NIIT

Final Project (Ecommerce Website) Documentation – Mr. David Adediji

# Project Details

|  |  |
| --- | --- |
| Project Name | Coding Course Ordering Ecommerce Website |
| Student Name | Mr. David Adediji |
| Name of Instructor | Mr. Fredrick Jones |
| Presentation Date | 2nd of June, 2023 |
| Program Name | MMS |
|  |  |

# Acknowledgements

I want to deeply express gratitude to my Program Instructors, Mr. Belba Ngoy, Mr. Efe Otega & Ms. Esther Iyege, Mr. Fredrick Jones, for taking the class diligently through my courses. Their lessons were very concise and easy to understand. I also appreciate the opportunity given to defend a project pertaining to what I had been thought and had understood. More had been possible with the assistance of my helpful colleagues who helped make the lessons interactive, fun and had even contributed indirectly to the development of my Ecommerce Project.

# Project Overview

* The name of the ecommerce website is Code Avenue.
* This web project using jQuery and Django was created to enable users to buy programming courses.
* This project idea is inspired by Code with Mosh’s approach to ecommerce for his programming courses.
* The web application was created using HTML, CSS, jQuery programming language and Django REST API framework, stripe alongside other resources.

# Project Features

* Ecommerce logic to allow users buy courses
* Ecommerce logic to allow users subscribe for courses
* Payment using Stripe API
* Blog to convey information about programming
* Profile page to manage user information
* Properly validated logic and signup page
* Informative learning paths to help direct people to get the programming education they want
* Instructor accessible via email messaging on contact page
* Beautiful lollipop design
* User authentication using Django
* Project hosted on Heroku and GitHub Pages

# Project Process

* I designed a wireframe using Figma
* I downloaded font awesome
* Design of flowcharts for program logic
* Creation of Website markup with HTML
* Styling of website with CSS
* Programming of ecommerce website Logic with jQuery
* Repeating process to build peripheral blog
* Built backend with Django
* Payment Processing with Stripe
* Testing
* Live Hosting

# Technical Lessons applied in project

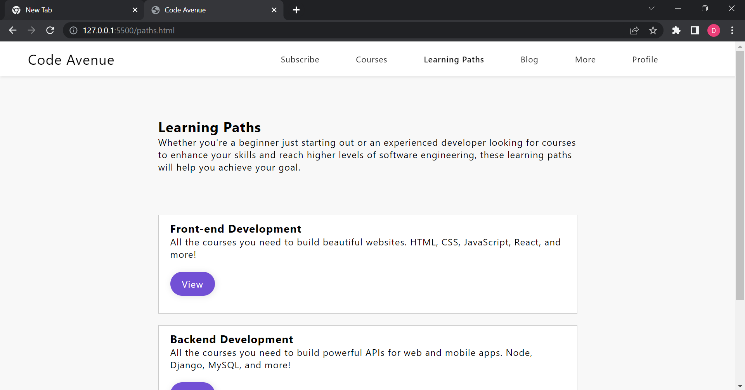
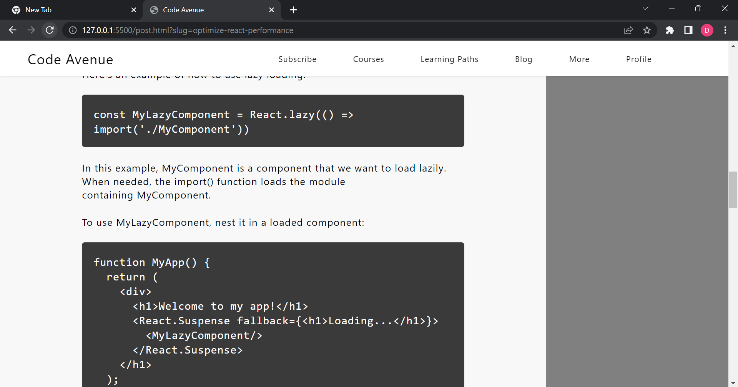
* Ajax requests
* Load function
* jQuery selectors
* Local Storage
* jQuery functions
* String Literals
* jQuery Event Listeners
* Django ORM
* Stripe API
* Django rest framework
* Django Authentication
* Atypical CSS selectors

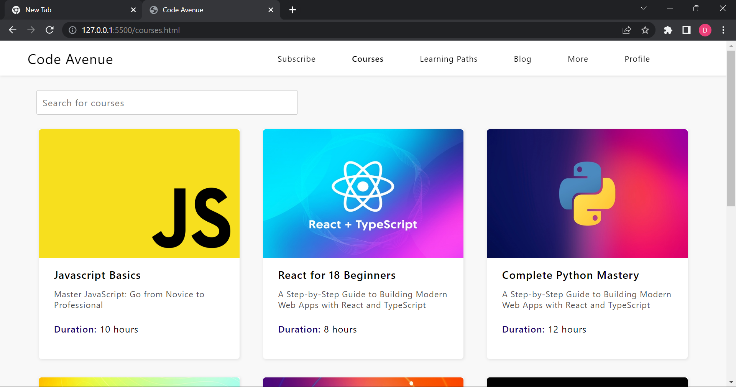
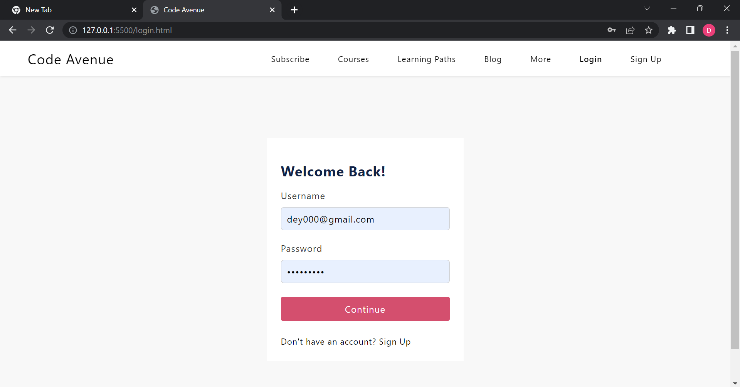
# Project Images

# 

# 

# 

# Source Code

## Django Models

core/models.py:

from django.contrib.auth.models import AbstractUser

from django.db import models

# Create your models here.

class User(AbstractUser):

email = models.EmailField(unique=True)

class ContactMessage(models.Model):

email = models.EmailField()

message = models.TextField()

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

def \_\_str\_\_(self):

return self.email

blog/models.py:

from django.db import models

from django.conf import settings

from ckeditor.fields import RichTextField

class Category(models.Model):

name = models.CharField(max\_length=100)

slug = models.SlugField(max\_length=100, unique=True)

def \_\_str\_\_(self):

return self.name

class Post(models.Model):

title = models.CharField(max\_length=255)

slug = models.SlugField(max\_length=255, unique=True)

author = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

content = RichTextField()

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

updated\_on = models.DateTimeField(auto\_now=True)

category = models.ForeignKey(Category, on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return self.title

class Comment(models.Model):

post = models.ForeignKey(Post, on\_delete=models.CASCADE, related\_name='comments')

user = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

email = models.EmailField()

content = models.TextField()

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

approved = models.BooleanField(default=False)

class Meta:

unique\_together = (('post', 'user'),)

def \_\_str\_\_(self):

return self.content

class Subscriber(models.Model):

email = models.EmailField(unique=True)

subscribed\_on = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.email

course\_store/models.py:  
from django.db import models

from django.conf import settings

from django.utils import timezone

# Course model class to represent a course

class Course(models.Model):

LEVEL\_CHOICES = (

('beginner', 'Beginner'),

('intermediate', 'Intermediate'),

('advanced', 'Advanced'),

)

level = models.CharField(max\_length=20, choices=LEVEL\_CHOICES)

endpoints = models.TextField()

features = models.TextField()

name = models.CharField(max\_length=35)

description = models.TextField()

summary = models.CharField(max\_length=80)

image = models.ImageField(upload\_to='course\_images/', default="./course\_images/js.png")

time\_length = models.PositiveSmallIntegerField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

discount\_percentage = models.PositiveSmallIntegerField()

tagline = models.CharField(max\_length=55)

available = models.BooleanField(default=True)

downloadable\_file = models.FileField(upload\_to='course\_files/', blank=True)

shortform = models.CharField(max\_length=35)

def \_\_str\_\_(self):

return self.name

class BillingAddress(models.Model):

user = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

street\_address = models.CharField(max\_length=255)

city = models.CharField(max\_length=255)

country = models.CharField(max\_length=255)

postal\_code = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return f'Billing address for {self.user.username}'

# Order model class to represent an order placed by a user for a course

class Order(models.Model):

user = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

price = models.DecimalField(max\_digits=10, decimal\_places=2)

date\_ordered = models.DateTimeField(auto\_now\_add=True)

billing\_address = models.ForeignKey(BillingAddress, on\_delete=models.PROTECT)

downloaded = models.BooleanField(default=False)

def \_\_str\_\_(self):

return f'{self.user.username} ordered {self.course.name} on {self.date\_ordered}'

# Payment model class to represent a payment made by a user for an order

class Payment(models.Model):

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

amount = models.DecimalField(max\_digits=10, decimal\_places=2)

card\_number = models.BigIntegerField()

card\_expiry\_date = models.DateField()

cvc = models.IntegerField()

intent\_id = models.CharField(max\_length=255)

card\_holder = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return f'{self.order.user.username} paid {self.amount} for {self.order.course.name}'

# Review model class to represent a review made by a user for a course

class Review(models.Model):

user = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

content = models.TextField()

rating = models.IntegerField()

def \_\_str\_\_(self):

return f'{self.user.username} reviewed {self.course.name}'

from django.db.models.signals import post\_save

from django.dispatch import receiver

from datetime import date

# Order model class to represent an order placed by a user for a course

class SubscriptionOrder(models.Model):

DURATION\_CHOICES = (

('monthly', 'Monthly'),

('annual', 'Annual'),

)

user = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

duration = models.CharField(max\_length=10, choices=DURATION\_CHOICES)

date\_ordered = models.DateTimeField(auto\_now\_add=True)

billing\_address = models.ForeignKey(BillingAddress, on\_delete=models.PROTECT)

def \_\_str\_\_(self):

return f'{self.user.username} ordered for all access on {self.date\_ordered}'

class SubscriptionPayment(models.Model):

order = models.OneToOneField(SubscriptionOrder, on\_delete=models.CASCADE)

amount = models.DecimalField(max\_digits=10, decimal\_places=2)

card\_number = models.BigIntegerField()

card\_expiry\_date = models.DateField()

cvc = models.IntegerField()

intent\_id = models.CharField(max\_length=255)

card\_holder = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return f'{self.order.user.username} paid {self.amount} for all access'

class AllAccessSubscriber(models.Model):

user = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

subscription\_start\_date = models.DateField(auto\_now\_add=True)

subscription\_end\_date = models.DateField()

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

subscription\_payment = models.OneToOneField(SubscriptionPayment, on\_delete=models.CASCADE)

# Other fields and methods specific to the AllAccessSubscriber model

@receiver(post\_save, sender=AllAccessSubscriber)

def update\_subscription\_status(sender, instance, \*\*kwargs):

if instance.subscription\_end\_date < date.today():

instance.is\_active = False

instance.save()