Northeastern University – Silicon Valley

CS 6620 Cloud Computing

Homework Set #3 [100 points]

INSTRUCTIONS: ***Please provide clear explanations in your own sentences, directly answering the question, demonstrating your understanding of the question and its solution, in depth, with sufficient detail. Submit your solutions [PDF preferred]. Include your full name. Do not email the solutions.***

PART I: Concepts and Theory, Algorithms [60 points]

All questions are in the context of a SaaS eCommerce platform. Here is a basic introduction:
https://www.toppr.com/guides/business-environment/emerging-trends-in-business/electronic-commerce/
Study the **CSCC Cloud Customer Architecture for e-Commerce.pdf** for understanding the use cases and architectural principles of eCom sites. Then, answer the below questions in your own words (1 – 2 small paras each; Diagrams as needed).

Please provide accurate, concise (maximum 10 sentence) answers. **All questions carry 10 points.**

Explain what the below concepts are and why and how are they used in an example eCommerce web application like Amazon.com etc. please provide example of how they are used for eCom. [40 points]

1. Model View Controller (MVC) design pattern
2. Cloud Computing reference Architecture for infrastructure
3. RESTful Web Services
4. Scalability

PART II: LAB (You don't have to use AWS Cloud for this) [60 points]

A. **Coding needed:** Build a basic Django Web App for a very basic ecommerce use case, keep it simple, following the below references. [45 points]
B. **No coding needed for this part:** Now inspect this fully functional eCommerce Web App using Django (from the reference below). How does this design compare with what you learned in Part I above? Can you use Django to extend your basic app from A to build this? (No need to implement or code. ONLY answer this part B using diagrams and your explanation.) [15 points]

References

Basic Django for Part II. (A)
https://docs.djangoproject.com/en/3.2/intro/tutorial01/
https://www.youtube.com/watch?v=UmljXZIypDc
https://realpython.com/get-started-with-django-1/

For Part II (B) **Please see step by step guidance below.**

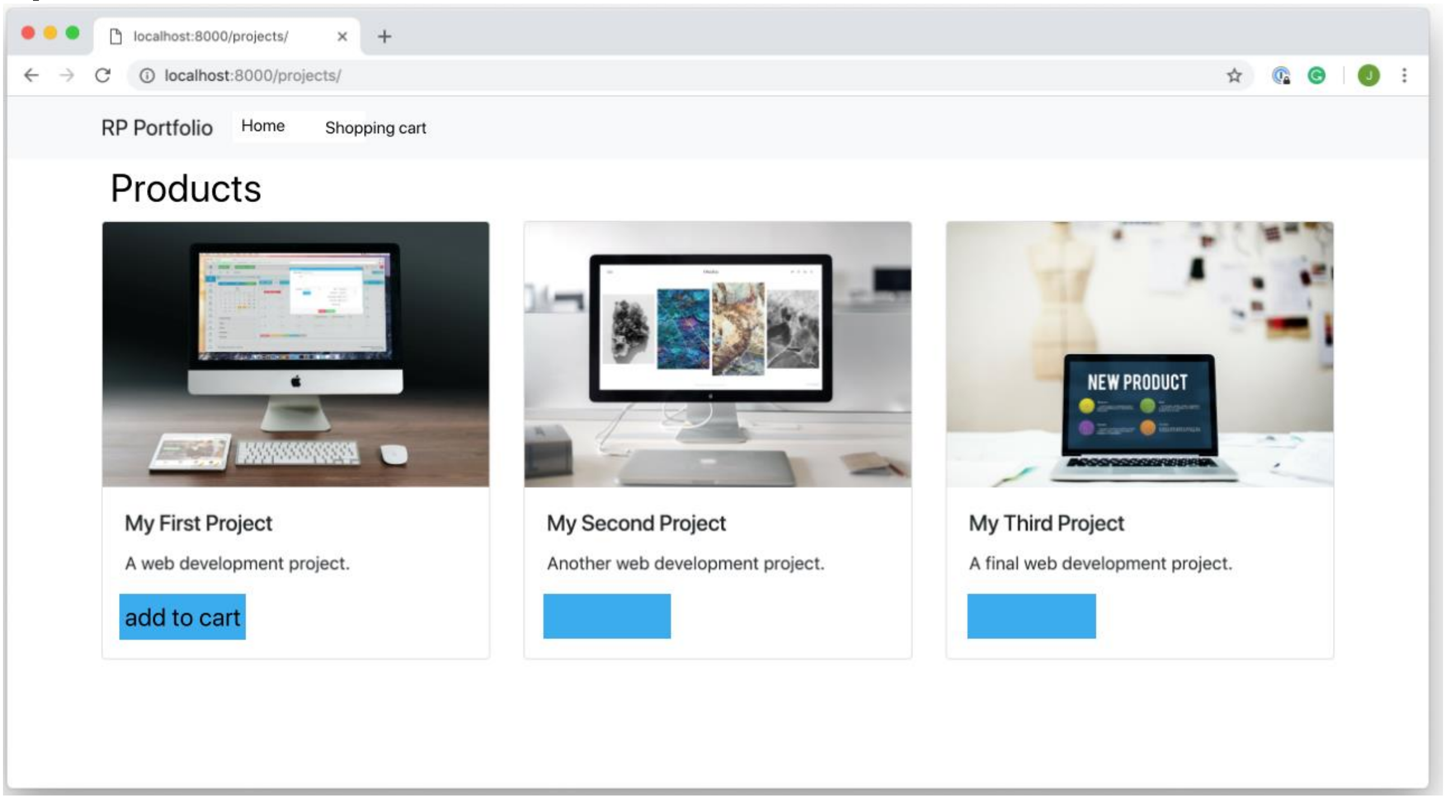
Full eCom site
Step 1 of 5: Setup and Configuration
https://medium.com/analytics-vidhya/how-to-create-simple-e-commerce-website-with-django-step-1-of-5-42c6cca414c2
Step 2 of 5: Make an Order
https://medium.com/analytics-vidhya/how-to-create-fully-functional-e-commerce-website-with-django-c95b46973b0
Step 3 of 5: Make an Order Summary

https://medium.com/analytics-vidhya/how-to-create-fully-functional-e-commerce-website-with-django-997ffaa0f040
Step 4 of 5: Create a Checkout Form
https://medium.com/analytics-vidhya/how-to-create-fully-functional-e-commerce-website-with-django-7205d250e76f
Step 5 of 5: Payment Handling with Stripe
https://medium.com/analytics-vidhya/how-to-create-a-fully-functional-e-commerce-website-with-django-d4b998bac01

Basic view of part A:

1. home view: (example as below)
user: display username on the up right corner
products: show the product and description, add a button to add to cart (change the below projects data to products information)
2. shopping cart view:
Show a table of added products, user can do CRUD of the product.

Stretched goal:
1. product details view: show some hard-coded comments



Products

My First Project
A web development project.

My Second Project
Another web development project.

My Third Project
A final web development project.

ANS (Qichen An)

Part 1

1. Model View Controller (MVC) design pattern

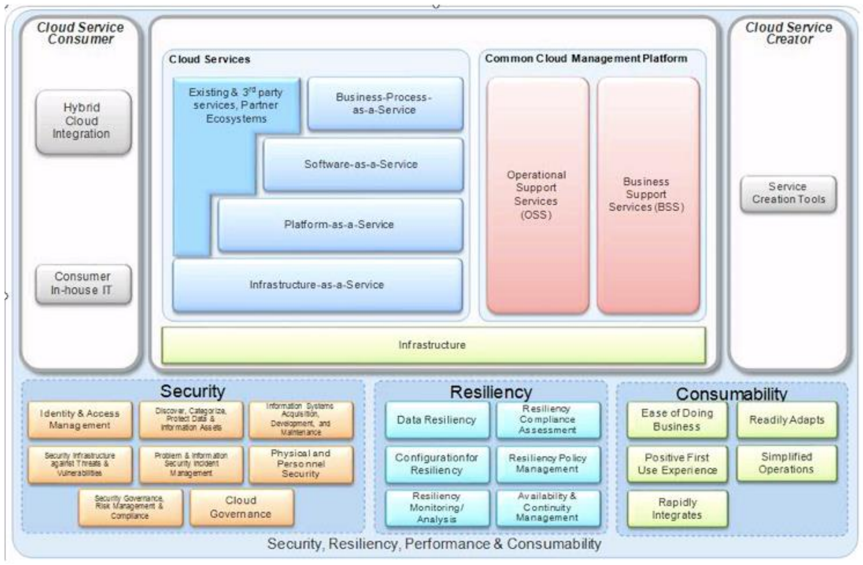
Model view controller (MVC) is a software design pattern used for developing user interfaces that divide the related program logic into three interconnected elements. The goal of MVC definition is to streamline communication among developers.

Model is the central component of the pattern. The controller is an intermediary between the Model and the View. It accepts user input from the View, and converts it to commands for the Model or View.

In the MVC eCommerce web application development, the controller receives all requests for the application and then instructs the model to prepare any information required by the view. The view uses that data prepared by the controller to bring the final output. For an example eCommerce web application (Amazon), MVC can support faster development process, support for asynchronous technique, and has ability to provide multiple Views. In summary, with the help of the MVC model, Amazon web application can be easily developed with lesser expenditure and within less time.

2. Cloud Computing reference Architecture for infrastructure

CCRA is an IBM-defined reference architecture for the cloud computing domain. It is an evolving architecture that is based on real-world input from many cloud implementations. The CCRA defines the basic building blocks—architectural elements and their relationships, which make up the cloud.


Cloud Service Consumer
Hybrid Cloud Integration
Consumer In-house IT
Cloud Services
Existing & 3rd party services, Partner Ecosystems
Business-Process-as-a-Service
Software-as-a-Service
Platform-as-a-Service
Infrastructure-as-a-Service
Common Cloud Management Platform
Operational Support Services (OSS)
Business Support Services (BSS)
Infrastructure
Cloud Service Creator
Service Creation Tools
Security
Identity & Access Management
Discover, Categorize, Protect Data & Information Assets
Information Systems Acquisition, Development, and Maintenance
Security Infrastructure against Threats & Vulnerabilities
Problem & Information Security Incident Management
Physical and Personnel Security
Security Governance, Risk Management & Compliance
Cloud Governance
Resiliency
Data Resiliency
Resiliency Compliance Assessment
Configuration for Resiliency
Resiliency Policy Management
Resiliency Monitoring/ Analysis
Availability & Continuity Management
Consumability
Ease of Doing Business
Readily Adapts
Positive First Use Experience
Simplified Operations
Rapidly Integrates
Security, Resiliency, Performance & Consumability

For an example eCommerce web application (Amazon), CCRA is used as a blueprint for architecting cloud implementations, driven by functional and non-functional requirements of the respective cloud implementation of eCommerce web application (Amazon). So, it improves the time to capability and reduces the overall IT costs associated with private, public and hybrid cloud models for the eCommerce web application.

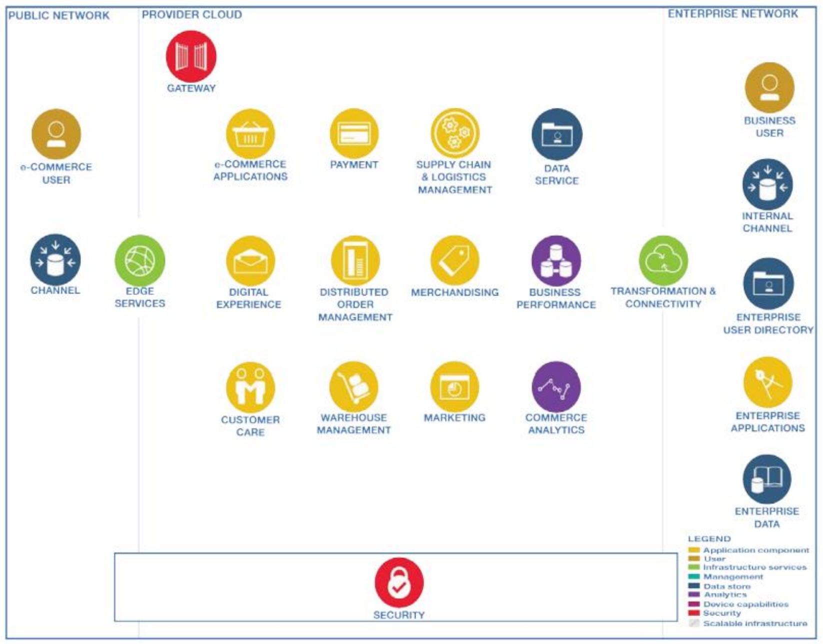


Figure 1: Elements of e-Commerce Solution

3. RESTful Web Services

RESTful APIs enable automated management of storage clouds that offer flexibility to meet the needs beyond standard cloud tools. With these APIs, it is possible to build custom cloud storage management software stacks. With the application of RESTful APIs to a web service, desirable properties could be induced, such as performance, scalability, and modifiability, to enable services to work best on the Web.

For an example eCommerce web application (Amazon), a centralized GUI along with RESTful applications that are provided by the underlying storage infrastructure are used to provide monitoring and metering capabilities. The design of object storage also simplifies how users access data of eCommerce web application through the use of RESTful APIs, such as GET, PUT, and DELETE. Openstack Swift and the Amazon Simple Storage Service (S3) are specific examples of object storage. Also, a RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery.

4. Scalability

Scalability is the ability to non-disruptively add capacity and remove it as needed in a global namespace is a key function for storage clouds. A global namespace aggregates disparate storage infrastructure, potentially across geographical boundaries, to provide a consolidated file or object view that simplifies administration

For an example eCommerce web application (Amazon), messaging systems can provide scalability. When using messaging tools, direct dependencies between services are removed, which improves reliability and scalability. The asynchronous relationship between saving and retrieving messages also provides scalability and reliability benefits. Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability.

Part2

A. Build a basic Django Web App for a very basic ecommerce use case, keep it simple, following the below references. [45 points]

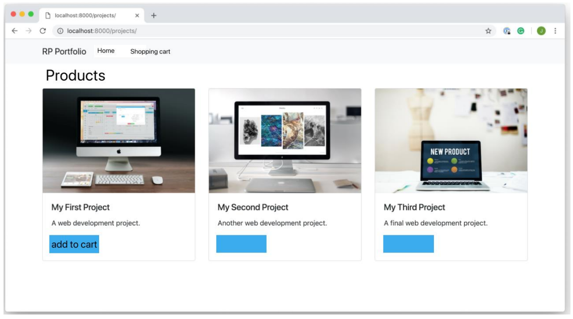
1. home view: (example as below)

user: display username on the up right corner

products: show the product and description, add a button to add to cart (change the below projects data to products information)

2. shopping cart view:

Show a table of added products, user can do CRUD of the product.


localhost:8000/projects/

RP Portfolio Home Shopping cart

Products

My First Project
A web development project.
add to cart

My Second Project
Another web development project.

My Third Project
A final web development project.

Before coding, it's a good idea to create a virtual environment to manage dependencies.

Shell

```
$ python3 -m venv venv
```

Shell

```
$ python3 -m venv venv
```

Then activate the virtual environment.

Windows Console

```
C:\> venv\Scripts\activate.bat
```

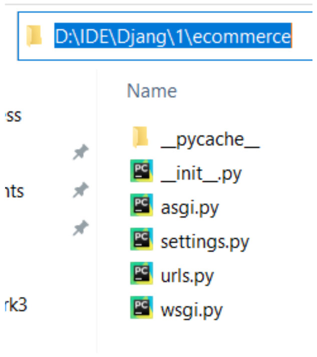
Install Django

Shell

```
(venv) $ pip install Django
```

Then, create a Django Project with command below:

$ django-admin startproject ecommerce



D:\IDE\Djang\1\ecommerce

Name

```
__pycache__
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
```

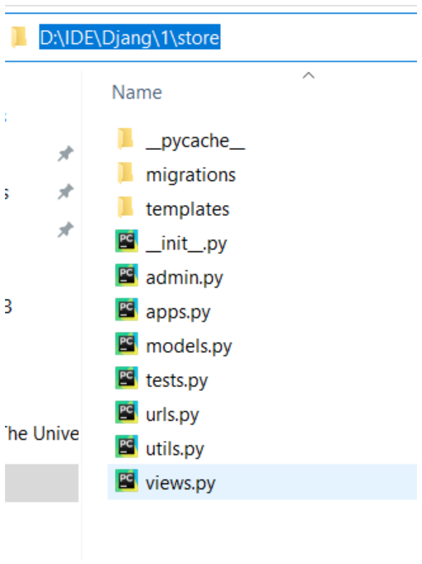
migrate admin database with command below:

$ python manage.py migrate

$ python manage.py makemigrations

Create a Django Application with command below:

$ python manage.py startapp



D:\IDE\Djang\1\store

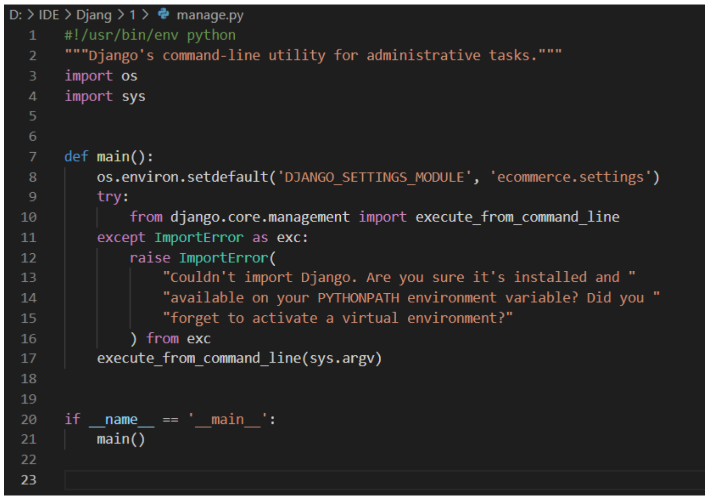
Name

```
__pycache__
migrations
templates
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
utils.py
views.py
```

Running the development server with command below:

$ python manage.py runserver

manage.py



D: > IDE > Djang > 1 > manage.py

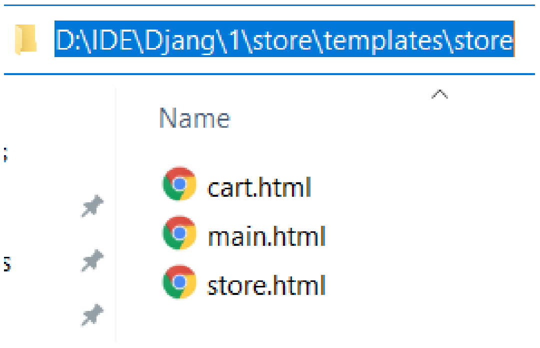
```python
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ecommerce.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()

```

Templates

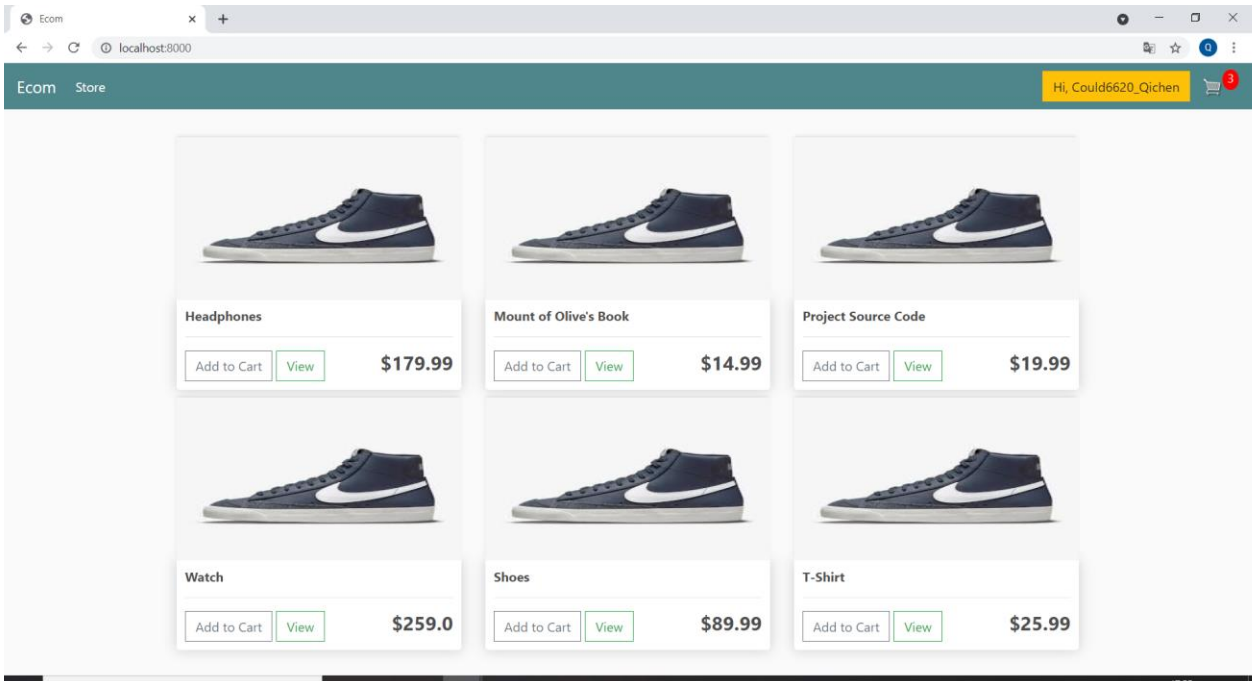


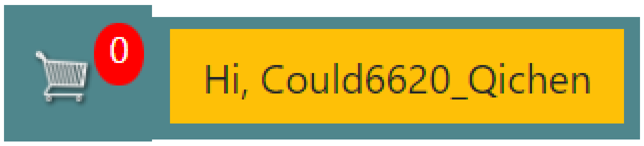
D:\IDE\Djang\1\store\templates\store

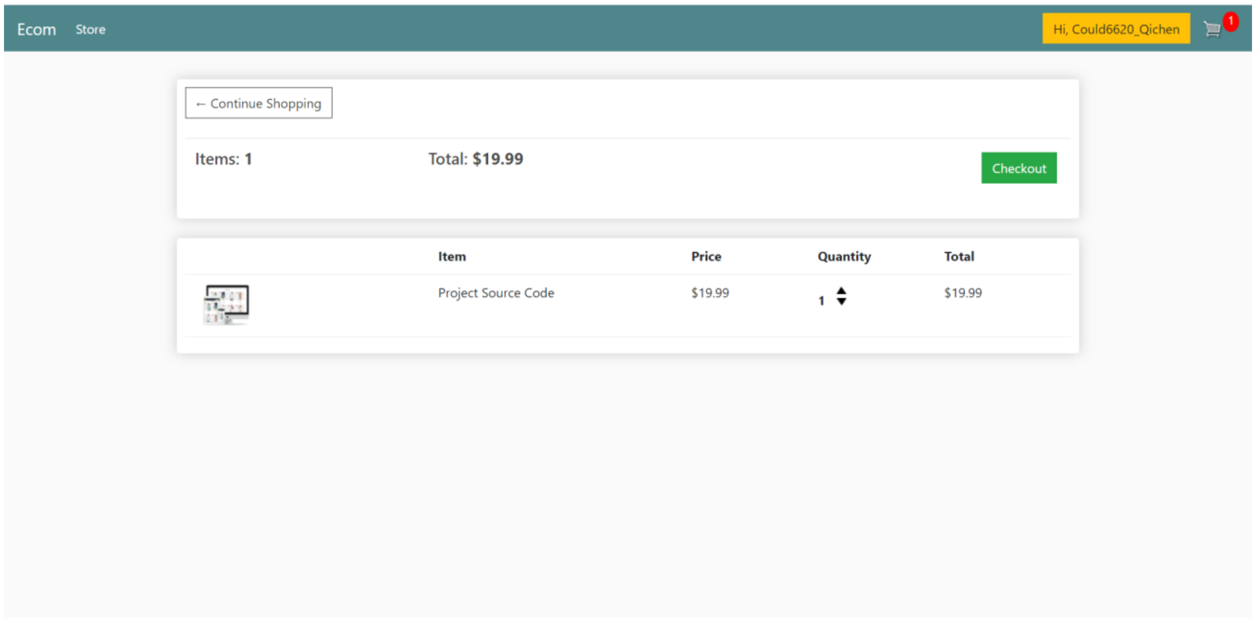
Name

cart.html
main.html
store.html

Final result

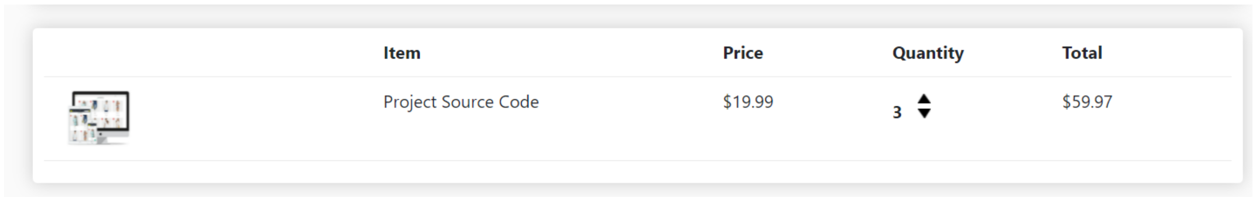






Items: 1 Total: $19.99

	Item	Price	Quantity	Total
	Project Source Code	$19.99	1	$19.99



	Item	Price	Quantity	Total
	Project Source Code	$19.99	3	$59.97

B. Now inspect this fully functional eCommerce Web App using Django (from the reference below). How does this design compare with what you learned in Part I above? Can you use Django to extend your basic app from A to build this? (No need to implement or code. ONLY answer this part B using diagrams and your explanation.) [15 points]

Compared to the design given with my own, there are some difference. For the difference, they could increase project integrity may improve the efficiency and aesthetics to a certain extent.

For the first difference, the design given suggests using Anaconda, whereas I used virtualenv. Thinking carefully about the reason of the difference, I think it might be more suitable for long term developers for web applications and it could improve the efficiency because they don't need to configure the environment every time they designing a web.

Secondly, the designed given support the logging in function. Due to the time limit of homework just for one week and I am still a newcomer to web development, I don't have the energy to add more features. With the logging in function, the integrity of the whole design could be more complete.

Another thing is the function of checkout. Similar to the second one, I think if adding this function could improve the integrity of the whole web application development.

Furthermore, the author supports a function – payment handling with Stripe. I think it's one of the most important function for eCommerce web application. Because the goal of us is to earn money and during this section, security should be protected as a favor for ease.

Lastly, I want to talk about the large amount use of databases and view for the design. As it's the crucial element of web development, I think it would be better to practice much more about databases and view.