dụng.
- react-router-native: dùng cho các ứng dụng di động.
- react-router-dom: dùng cho các ứng dụng Web.

Cài đặt React Router bằng lệnh.

```
npm install react-router-dom
```

Các thành phần trong React Router:

BrowserRouter: xử lý các URL động.

HashRouter: xử lý các request tĩnh.

Route: chỉ định các component được render dựa trên path chỉ định.

Switch: chứa các Route để render các component kho path khớp.

Link: tạo liên kết đến URL chỉ định nhưng không nạp lại trang.

NavLink: tương tự Link nhưng cho phép thêm style cho active link.

Redirect: dùng chuyển hướng đến route khác trong ứng dụng, nhưng vẫn duy trì URL cũ.

3.6. React Hooks

Từ phiên bản 16.8 của React trở đi sẽ hỗ trợ thêm đặc trưng mới là Hooks giúp sử dụng các đặc trưng của React và state mà không phải tạo class. Hooks là các hàm móc vào state cũng như các đặc trưng của vòng đời React bằng component function.

Hàm useState() được sử dụng trong Hooks để tìm kiếm và thiết lập state.

Phương thức useState() yêu cầu hai giá trị.

- Giá trị 1: giá trị hiện tại của state.
- Giá trị 2: hàm dùng để cập nhật state.

Ngoài ra trong Hooks có Hooks effect tương đương với các phương thức componentDidMount(), componentDidUpdate(), componentWillUnmount(). Phương thức áp dụng của Hooks effect là useEffect().

3.7. React Axios

Axios là một HTTP client làm việc dựa trên promise cung cấp các API dễ dàng sử dụng cho phía trình duyệt và NodeJS.

Cài đặt React Axios bằng lệnh.

```
yarn add axios
```

Ví dụ sử dụng axios trong React.

Đầu tiên, tạo file APIs.js.

```
import axios from "axios";



export let endpoints = {

    "lobbies": "lobbies/",



export default axios.create({

    baseURL: 'http://127.0.0.1:8000/'

})
```

Sau đó, tạo file Lobbies.js để gọi api.

```
import { useEffect, useState } from "react"
import { Row } from "react-bootstrap"
import APIs, { endpoints } from "../configs/APIs"
import LobbiesCard from "./LobbieCard"


export default function Home() {
    const [lobbies, setLobbies] = useState([])
    useEffect(() => {
        let loadLobbies = async () => {
            try {
                let res = await
APIs.get(endpoints['lobbies'])
                setLobbies(res.data)


            } catch (err) {
                console.error(err)
```

```
            }
        }
        loadLobbies()
    }, [])



    return (
        <>
            <h1 style={{ textAlign: "center" }}> Sảnh cưới
tại trung tâm SHANE</h1>
            <Row>
                {lobbies.map((c, index) => <LobbiesCard
obj={c} key={index} />)}
            </Row>
        </>
    )
}
```

3.8. React Cookies

Khi một trang web được một người dùng ghé thăm thì cookie sẽ lưu thông tin của người đó và nhận diện được người dùng khi truy cập lại trang web ở lần truy cập sau. React Cookies hỗ trợ giúp việc tương tác với cookie trong React trở nên dễ dàng. Dùng lệnh sau để cài đặt React Cookies.

```
npm install react-cookies --save
```

Sử dụng bằng cách import vào component cần dùng.

```
import cookies from 'react-cookies
```

Một số thuộc tính quan trọng của React Cookies:

cookies.load(name): nạp giá trị một cookie.

cookies.loadAll(): nạp tất cả các cookies.

cookies.select([regex]): tìm cookie có tên khớp với một biểu thức chính quy.

cookies.save(name, value, [options]): lưu một cookie.

remove(name, [options]): xoá một cookie.

3.9. React Redux

Redux là thư viện dùng để cập nhật và quản lý các state sử dụng các sự kiện gọi là actions. Redux có một store trung tâm sử dụng trong toàn ứng dụng dùng để lưu trữ

state. Một state của ứng dụng trong Redux không thể thay đổi trực tiếp, khi muốn thay đổi state trong Redux thì phải phát (emit) một action, đây là đối tượng mô tả những gì sẽ thay đổi.

Cài đặt Redux bằng lệnh.

```
npm install --save redux
```

Cài Redux cùng với ứng dụng React.

```
npm install --save react-redux
```

3.9.1. Action

Action là một đối tượng JS với thuộc tính type là loại action chứa giá trị là một chuỗi. Ngoài ra, Action cũng có thể chứa các thông tin khác được đặt trong thuộc tính payload của đối tượng Action.

3.9.2. Reducer

Một hàm bao gồm action và state gọi là Reducer và đây cũng là nơi duy nhất được thực hiện khi cập nhật state trong Redux. Một Reducer chứa state hiện tại và đối tượng action để cập nhật state và trả về state khi cần thiết.

Cú pháp tạo Reducer.

```
(state=initState, action) => newState
```

3.9.3. Store

State hiện hành của ứng dụng được chứa tại Store. Sử dụng phương thức createStore để tạo Store, cần phải truyền reducer khi tạo Store.

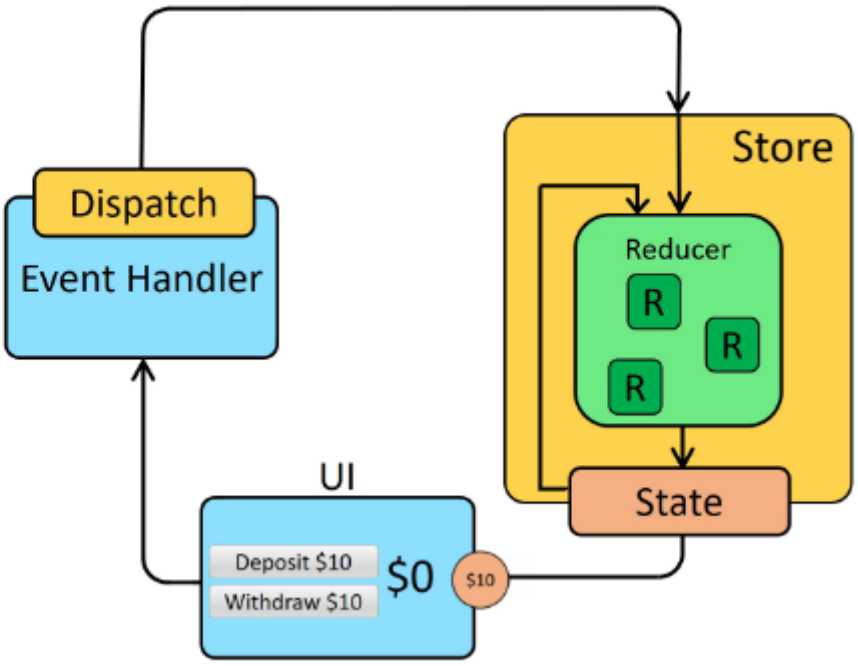
Các phương thức của Store:

getState(): trả state hiện tại.

subscribe(): đăng ký hàm callback sẽ được gọi mỗi khi store được cập nhật.

dispatch(): được gọi, store chạy reducer nhận state được cập nhật và chạy các hàm callback được subscribe để cập nhật UI.

3.9.4. Cơ chế hoạt động của Redux



Hình 3.2: Cơ chế hoạt động của Redux

(Nguồn: https://redux.js.org/tutorials/essentials/part-1-overview-concepts)

Khi user tương tác trên ứng dụng web sẽ dẫn đến xảy ra một action. Lúc này, hàm reducer sẽ được gọi cùng với state hiện tại, sau đó action được truyền vào reducer và state mới được trả về.

Các view thực thi sẽ được store thông báo bằng cách gọi hàm callback của chúng. React component đọc dữ liệu từ store và truyền action tới store để cập nhật dữ liệu. Subcribe của store được Redux theo dõi để kiểm tra dữ liệu của component nào thay đổi sẽ nạp lại component đó.

3.10. Thiết lập chương trình đầu tiên phía client

Đầu tiên, ta cần cài đặt môi trường để sử dụng ReactJS.

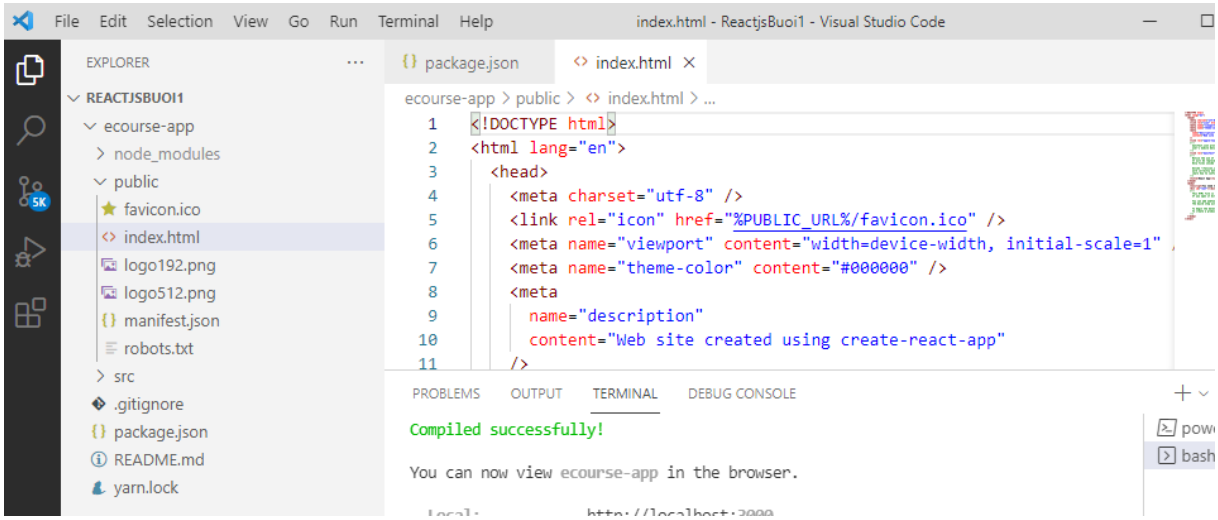
Cài đặt nodejs: https://nodejs.org/en/download/

Tiếp theo, ta tiến hành cài công cụ quản lý thư viện bằng lệnh.

```
npm install -g yarn.
```

Kế tiếp, tạo project đầu tiên bằng lệnh

```
yarn create react-app course-app
```



Hình 3.3: Project React JS đầu tiên

Các file và thư mục cần lưu trong cấu trúc của project ReactJS.

Trong file index.html thẻ có id=”root” là nơi thực thi tất cả mọi thứ bạn muốn thể hiện trên website.

File package.json là nơi chứa các thư viện, lệnh hiện tại project reactjs đang sử dụng.

Thư mục node_modules là nơi chứa các thư viện default của project reactjs, nếu có cài thêm thư viện mới thì thư viện đó sẽ được lưu vào thư mục node_modules và được khai báo ở file package.json.

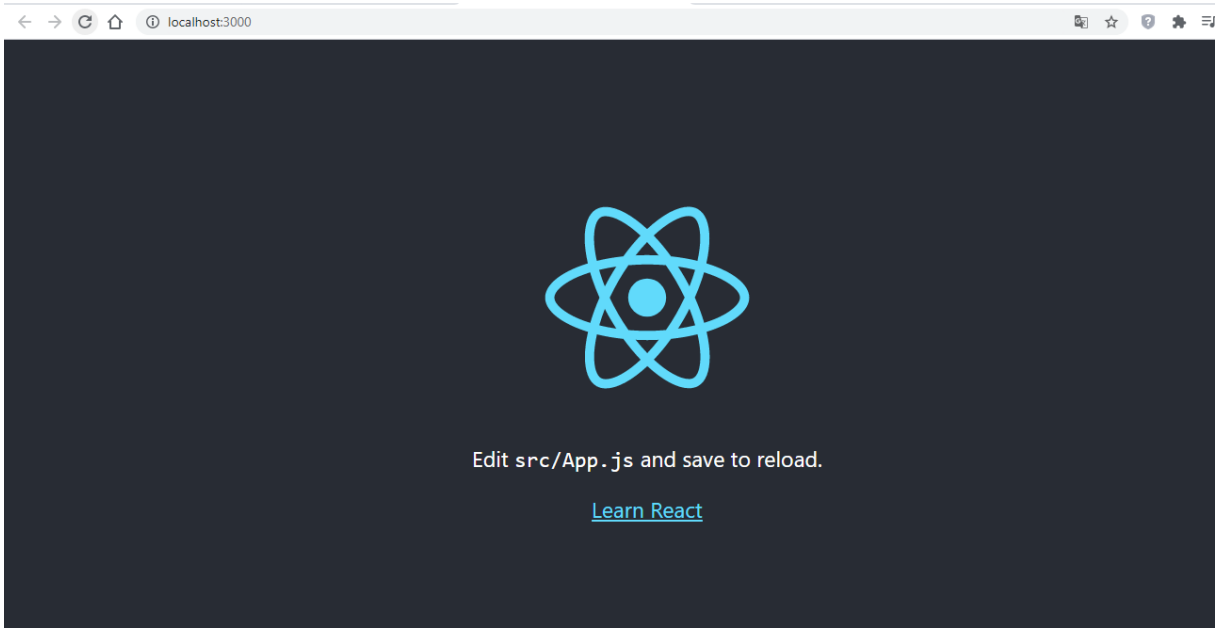
File robots.txt là file cấu hình các đường dẫn, domain mà bạn không muốn công cụ tìm kiếm của google quan tâm tới, khi bạn cấu hình trong file này thì đường dẫn, domain đó sẽ bị ẩn đi.

SRC là thư mục để chạy source code của reacjs.

File App.js là nơi chứa các component dưới dạng các function.

Chạy project reactjs bằng lệnh sau.

```
yarn start
```



Hình 3.4: Giao diện React đầu tiên

Tiếp tục, tiến hành tạo một component đơn giản để hiển thị lên giao diện của React.

Bước 1: tạo một file User.js trong thư mục src

Bước 2: trong file User.js thêm các dòng sau.

```
import React from 'react'


class User extends React.Component {
    render() {
        return (
            <div>
                <h1> WELCOME TO OUR WEBSITE!!!</h1>
                <h2>E-COURSE</h2>
            </div>
        )
    }

}
export default User // phải có dòng này thì bên trang
index.html mới import được.
```

Bước 3: trong thư mục src trong file index.js thay đổi như sau.

```
import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

//import App from './App';
```

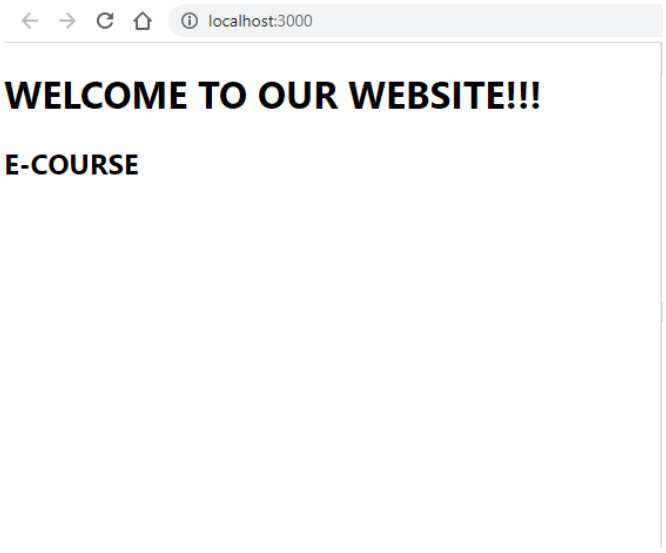
```
import reportWebVitals from './reportWebVitals';

import User from './User';


ReactDOM.render(
  <React.StrictMode>
    <User /> // khai báo component
  </React.StrictMode>,
  document.getElementById('root')
);


// If you want to start measuring performance in your app,
 pass a function
// to log results (for example: reportWebVitals(console.lo
g))
// or send to an analytics endpoint. Learn more: https://b
it.ly/CRA-vitals
reportWebVitals();
```

B4: trong terminal bạn phải cd đến app react và thực lệnh Yarn start để xem kết quả.



Hình 3.5: Kết quả của chương trình React đầu tiên

Chương 4. HỆ THỐNG QUẢN LÝ NHÀ HÀNG TIỆC CƯỚI

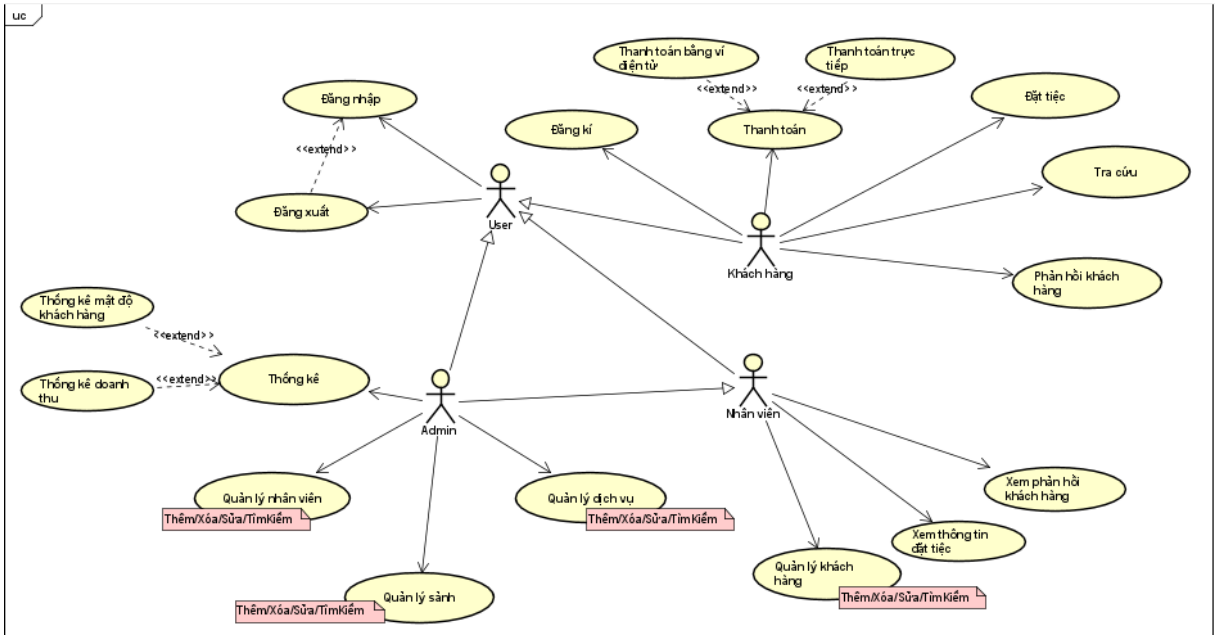
Đây là chương em sẽ trình bày tổng quan về các lược đồ phân tích và thiết kế của hệ thống. Đồng thời, em sẽ mô tả chi tiết hoạt động của từng chức năng.

4.1. Giới thiệu hệ thống

Hệ thống quản lý nhà hàng tiệc cưới là trang web cho phép người dùng nắm bắt các thông tin về các món ăn, dịch vụ, sảnh cưới nhà hàng, đồng thời có thể đặt tiệc online. Các chức năng của hệ thống gồm đăng nhập, đăng kí, xem thông tin, bình luận, … . Hơn thế nữa, hệ thống còn cung cấp trang quản trị admin giúp người quản trị quản lý tài khoản người dùng, quản lý dữ liệu và thống kê dữ liệu, giúp công việc quản lý hiệu quả hơn.

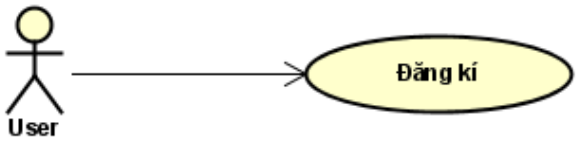
4.2. Phân tích thiết kế hệ thống

4.2.1. Lược đồ Use Case tổng quát



Hình 4.1: Lược đồ Use Case tổng quát

4.2.2. Use Case đăng kí



Hình 4.2: Lược đồ Use Case đăng kí

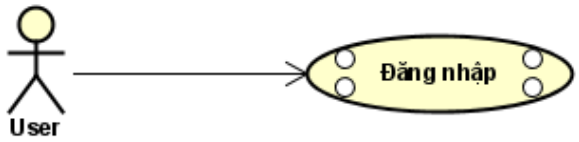
4.2.2.1. Đặc tả Use Case đăng kí

Use Case ID	1
Tên Use Case	Đăng kí.
Mô tả Use Case	Use Case này cho phép người dùng tạo tài khoản để truy cập và sử dụng dịch vụ của hệ thống.
Actor chính	Người dùng (User).
Actor phụ	Không có.
Tiền điều kiện	User nhập đúng các thông tin hệ thống yêu cầu. Có kết nối internet.
Hậu điều kiện	Hệ thống hiển thị giao diện thông báo đến người dùng biết họ đã đăng kí thành công.
Luồng hoạt động	1. Người dùng vào hệ thống và chọn chức năng đăng kí. 2. Hệ thống yêu cầu người dùng nhập các thông tin cần thiết. 3. Người dùng nhập thông tin theo yêu cầu hệ thống. 4. Hệ thống kiểm tra thông tin.

	5. Hệ thống hiển thị giao diện đăng kí thành công.
Luồng thay thế	Không có.
Luồng ngoại lệ	Ở bước 4 nếu hệ thông kiểm tra thông tin có chưa phù hợp yêu cầu và hiển thị thông báo yêu cầu nhập lại thông tin. • Người dùng nhập lại thông tin như bước 3. • Người dùng chọn hủy đăng kí -> Use Case dừng lại.

Bảng 4.1: Đặc tả Use Case "Đăng kí"

4.2.3. Use Case đăng nhập



Hình 4.3: Lược đồ Use Case đăng nhập

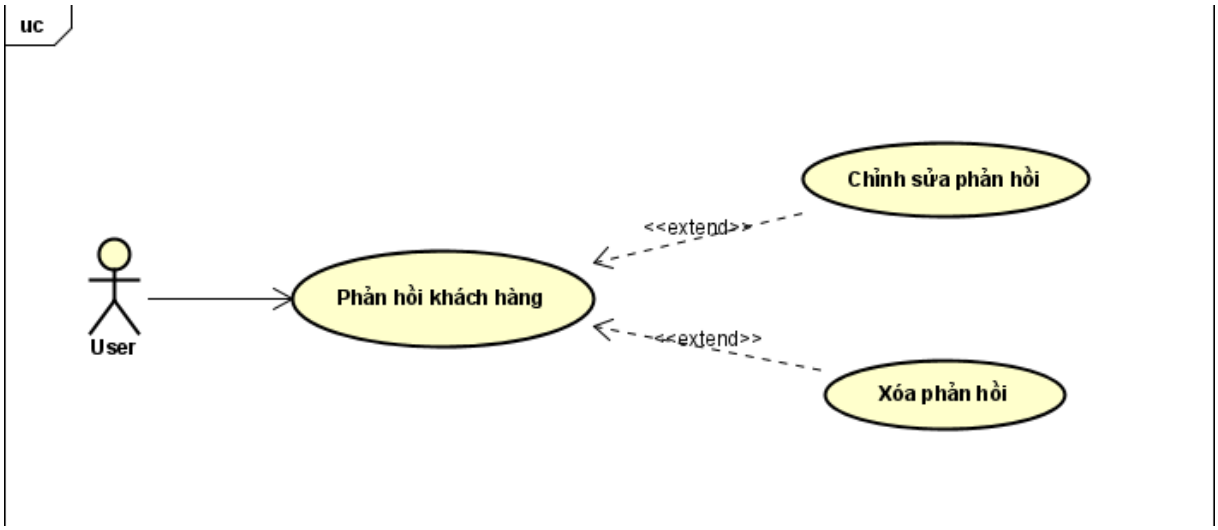
4.2.3.1. Đặc tả Use Case đăng nhập

Use Case ID	2
Tên Use Case	Đăng nhập.
Mô tả Use Case	Use Case này cho phép người dùng đăng nhập và sử dụng các chức năng của hệ thống.
Actor chính	Người dùng (User).
Actor phụ	Không có.

Tiền điều kiện	User đã tạo tài khoản. Có kết nối internet.
Hậu điều kiện	Hiển thị giao diện cho biết người dùng đăng nhập thành công.
Luồng hoạt động	1. Người dùng truy cập hệ thống và chọn chức năng đăng nhập. 2. Hệ thống hiển thị giao diện đăng nhập. 3. Người dùng nhập thông tin tài khoản và mật khẩu. 4. Hệ thống kiểm tra thông tin tài khoản. 5. Hệ thống hiển thị thông báo đăng nhập thành công và chuyển hướng đến trang chủ của hệ thống.
Luồng thay thế	Không có.
Luồng ngoại lệ	Ở bước 4 nếu hệ thống kiểm tra thông tin có chưa phù hợp và hiển thị thông báo sai tên đăng nhập hoặc mật khẩu, yêu cầu nhập lại. • Người dùng nhập lại tài khoản mật khẩu giống bước 3.

Bảng 4.2: Đặc tả Use Case "Đăng nhập"

4.2.4. Use Case phản hồi khách hàng



Hình 4.4: Lược đồ Use Case phản hồi khách hàng

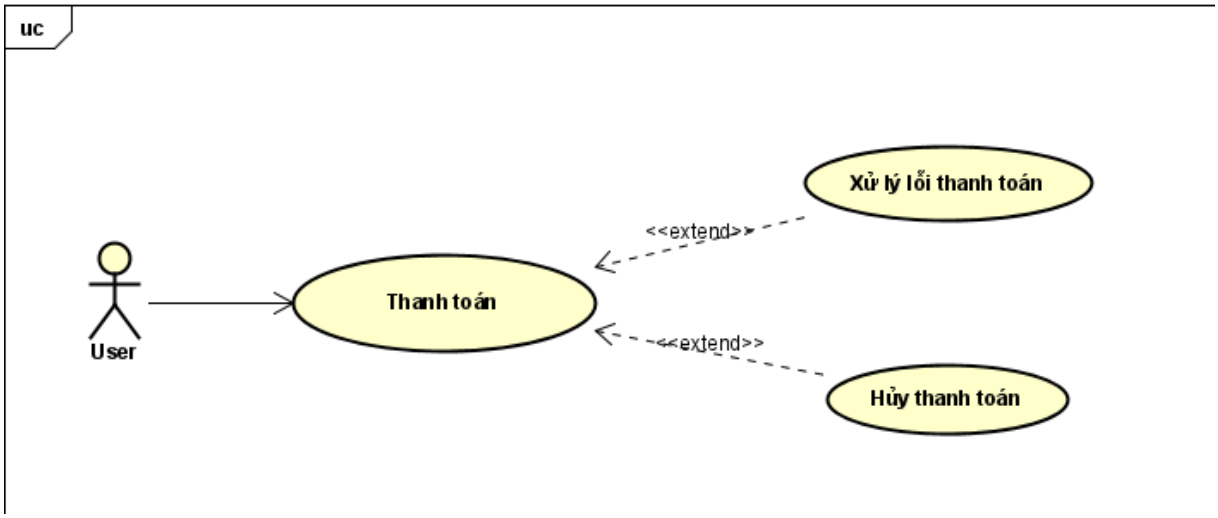
4.2.4.1. Đặc tả Use Case phản hồi khách hàng

Use Case ID	3
Tên Use Case	Phản hồi khách hàng.
Mô tả Use Case	Use Case này cho phép người dùng phản hồi về dịch vụ. Có thể chỉnh sửa hoặc xóa phản hồi.
Actor chính	Người dùng (User).
Actor phụ	Không có.
Tiền điều kiện	User đã đăng nhập
Hậu điều kiện	Hiển thị thông tin bài viết sau khi người dùng thực hiện phản hồi.
Luồng hoạt động	1. Người dùng nhấn mục bình luận trên món ăn, thực đơn, sảnh, đồ uống, dịch vụ và thực hiện phản hồi. 2. Hệ thống hiển thị giao diện cập nhật bài viết sau khi người dùng thực hiện phản hồi.
Luồng thay thế	Không có.

Luồng ngoại lệ	Ở bước một, nếu như bài viết phản hồi đã bị xóa nhưng hệ thống chưa cập nhật lên giao diện cho người dùng thì sẽ thông báo lỗi.

Bảng 4.3: Đặc tả Use Case "Phản hồi khách hàng"

4.2.5. Use Case thanh toán



Hình 4.5: Lược đồ Use Case thanh toán

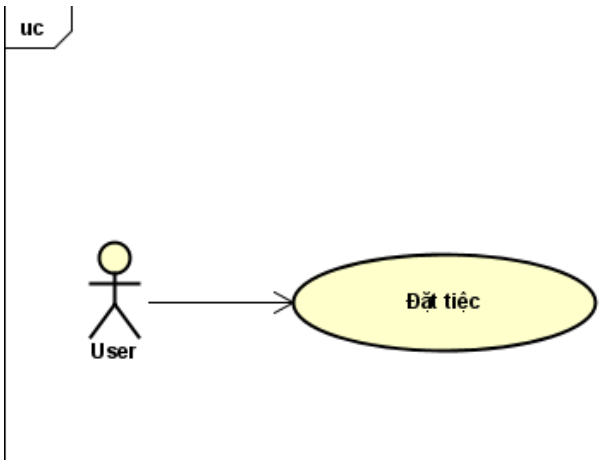
4.2.5.1. Đặc tả Use Case thanh toán

Use Case ID	4
Tên Use Case	Thanh toán.
Mô tả Use Case	Use Case này cho phép người dùng thanh toán tiền cho các hóa đơn đặt tiệc của mình.
Actor chính	Người dùng (User).
Actor phụ	Ví điện tử Momo
Tiền điều kiện	User đã đăng nhập. Tài khoản có liên kết với ví Momo.
Hậu điều kiện	Thông báo thông tin giao dịch cho người dùng.

Luồng hoạt động	1. Người dùng chọn chức năng thanh toán. 2. Hệ thống yêu cầu người dùng nhập thông tin giao dịch. 3. Hệ thống kiểm tra thông tin và chuyển đến ví điện tử Momo. 4. Ví điện tử Momo yêu cầu nhập mật khẩu xác nhận giao dịch. 5. Ví Momo kiểm tra thông tin và trả về kết quả thanh toán cho hệ thống. 6. Hệ thống hiển thị giao diện thanh toán thành công cho người dùng.
Luồng thay thế	Không có.
Luồng ngoại lệ	Ở bước 2, nếu người dùng không nhập thông tin hoặc nhập sai giao dịch thì hiển thị giao diện yêu cầu người dùng nhập lại thông tin hoặc chọn thoát. Ở bước 4, nếu người dùng nhập mật khẩu sai ba lần thì giao dịch bị hủy.

Bảng 4.4: Đặc tả Use Case "Thanh toán"

4.2.6. Use Case đặt tiệc



Hình 4.6: Lược đồ Use Case đặt tiệc

4.2.6.1. Đặc tả Use Case đặt tiệc

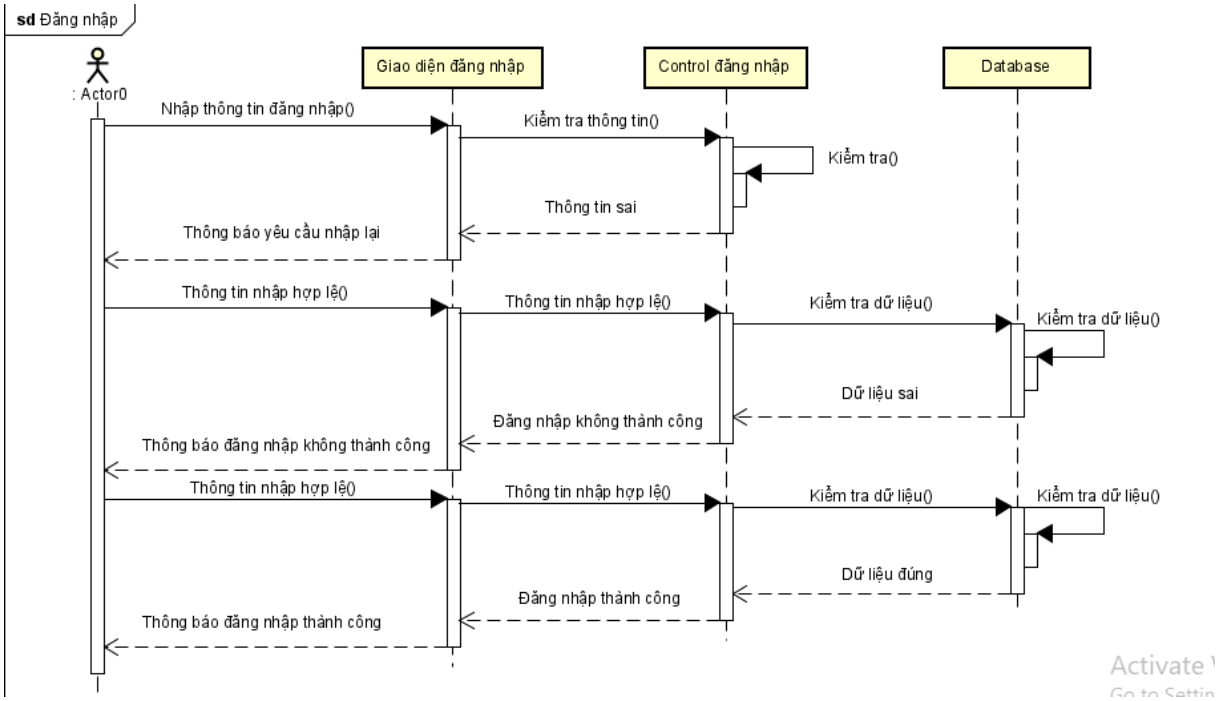
Use Case ID	5
Tên Use Case	Đặt tiệc

Mô tả Use Case	Use Case này cho phép người dùng chọn các thực đơn, đồ uống, dịch vụ, sảnh và thêm vào phần thông tin đặt tiệc
Actor chính	Người dùng (User).
Actor phụ	Không.
Tiền điều kiện	User đã đăng nhập.
Hậu điều kiện	Thông báo và hiển thị các thông tin đặt tiệc.
Luồng hoạt động	1. Người dùng chọn các các thực đơn, đồ uống, dịch vụ, sảnh và thêm vào phần thông tin đặt tiệc 2. Hệ thống lưu thông tin tiệc và khi người dùng chọn vào biểu tượng giỏ sẽ hiển thị thông tin đặt tiệc. 3. Người dùng kiểm tra thông tin đặt tiệc đúng và nhấn đặt tiệc. 4. Hệ thống lưu thông tin đặt tiệc và chuyển sang trang thanh toán.
Luồng thay thế	Không có.
Luồng ngoại lệ	Ở bước 2, nếu người dùng không chọn thông tin đã chọn trước đó có thể bỏ chọn và chọn thông tin khác.

Bảng 4.5: Đặc tả Use Case "Đặt tiệc"

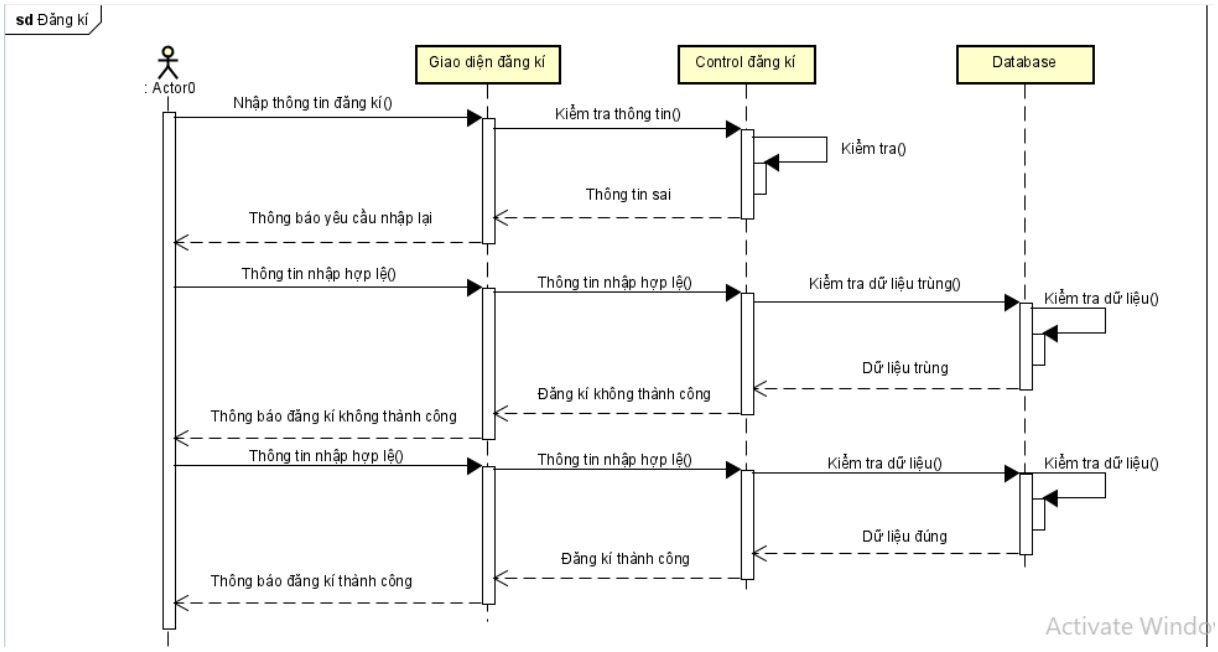
4.3. Lược đồ tuần tự

4.3.1. Lược đồ tuần tự chức năng đăng nhập



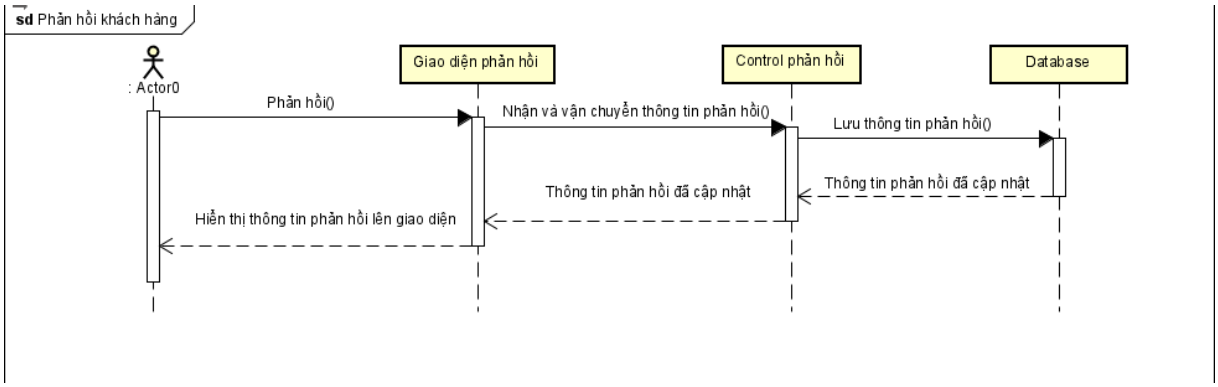
Hình 4.7: Lược đồ tuần tự chức năng "Đăng nhập"

4.3.2. Lược đồ tuần tự chức năng đăng kí



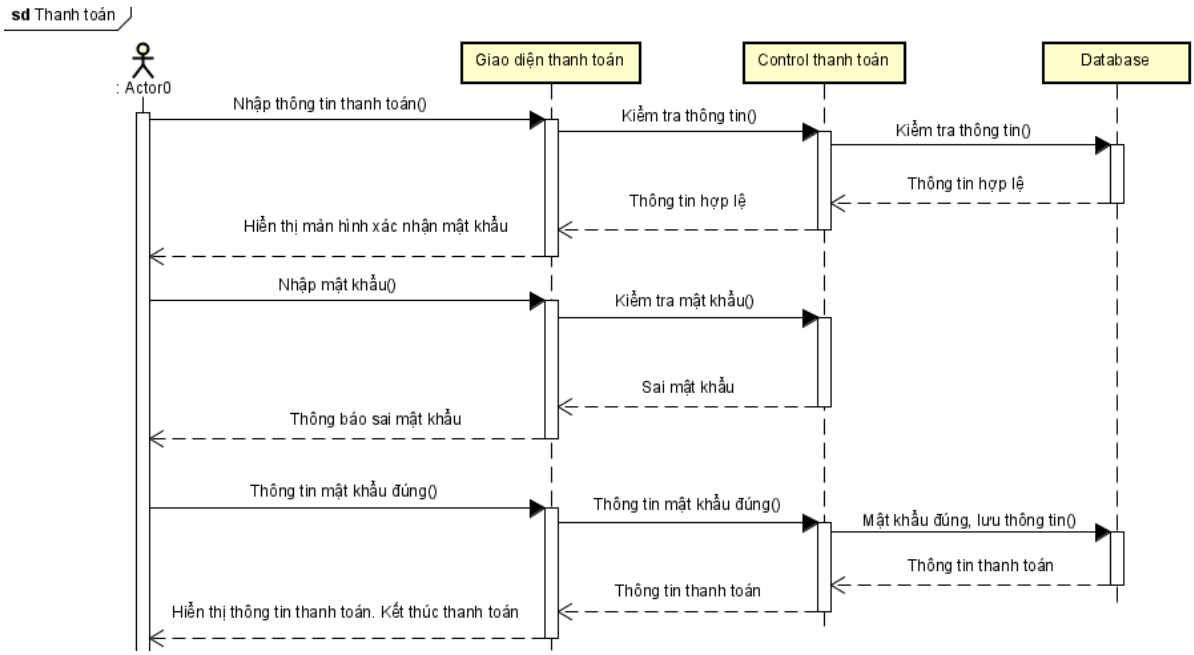
Hình 4.8: Lược đồ tuần tự chức năng "Đăng kí"

4.3.3. Lược đồ tuần tự chức năng phản hồi khách hàng



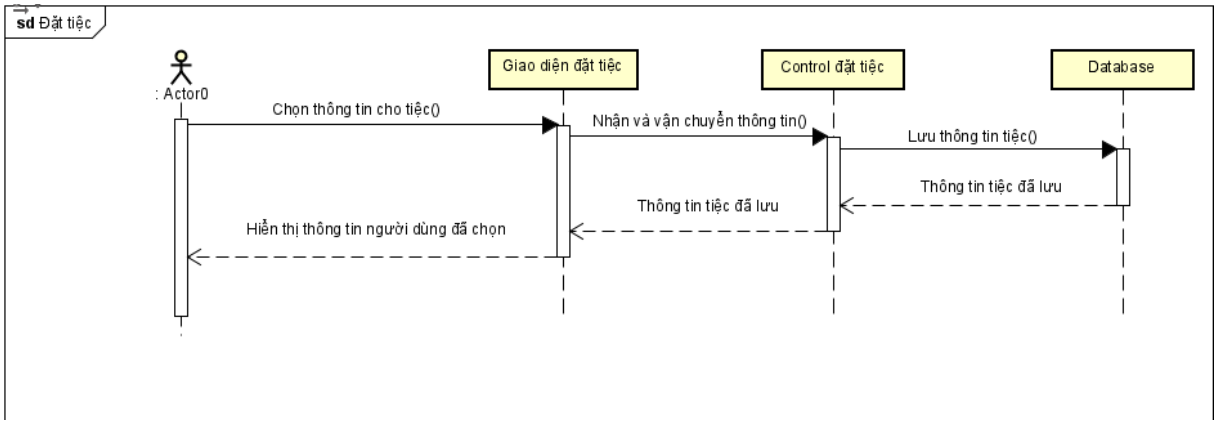
Hình 4.9: Lược đồ tuần tự chức năng "Phản hồi khách hàng"

4.3.4. Lược đồ tuần tự chức năng thanh toán



Hình 4.10: Lược đồ tuần tự chức năng "Thanh toán"

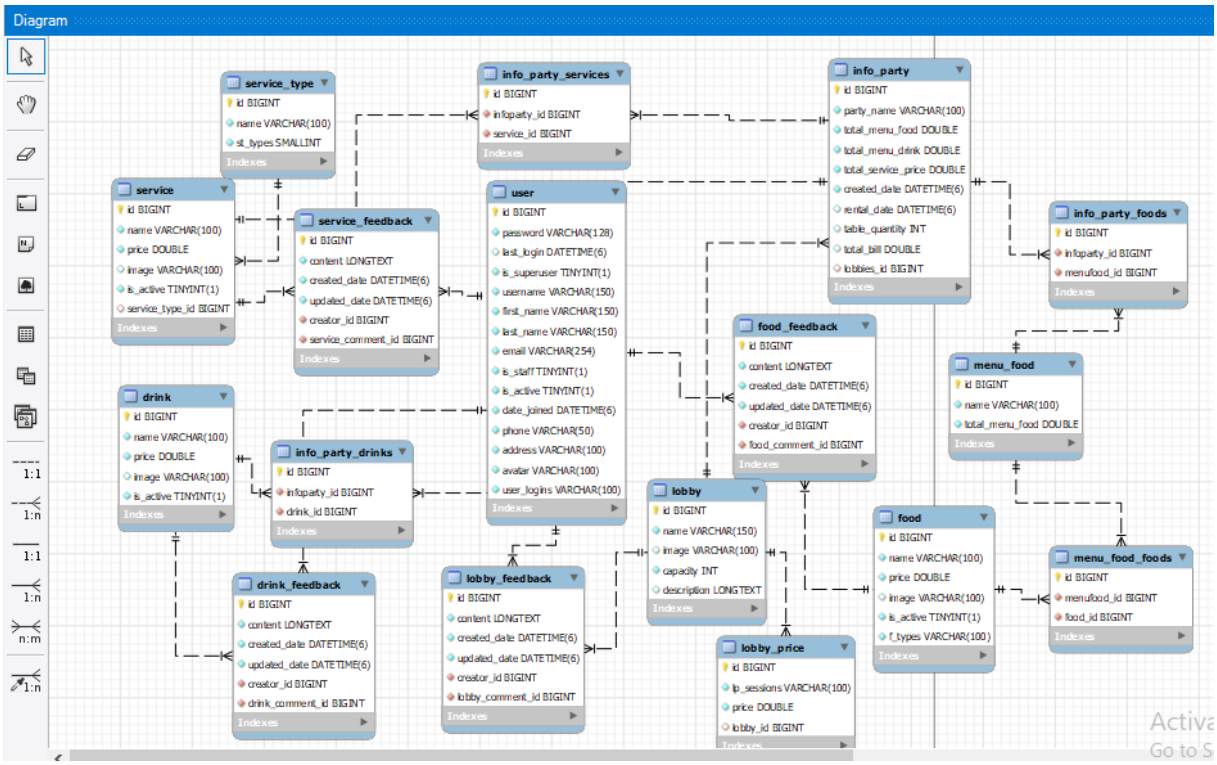
4.3.5. Lược đồ tuần tự chức năng đặt tiệc



Hình 4.11: Lược đồ tuần tự chức năng "Đặt tiệc"

4.4. Thiết kế cơ sở dữ liệu

4.4.1. Lược đồ cơ sở dữ liệu tổng quát



Hình 4.12: Lược đồ cơ sở dữ liệu tổng quát

4.4.2. Các thành phần lưu trữ và ràng buộc quan trọng

Bảng User				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	username	VARCHAR	Not Null	Tên đăng nhập

3	password	VARCHAR	Not Null	Mật khẩu
4	first_name	VARCHAR	Not Null	Tên người dùng
5	last_name	VARCHAR	Not Null	Họ và tên đệm người dùng
6	email	VARCHAR	Not Null	Email người dùng
7	phone	VARCHAR	Not Null	Số điện thoại người dùng
8	avatar	VARCHAR	Not Null	Đường dẫn lưu ảnh đại diện người dùng
9	user_logins	VARCHAR	Not Null	Loại người dùng

Bảng 4.6: Bảng User

Bảng Lobby				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	name	VARCHAR	Not Null	Tên sảnh
3	image	VARCHAR	Not Null	Đường dẫn lưu ảnh của sảnh
4	capacity	INT	Not Null	Sức chứa của sảnh
5	description	VARCHAR	Not Null	Mô tả thông tin sảnh

Bảng 4.7: Bảng Lobby

Bảng Lobby FeedBack				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**

1	id	INT	Khóa chính	Khóa chính
2	content	LONGTEXT		Nội dung phản hồi
3	created_date	DATETIME	Not Null	Ngày tạo phản hồi
4	updated_date	DATETIME	Not Null	Ngày cập nhật phản hồi
5	creator	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng User
6	lobby_comment	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng Lobby

Bảng 4.8: Bảng Lobby FeedBack

Bảng Lobby Price				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	price	DOUBLE	Not Null	Giá sảnh
3	lp_sessions	VARCHAR	Not Null	Phân loại giá theo buổi
4	lobby	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng Lobby

Bảng 4.9: Bảng Lobby Price

Bảng Food				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính

2	name	VARCHAR	Not Null	Tên món ăn
3	price	DOUBLE	Not Null	Giá món ăn
4	image	VARCHAR	Not Null	Đường dẫn ảnh món ăn
5	is_active	BOOLEAN	Not Null	Trạng thái món ăn
6	f_types	VARCHAR	Not Null	Loại món ăn

Bảng 4.10: Bảng Food

Bảng Food FeedBack				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	content	LONGTEXT		Nội dung phản hồi
3	created_date	DATETIME	Not Null	Ngày tạo phản hồi
4	updated_date	DATETIME	Not Null	Ngày cập nhật phản hồi
5	creator	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng User
6	food_comment	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng Food

Bảng 4.11: Bảng Food FeedBack

Bảng MenuFood				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính

2	name	VARCHAR	Not Null	Tên thực đơn setup sẵn
3	total_menu_food	DOUBLE	Not Null	Giá tiền của thực đơn
4	foods	MANYTOMANY	MANYTOMANY	Đại diện cho bảng trung gian kết hợp từ hai bảng Food và MenuFood

Bảng 4.12: Bảng MenuFood

Bảng Drink				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	name	VARCHAR	Not Null	Tên đồ uống
3	price	DOUBLE	Not Null	Giá đồ uống
4	image	VARCHAR	Not Null	Đường dẫn ảnh đồ uống
5	is_active	BOOLEAN	Not Null	Trạng thái đồ uống

Bảng 4.13: Bảng Drink

Bảng Drink FeedBack				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	content	LONGTEXT		Nội dung phản hồi
3	created_date	DATETIME	Not Null	Ngày tạo phản hồi

4	updated_date	DATETIME	Not Null	Ngày cập nhật phản hồi
5	creator	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng User
6	drink_comment	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng Drink

Bảng 4.14: Bảng Drink FeedBack

Bảng ServiceType				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	name	VARCHAR	Not Null	Tên loại dịch vụ
3	st_types	INT	Not Null	Phân loại tiệc cưới hay tiệc thường

Bảng 4.15: Bảng ServiceType

Bảng Service				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	name	VARCHAR	Not Null	Tên dịch vụ
3	price	DOUBLE	Not Null	Giá dịch vụ
4	image	VARCHAR	Not Null	Đường dẫn ảnh dịch vụ
5	is_active	BOOLEAN	Not Null	Trạng thái món ăn

6	service_type	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng ServiceType

Bảng 4.16: Bảng Service

Bảng Service FeedBack				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	content	LONGTEXT		Nội dung phản hồi
3	created_date	DATETIME	Not Null	Ngày tạo phản hồi
4	updated_date	DATETIME	Not Null	Ngày cập nhật phản hồi
5	creator	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng User
6	service_comment	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng Service

Bảng 4.17: Bảng Service FeedBack

Bảng InfoParty				
STT	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
1	id	INT	Khóa chính	Khóa chính
2	party_name	VARCHAR	Not Null	Tên buổi tiệc
3	total_menu_food	DOUBLE	Not Null	Tổng tiền thực đơn
4	total_menu_drink	DOUBLE	Not Null	Tổng tiền đồ

				uống
5	total_service_price	DOUBLE	Not Null	Tổng tiền dịch vụ
6	table_quantity	INT	Not Null	Số lượng bàn
7	rental_date	DATETIME	Not Null	Ngày thuê
8	created_date	DATETIME	Not Null	Ngày tạo thông tin đặt tiệc
9	total_bill	DOUBLE	Not Null	Tổng tiền của tiệc
10	lobbies	FOREIGNKEY	Khóa ngoại	Khóa ngoại tham khảo đến bảng Lobby
11	foods	MANYTOMANY	MANYTOMANY	Đại diện cho bảng trung gian kết hợp từ hai bảng MenuFood và InfoParty
12	drinks	MANYTOMANY	MANYTOMANY	Đại diện cho bảng trung gian kết hợp từ hai bảng Drink và InfoParty
13	services	MANYTOMANY	MANYTOMANY	Đại diện cho bảng trung gian kết hợp từ hai bảng Service và InfoParty

Bảng 4.18: Bảng InfoParty

4.5. Hệ thống quản lý nhà hàng tiệc cưới

Trong phần này em sẽ trình những API quan trọng trong hệ thống. Mỗi API bao gồm: URL, phương thức gọi API, dữ liệu đầu vào (body data), trạng thái phản hồi từ server (response status) và dữ liệu trả ra (response data). Đồng thời, em sẽ trình bày giao diện hiển thị phía client khi api được gọi.

4.5.1. Các API của hệ thống

4.5.1.1. API chức năng đăng kí

Url: /users/

Method: POST

Body data:

```
{
 "id": int,
 "first_name": "Thanh”,
 "last_name": "Do Cao",
 "email": "ctdo55@gmail.com",
"username": "thanh",
 "password": "Thait123$",
 "avatar": "string",
"user_logins": "string"
}
```

Response status: 201 – Created

Response data:

```
{
 "first_name": "Thanh”,
 "last_name": "Do Cao",
```

```
 "email": "ctdo55@gmail.com",
"username": "thanh",
 "password": "Thait123$",
 "avatar": "string",
"user_logins": "string"
}
```

4.5.1.2. API chức năng đăng nhập

Url: /oauth2-info/

Method: POST

Body data:

```
{
 "username": "thanh",
 "password": "Thait123$",
 "client_id": "abcnmlxyz",
 "client_secret": "quertyuio",
 "grant_type": "password",
}
```

Response status: 200 – OK

Response data:

```
{
 "access_token": "string",
 "expires_in": 36000,
 "token_type": "Bearer",
 "scope": "read write",
```

```
 "refresh_token": "string"
}
```

4.5.1.3. API lấy thông tin người dùng đăng nhập hiện tại

Url: /users/current-user/

Method: GET

Body data:

```
{
 "username": "string",
 "password": "string",
}
```

Request headers

```
{
 "Authorization": "Bearer <access-token>"
}
```

Response status: 200 – OK

Response data:

```
{
 "id": int,
 "first_name": "string",
 "last_name": "string",
 "email": "string",
 "username": "string",
 "avatar": "string",
"user_logins": "string"
```

```
}
```

4.5.1.4. API chức năng lấy danh sách sảnh

Url: /lobbies/

Method: GET

Body data: None

Response status: 200 – OK

Response data:

```
{
  "count": 0,
  "next": "string",
  "previous": "string",
  "results": [
    {
      "id": 0,
      "name": "string",
      "image": "string",
      "capacity": 0,
      "description": "string",
      "lobby_prices": [
        {
          "id": 0,
          "lp_sessions": "Morning",
          "price": 0,
          "lobby": 0
        }
```

```
        ]
      }
    ]
}
```

4.5.1.5. API chức năng lấy danh sách thực đơn

Url: /lobbies/

Method: GET

Body data: None

Response status: 200 – OK

Response data:

```
{
  "count": 0,
  "next": "string",
  "previous": "string",
  "results": [
    {
      "id": 0,
      "name": "string",
      "total_menu_food": 0,
      "foods": [
        {
          "id": 0,
          "name": "string",
          "price": 0,
          "image": "string",
```

```
            "f_types": "Appetizer"
          }
        ]
      }
    ]
}
```

4.5.1.6. API chức năng lấy danh sách món ăn

Url: /foods/

Method: GET

Body data: None

Response status: 200 – OK

Response data:

```
{
  "count": 0,
  "next": "string",
  "previous": "string",
  "results": [
    {
      "id": 0,
      "name": "string",
      "price": 0,
      "image": "string",
      "is_active": true,
      "f_types": "Appetizer"
    }
```

```
  ]
}
```

4.5.1.7. API chức năng lấy danh sách đồ uống

Url: /drinks/

Method: GET

Body data: None

Response status: 200 – OK

Response data:

```
{
  "count": 0,
  "next": "string",
  "previous": "string",
  "results": [
    {
      "id": 0,
      "name": "string",
      "price": 0,
      "image": "string",
      "is_active": true
    }
  ]
}
```

4.5.1.8. API chức năng lấy danh mục dịch vụ

Url: /service_types/

Method: GET

Body data: None

Response status: 200 – OK

Response data:


```
{
  "count": 0,
  "next": "string",
  "previous": "string",
  "results": [
    {
      "id": 0,
      "name": "string",
      "st_types": 0,
      "services": [
        {
          "id": 0,
          "name": "string",
          "price": 0,
          "image": "string",
          "is_active": true,
          "service_type": 0
        }
      ]
    }
  ]
}
```

4.5.1.9. API chức năng lấy danh sách dịch vụ

Url: /services/

Method: GET

Body data: None

Response status: 200 – OK

Response data:


```
{
  "count": 0,
  "next": "string",
  "previous": "string",
  "results": [
    {
      "id": 0,
      "name": "string",
      "price": 0,
      "image": "string",
      "is_active": true,
      "service_type": 0
    }
  ]
}
```


4.5.1.10. API chức năng lấy hóa đơn

Url: /info_parties/{id}/

Method: GET

Body data: None

Response status: 200 – OK

Response data:


```
{
  "id": 0,
  "party_name": "string",
  "lobbies": 0,
  "foods": [
    0
  ],
  "drinks": [
    0
  ],
  "services": [
    0
  ],
  "total_menu_food": 0,
  "total_menu_drink": 0,
  "total_service_price": 0,
  "table_quantity": 0,
  "rental_date": "2021-12-05T05:48:46.838Z",
  "created_date": "2021-12-05T05:48:46.838Z",
  "total_bill": 0
}
```


4.5.1.11. API chức năng tạo hóa đơn

Url: /services/

Method: GET

Body data:

```
{
  "party_name": "string",
  "lobbies": 0,
  "foods": [
    0
  ],
  "drinks": [
    0
  ],
  "services": [
    0
  ],
  "total_menu_food": 0,
  "total_menu_drink": 0,
  "total_service_price": 0,
  "table_quantity": 0,
  "rental_date": "2021-12-05T05:43:41.620Z",
  "total_bill": 0
}
```

Response status: 201 – Created

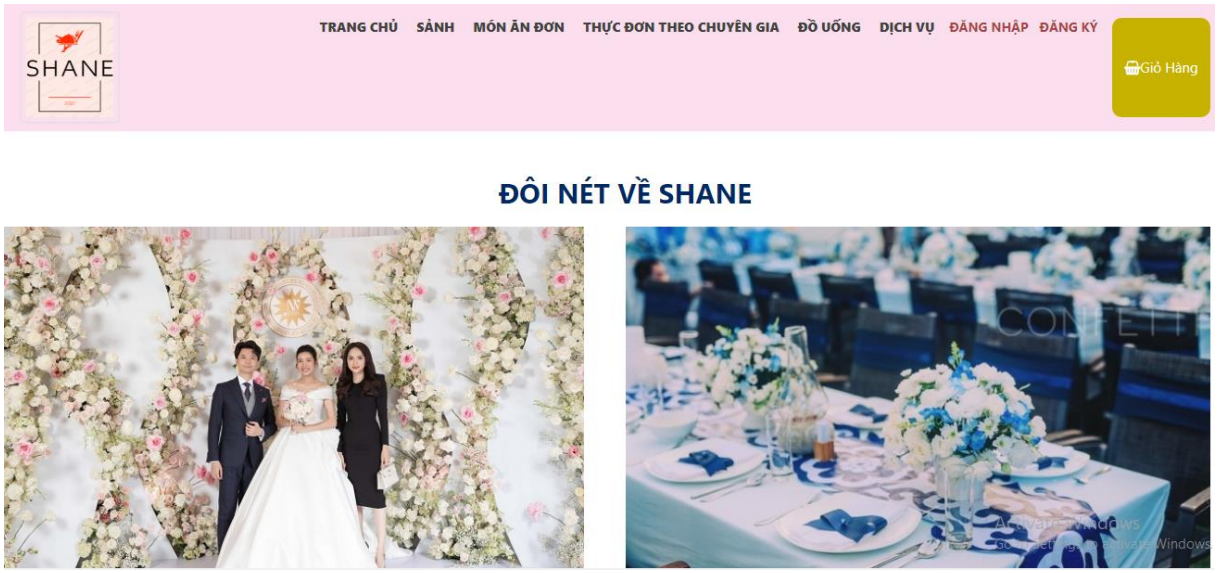
Response data:

```
{
  "id": 0,
```

```
  "party_name": "string",
  "lobbies": 0,
  "foods": [
    0
  ],
  "drinks": [
    0
  ],
  "services": [
    0
  ],
  "total_menu_food": 0,
  "total_menu_drink": 0,
  "total_service_price": 0,
  "table_quantity": 0,
  "rental_date": "2021-12-05T05:43:48.455Z",
  "created_date": "2021-12-05T05:43:48.455Z",
  "total_bill": 0
}
```

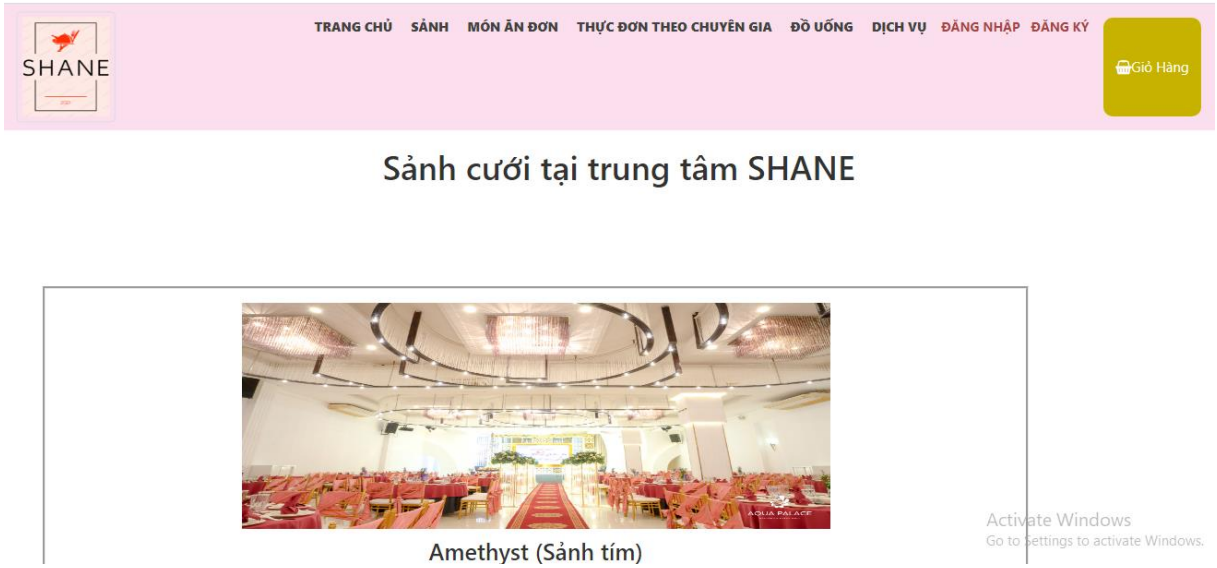
4.5.2. Giao diện phía client của hệ thống

4.5.2.1. Giao diện trang chủ



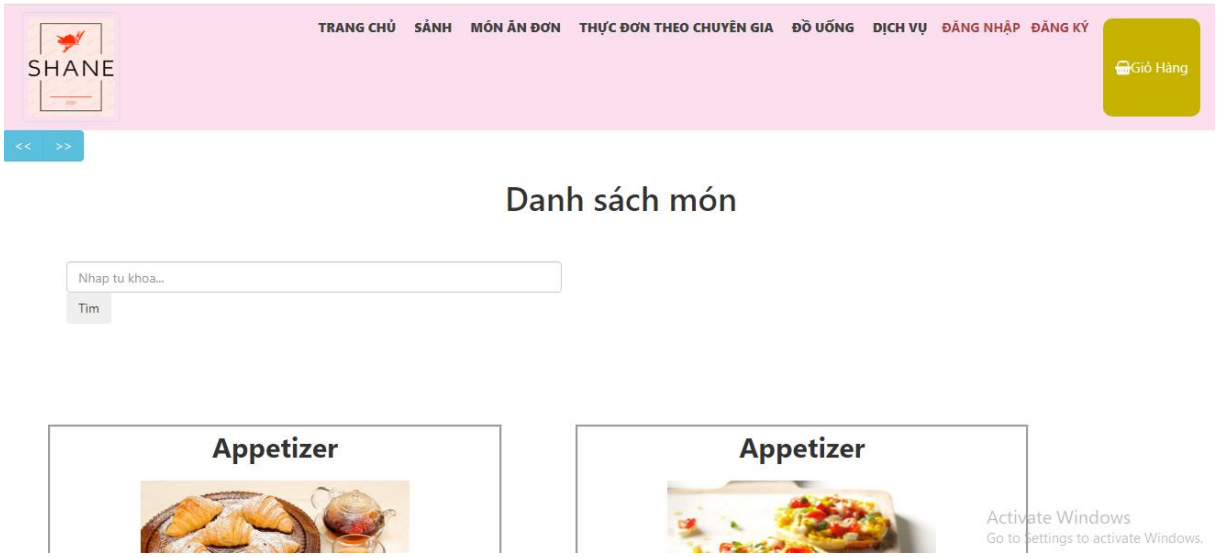
Hình 4.13: Giao diện trang chủ

4.5.2.2. Giao diện danh sách sảnh



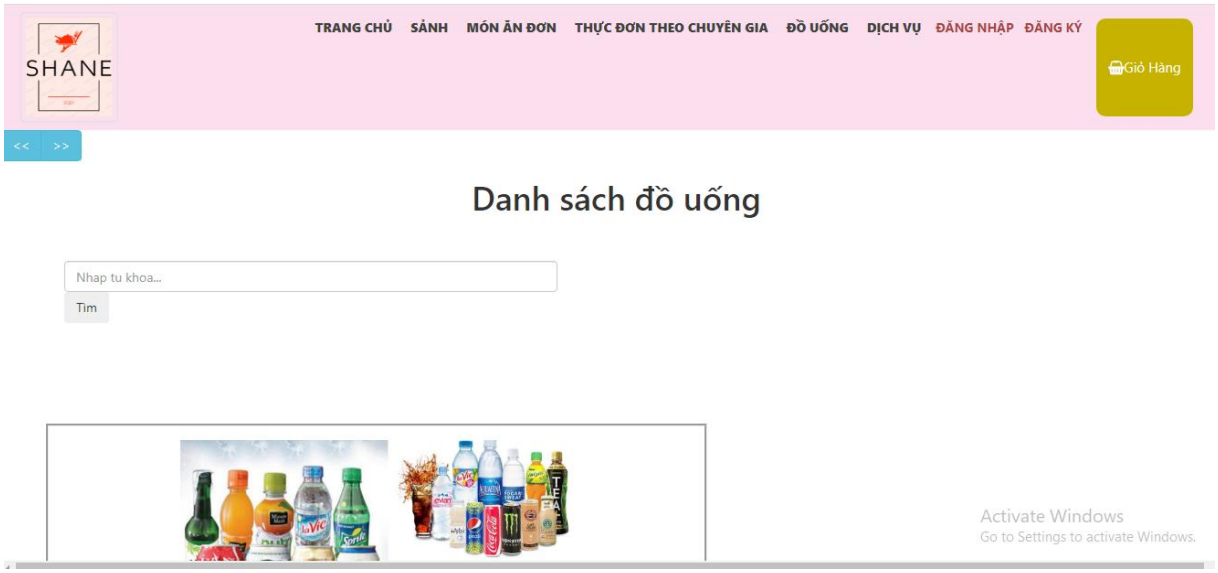
Hình 4.14: Giao diện danh sách sảnh

4.5.2.3. Giao diện danh sách món ăn



Hình 4.15: Giao diện danh sách món ăn

4.5.2.4. Giao diện danh sách đồ uống

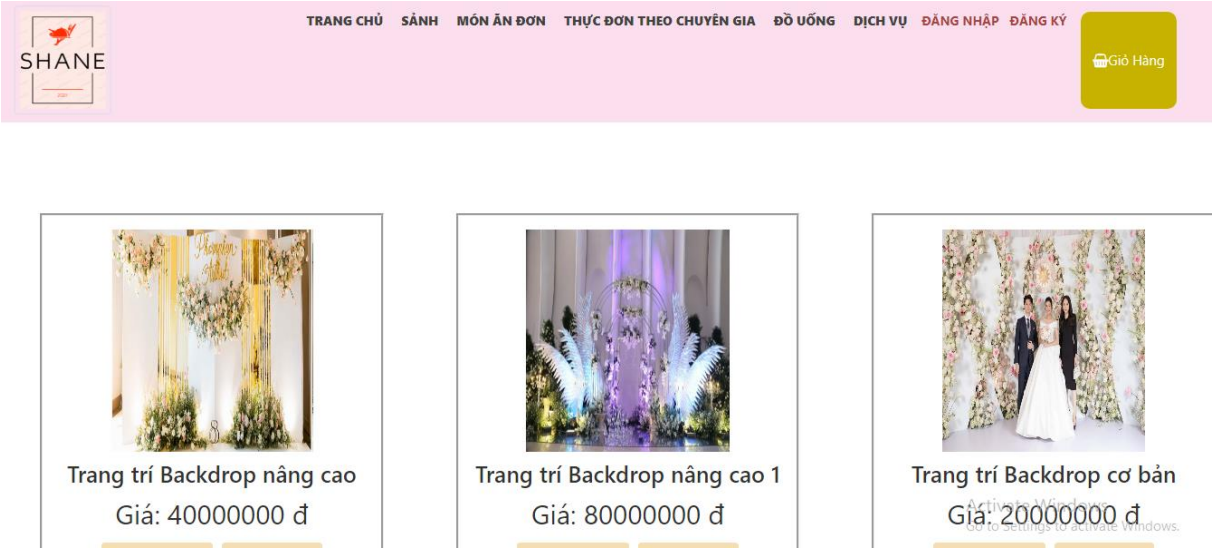


Hình 4.16: Giao diện danh sách đồ uống

4.5.2.5. Giao diện danh sách thực đơn

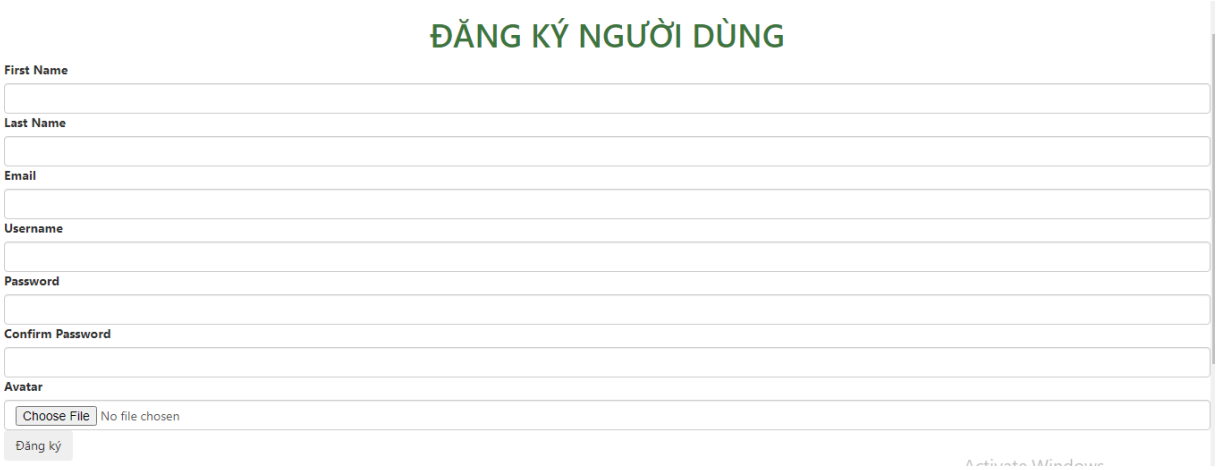
Hình 4.17: Giao diện danh sách thực đơn

4.5.2.6. Giao diện danh sách dịch vụ



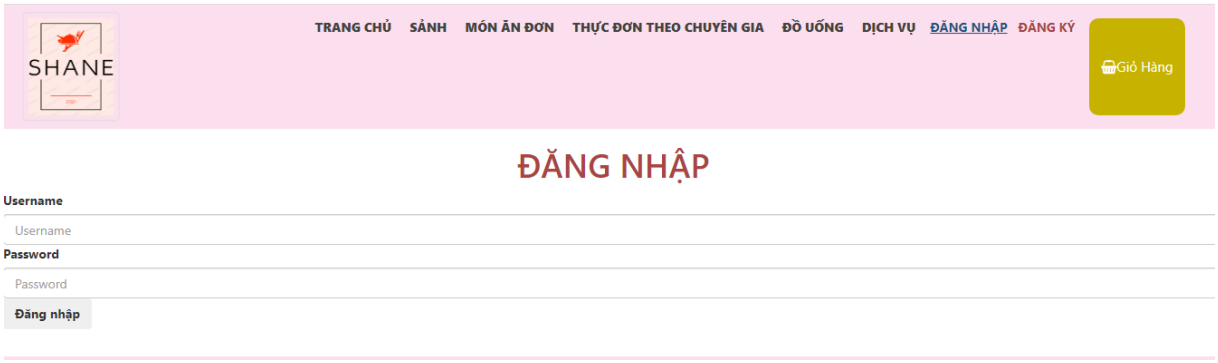
Hình 4.18: Giao diện danh sách dịch vụ

4.5.2.7. Giao diện đăng kí



Hình 4.19: Giao diện đăng kí

4.5.2.8. Giao diện đăng nhập



Hình 4.20: Giao diện đăng nhập

Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong chương này, em sẽ trình bày tổng quan về những điều hệ thống đã làm được, chưa làm được và chưa hoàn thiện. Đồng thời, em sẽ nêu lên hướng đi nâng cấp ứng dụng web này trong tương lai.

5.1. Kết luận

Những kết quả em thu được trong quá trình thực hiện đồ án ngành.

- Hiểu được cách viết một API từ Django Rest Framework và hiểu cơ bản cơ chế hoạt động của Django và Django Rest Framework.
- Hiểu được cách xây dựng các models để ánh xạ xuống cơ sở dữ liệu, phục vụ quá trình truy vấn dữ liệu và lưu trữ.

- Hiểu được cách xây dựng của lớp Serializer sử dụng để chuyển đổi dữ liệu từ những đối tượng thành các dạng dữ liệu như JSON.
- Xây dựng được hệ thống các API cơ bản cho ứng dụng quản lý nhà hàng tiệc cưới như đăng nhập, đăng kí, lấy danh sách món ăn, thực đơn, sảnh, dịch vụ, phản hồi khách hàng, …
- Xây dựng trang Admin từ Django thuận tiện cho việc quản lý dữ liệu.

Bên cạnh đó, hệ thống còn những điểm chưa hoàn thành được do vốn tích lũy kiến thức của em còn hạn chế.

- Chưa xây dựng được các API thanh toán trực tuyến qua các cổng thanh toán điện tử.
- Chưa xây dựng được chức năng thống kê mật độ tiệc cưới, thống kê doanh thu.
- Chưa xây dựng được trang phản hồi khách hàng.
- Chưa xây dựng được chức năng đặt tiệc.
- Hiệu suất truy vấn dữ liệu chưa tốt.

5.2. Hướng phát triển

Trong tương lai, em sẽ dành thời gian nghiên cứu thêm và hoàn thiện những chức năng chưa đạt được. Đồng thời, em sẽ mở rộng ứng dụng với các chức năng nâng cao như tư vấn online, mô phỏng tiệc cưới thông qua hệ thống thực tế ảo, tối ưu hiệu quả truy vấn dữ liệu.

TÀI LIỆU THAM KHẢO

[1] MISA CukCuk, "Cẩm nang chuyển đổi số ngành F&B cho các doanh nghiệp vừa và nhỏ", 4/10/2021. [Trực tuyến]. Địa chỉ: https://www.cukcuk.vn/11356/cam-nang-chuyen-doi-so-nganh-fb/#7-tam-ket. [Truy cập 22/11/2021].

[2] Nguyễn Thị Hương, "Báo cáo tình hình kinh tế - xã hội quý III và 9 tháng năm 2021", 29/9/2021. [Trực tuyến]. Địa chỉ: https://www.mpi.gov.vn/Pages/tinbai.aspx?idTin=51608&idcm=293. [Truy cập 22/11/2021].

[3] Thủy Nguyễn, "Django là gì và lý do vì sao nên sử dụng Django trong thiết kế web", 29/3/2021. [Trực tuyến]. Địa chỉ: Django là gì? Lý do vì sao nên sử dụng Django trong thiết kế web (bizfly.vn). [Truy cập 22/11/2021].

[4] Ths. Dương Hữu Thành, "Python Django", trong *Slide bài giảng môn học Các công nghệ lập trình hiện đại,* Tp. HCM: Nxb Khoa CNTT trường đại học Mở Tp. HCM, 2021, tr. 18-143.

[5] Cộng đồng Django, "URL dispatcher", 2019. [Trực tuyến]. Địa chỉ: https://docs.djangoproject.com/en/3.2/topics/http/urls/. [Truy cập 29/11/2021].

[6] Ths. Dương Hữu Thành, "Django Rest API", trong *Slide bài giảng môn học Các công nghệ lập trình hiện đại,* Tp. HCM: Nxb Khoa CNTT trường đại học Mở Tp. HCM, 2021, tr. 10-103.

[7] Ths. Dương Hữu Thành, "React JS", trong *Slide bài giảng môn học Các công nghệ lập trình hiện đại,* Tp. HCM: Nxb Khoa CNTT trường đại học Mở Tp. HCM, 2021, tr. 9-90.

[8] Cộng đồng Django Rest Framework, "Serialization", 2019. [Trực tuyến].

Địa chỉ: https://www.django-rest-framework.org/tutorial/1-serialization/. [Truy cập 29/11/2021].

[9] Cộng đồng React JS, "Handing Events", 2019. [Trực tuyến].

Địa chỉ: https://reactjs.org/docs/handling-events.html . [Truy cập 29/11/2021].

PHỤ LỤC