E-commerce Website -Project Report

Submitted To:

Prof. John Agar
(Adjunct Faculty)

Submitted By:

1) Sarath Upadrista su72648n@pace.edu
2) Pavan Kumar Varkala - Pv24664n@pace.edu
3) Gowthami Singamsetty – gs63100n@pace.edu
4) Phanindhr Thota – PT23858N@pace.edu
5) Bhavya Panguluri – Bp44114n@pace.edu

Contents

1. ABSTRACT

E-commerce, short for electronic commerce, refers to the buying and selling of goods and services over the internet. E-commerce websites serve as digital platforms that facilitate online transactions between businesses and consumers. These platforms have become integral to modern commerce, offering a convenient and efficient way for individuals and organizations to engage in commercial activities.

Key features of e-commerce websites include online product catalogs, secure payment gateways, and streamlined checkout processes. Users can browse through a diverse range of products, compare prices, and make purchases without the need to physically visit a brick-and-mortar

store. E-commerce platforms often leverage technologies such as secure sockets layer (SSL) encryption to ensure the security of financial transactions and protect sensitive information.

The success of an e-commerce website is often influenced by factors such as user experience, website design, product assortment, and the efficiency of order fulfillment and delivery processes. Many e-commerce sites also incorporate features like customer reviews, ratings, and personalized recommendations to enhance the overall shopping experience.

E-commerce has expanded beyond traditional retail, encompassing various models such as business-to-consumer (B2C), business-to-business (B2B), and consumer-to-consumer (C2C) transactions. Mobile commerce (m-commerce) has further contributed to the growth of e-commerce, enabling users to make purchases using smartphones and other mobile devices.

As the digital economy continues to evolve, e-commerce websites play a pivotal role in shaping the future of commerce. They provide businesses with global reach, enable consumers to access a vast array of products, and contribute to the ongoing transformation of the retail landscape. As technology advances, e-commerce platforms are likely to incorporate innovations such as artificial intelligence, augmented reality, and virtual reality to create more immersive and personalized shopping experiences.

2. INTRODUCTION

The E-commerce project is a sophisticated online shopping platform developed using cutting-edge technologies, including Python, Django, and SQLite. This project aims to provide a seamless and user-friendly e-commerce experience, leveraging the power of these technologies to create a robust and efficient online marketplace.

3. Project Overview:

The project is structured around key modules that collectively cover the entire user journey, from product discovery to order tracking. These modules include the Home Page, Cart Management, Checkout Page, and Order Tracker, each playing a crucial role in enhancing the user experience.

3.1. Home Page:

The Home Page serves as the gateway to the online store, offering a well-organized and categorized list of products. This ensures a user-friendly navigation experience, allowing users to explore various product categories effortlessly.

The module also allows users to directly add products to their shopping cart from the home page, streamlining the process of product selection.

3.2. Cart Management:

The Cart Management module provides users with control over their shopping cart. Users can view, update quantities, and remove items, offering flexibility and convenience in managing their selected products.

Real-time calculation of the total cost ensures transparency, and the seamless integration of this module into the shopping process enhances the overall user experience.

3.3. Checkout Page:

The Checkout Page is a critical component, collecting vital information such as the user's delivery address. A comprehensive order summary is presented, including the total amount to be paid.

Users seamlessly transition from product selection to the payment stage, with a user-friendly interface contributing to a hassle-free checkout process.

3.4. Order Tracker:

The Order Tracker module enhances post-purchase visibility. Users can search for orders based on the Order ID and Email ID, providing real-time updates on the status and progress of their purchases.

This module contributes to customer satisfaction by offering transparency and building trust in the order fulfillment process.

4. Technologies Used:

- **Frontend:**
  - HTML
  - CSS
  - JavaScript
- **Backend:**
  - Python
  - Django
- **Database:**
  - SQLite3

4.1 Python:

Python, a versatile and widely-used programming language, is employed for backend development. Its readability and extensive ecosystem make it an ideal choice for handling server-side logic and database interactions.

4.2 Django:

Django, a high-level web framework for Python, accelerates development by providing pre-built components and adhering to the Model-View-Controller (MVC) architectural pattern. It ensures the creation of a secure, scalable, and maintainable web application.

4.3 SQLite:

SQLite serves as the database management system, offering a lightweight and embedded solution. It efficiently manages data related to products, user information, and order details, contributing to the overall performance of the application.

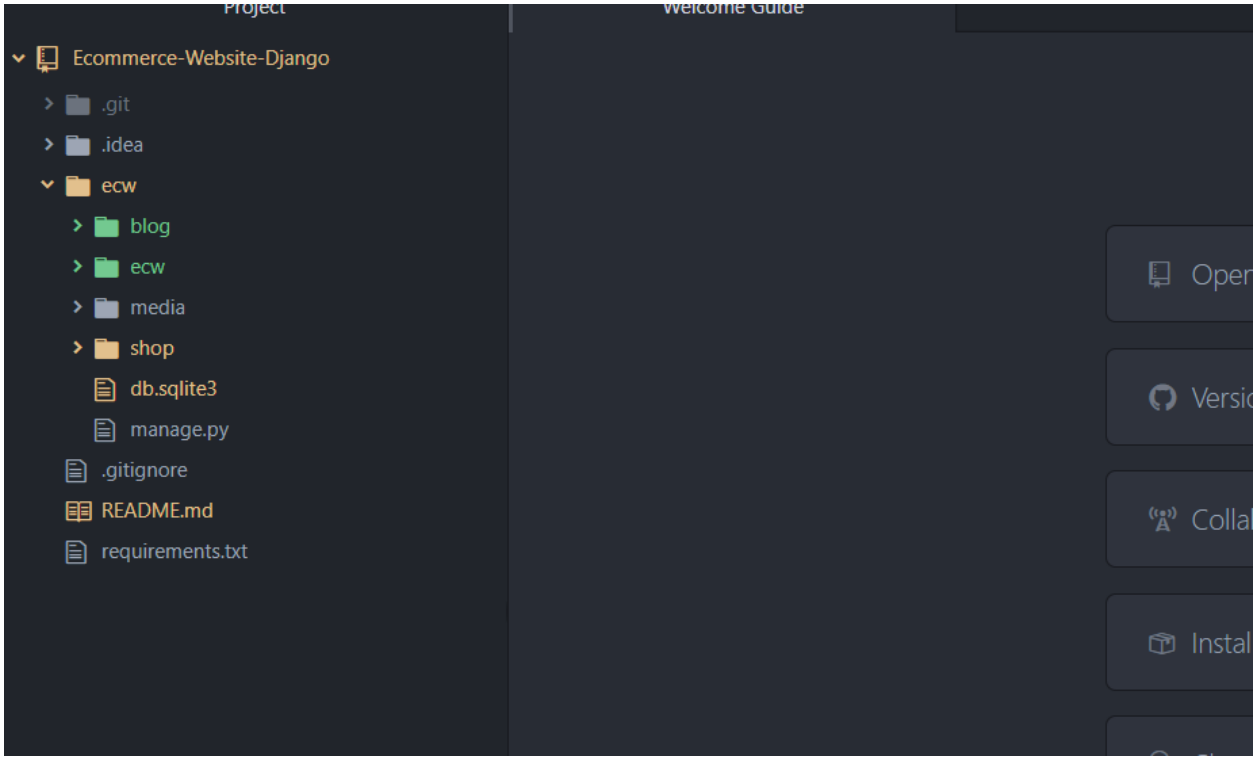
4.4 HTML, CSS:

HTML and CSS are fundamental for creating a visually appealing and responsive user interface. HTML structures the content of web pages, while CSS ensures a consistent and visually pleasing layout, enhancing the overall user experience.

4.5 Project Structure:

This project is grouped into 3 modules.

1) ecw – this is the entry point for the project. It contains index.html, settings.py, urls.py
    - This module will combine the shop and blog modules into one project.
2) Shop
    - This is the main project. It contains all the pages and backend logic to perform, interactions with the database etc
3) Blog
    - This module contains the blogging information ie.., it will give information about the products in the e-commerce web site. It will retrieve the data from the database and renders in the UI.
4) Media – This folder contains the product images that are displayed in the home page.



- /static

  - /css

  - style.css

  - /js

  - script.js

Ecommerce-Website-Django

- ecw

  - ecw

    - /templates (contains html pages)

      - index.html

    - urls.py

    - views.py

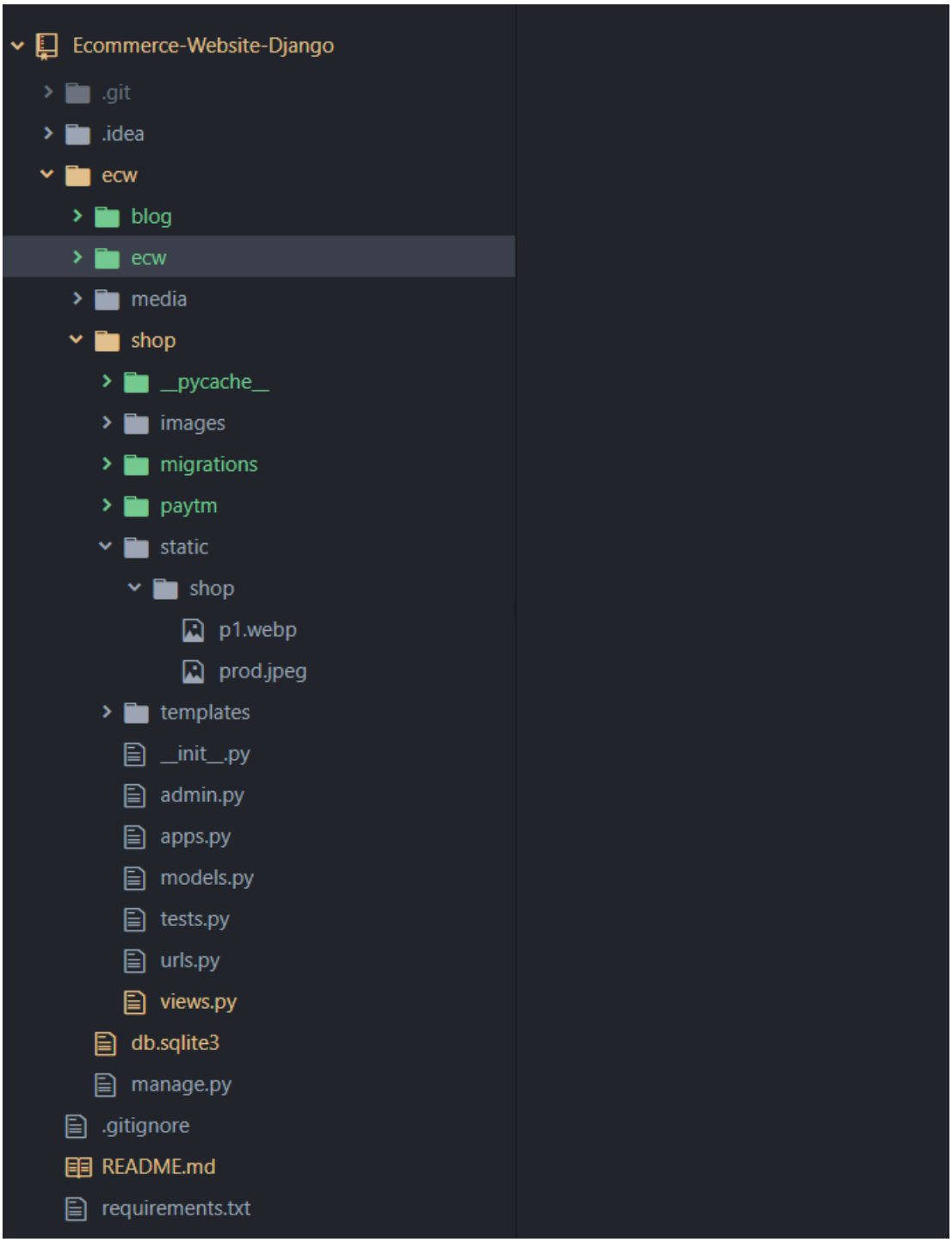
    - models.py

    - settings.py

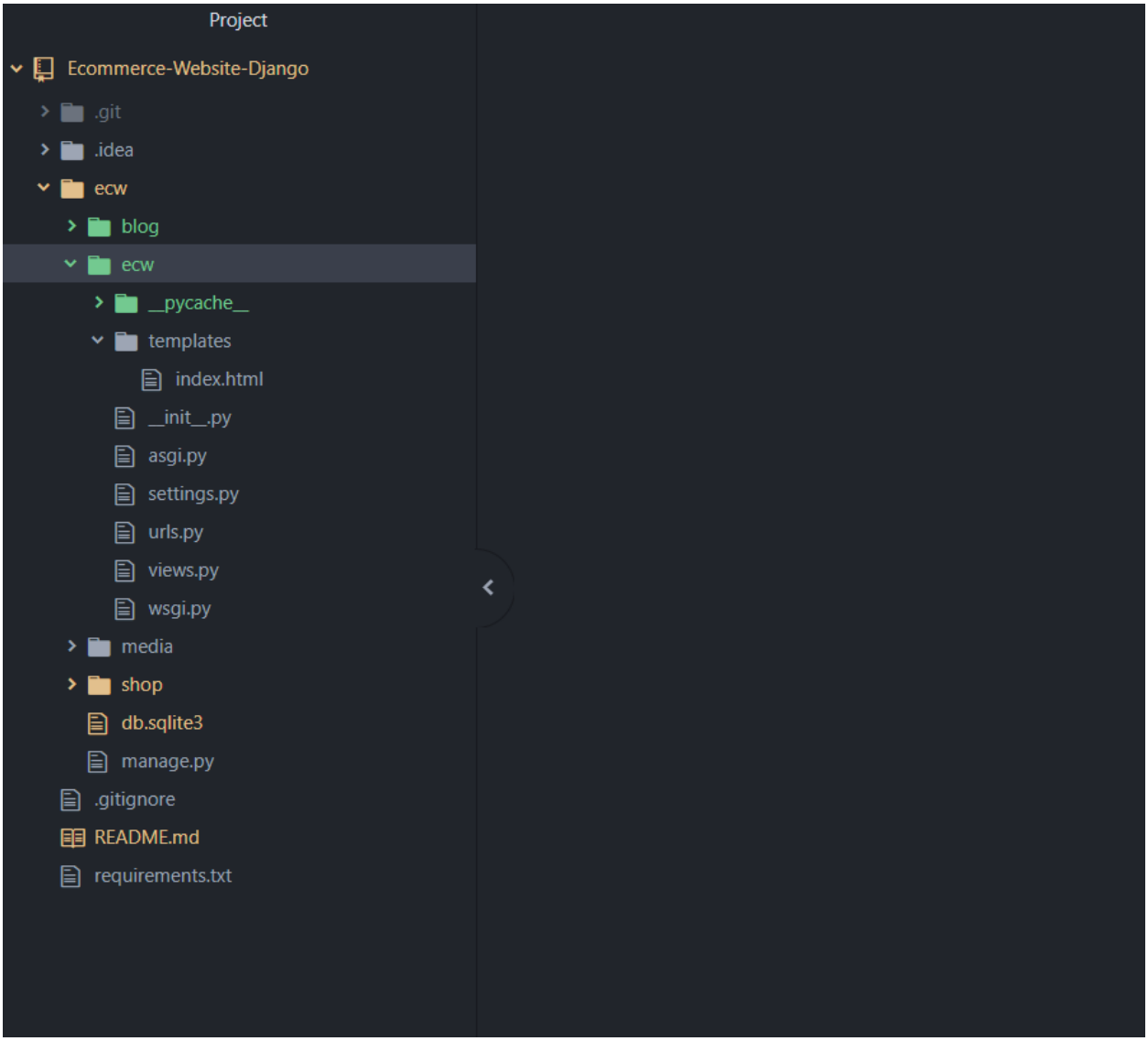
  - shop

- /templates (contains html pages)
    - index.html
    - about.html
    - basic.html
    - checkout.html
    - contact.html
    - paymentstatus.html
    - paytm.html
    - seatch.html
    - tracker.html
  - urls.py
  - views.py
  - models.py
- manage.py
- requirements.txt
- README.md

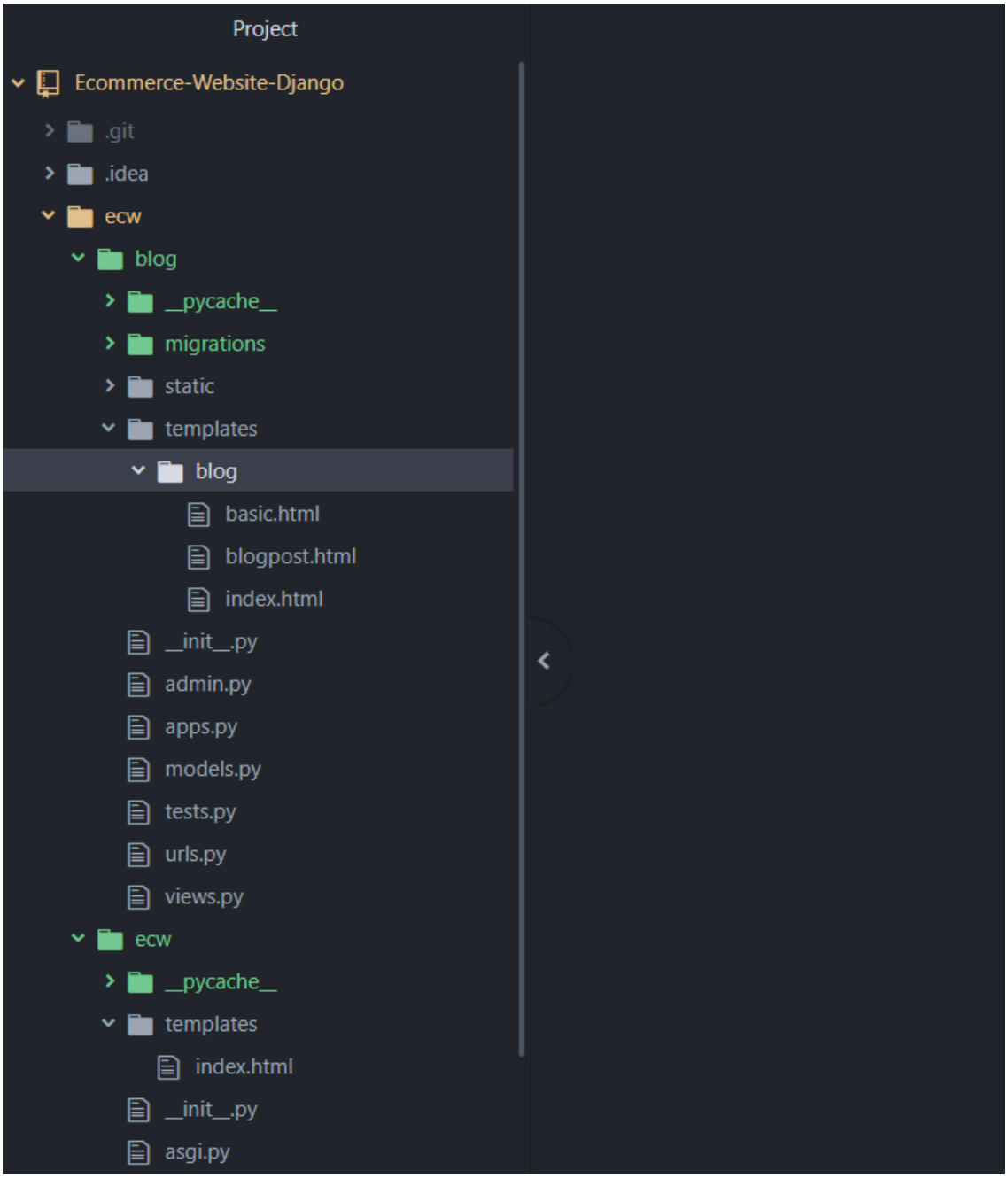
4.6 Shop Module structure:


Ecommerce-Website-Django
.git
.idea
ecw
blog
ecw
media
shop
__pycache__
images
migrations
paytm
static
shop
p1.webp
prod.jpeg
templates
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
db.sqlite3
manage.py
.gitignore
README.md
requirements.txt

4.7 Ecw module structure:



4.8 Blog Module Structure:



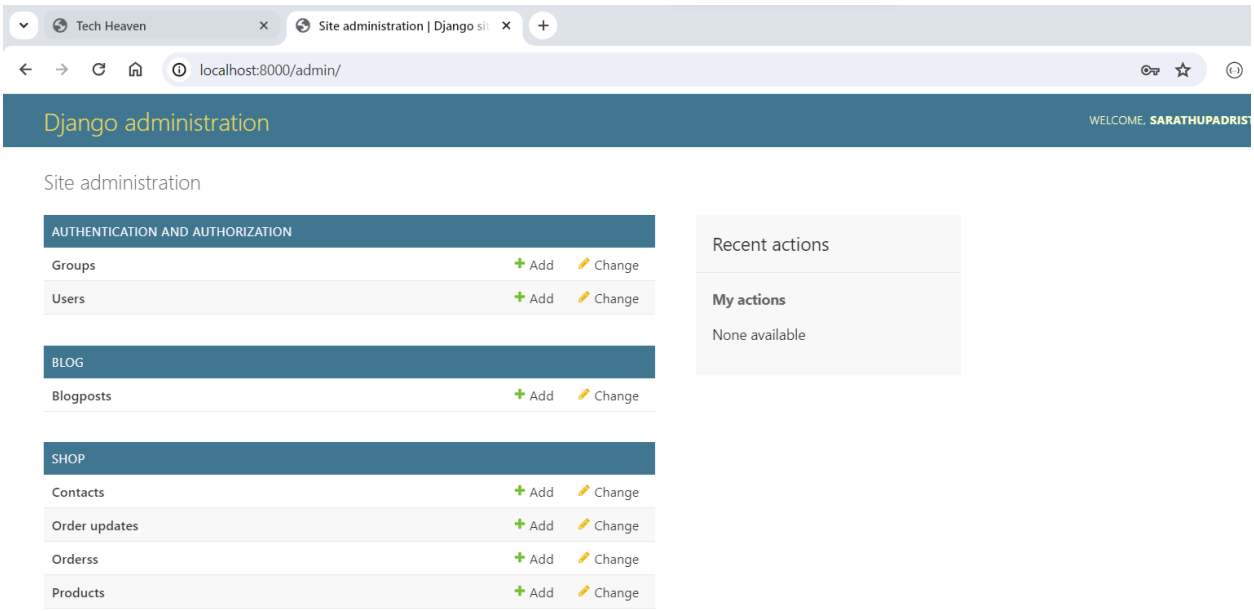
5. Running the Project:

1. Clone the project repository.
2. Install the required dependencies using pip install
   asgiref==3.2.7
   Django==3.0.6
   Pillow==7.1.2
   pycryptodome==3.9.7
   pytz==2020.1
   sqlparse==0.3.1

3. Run the Django development server using python manage.py runserver.
4. Access the website by navigating to http://localhost:8000 in your web browser.

6. Technical Details:

6.1 SQL Lite Database Tables:



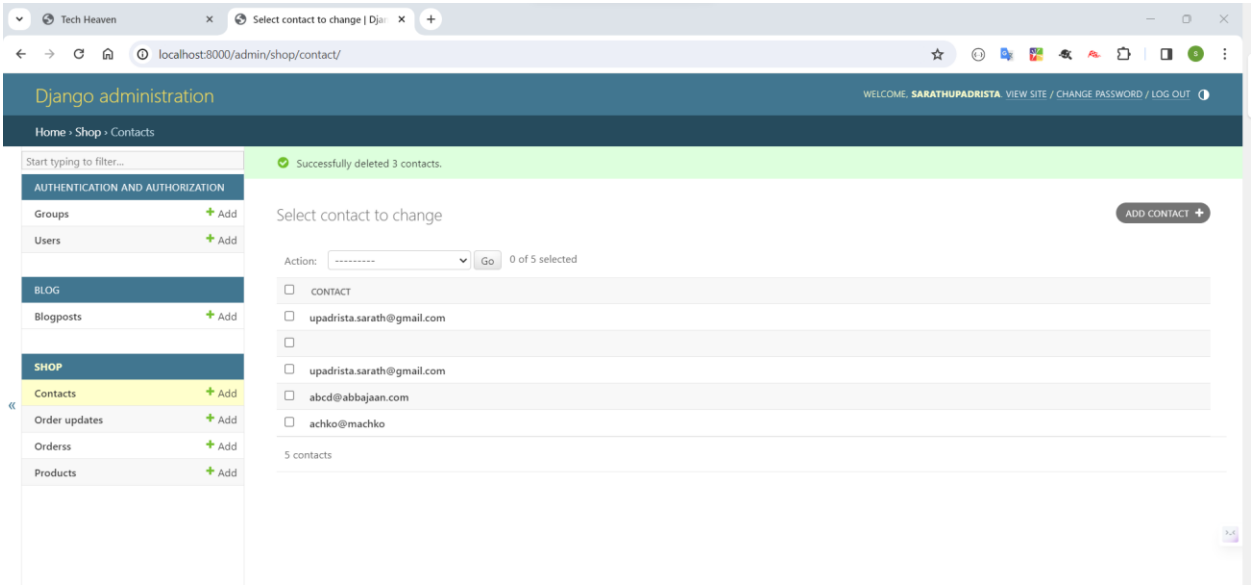
6.2 Shop Module Tables:

- Contacts
- Order Updates
- Orderss

- Products

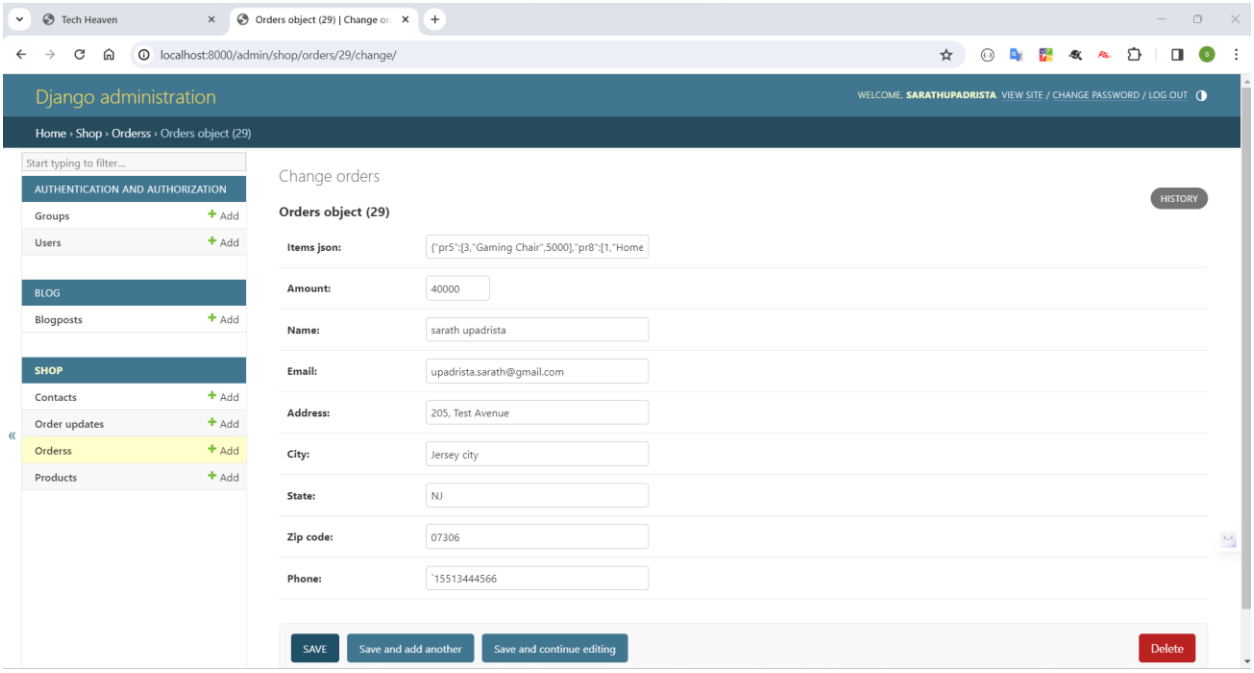
Contacts:

This will have the login contact information

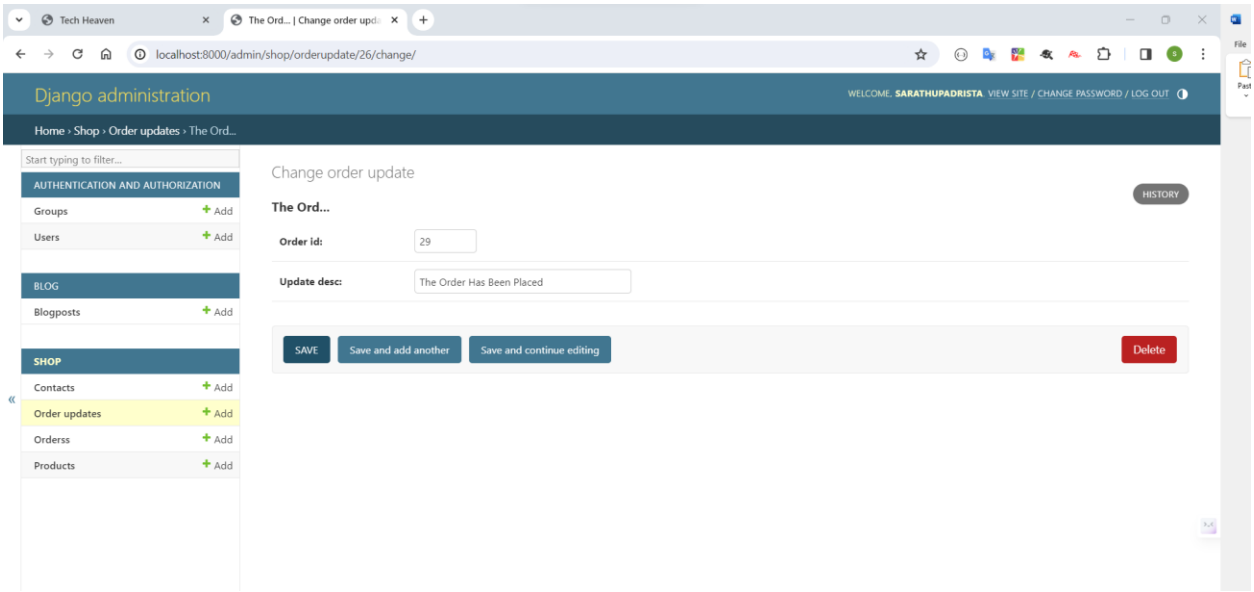


Orderss: This will have the details entered during the checkout process

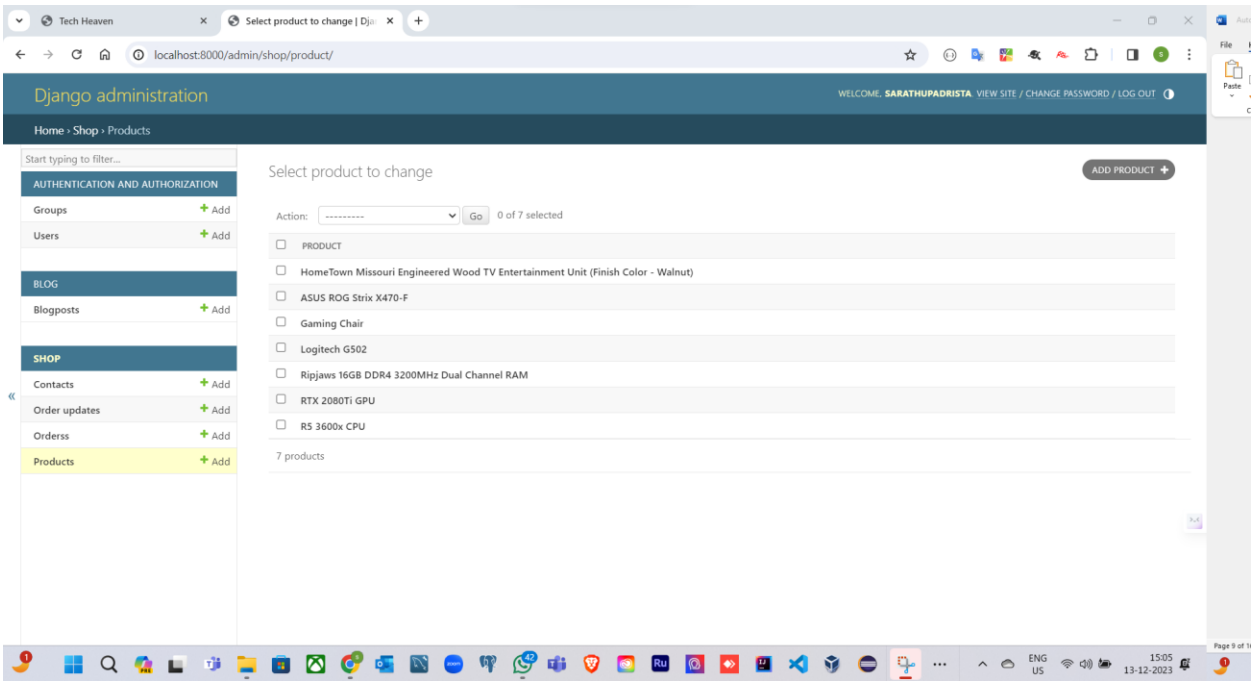
Eg: address, cart items etc.

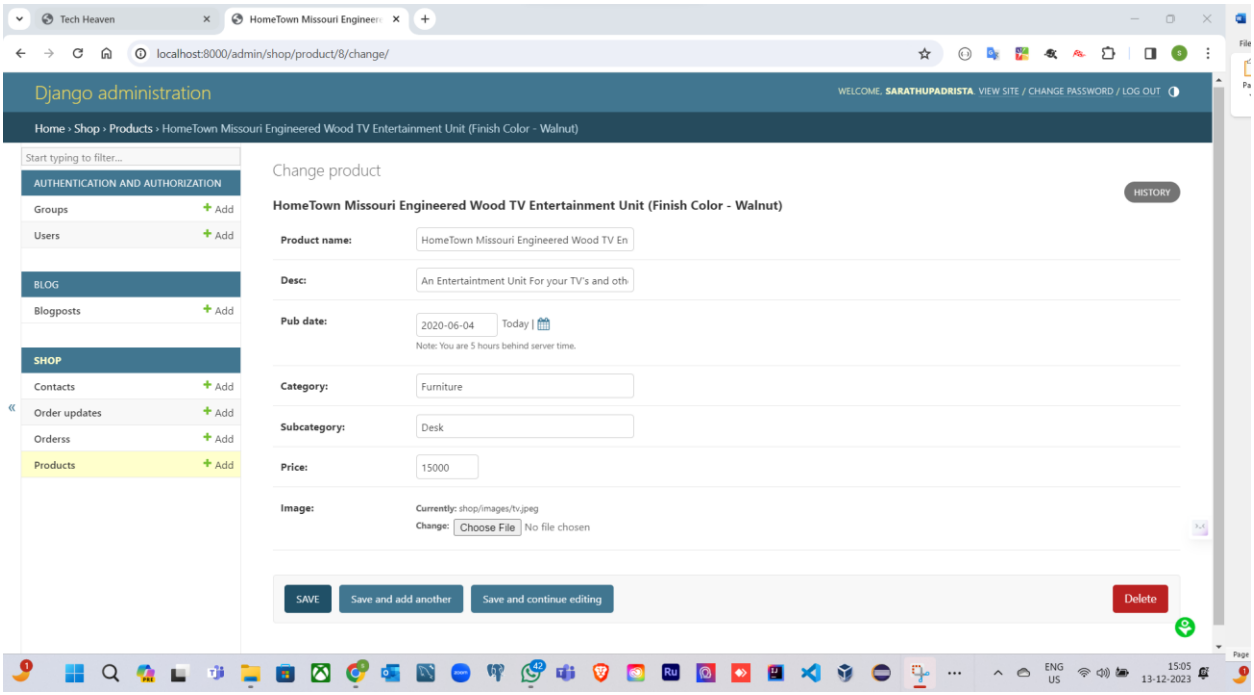


Order Updates: After the payment is done, this table gets updated and order id is generated.

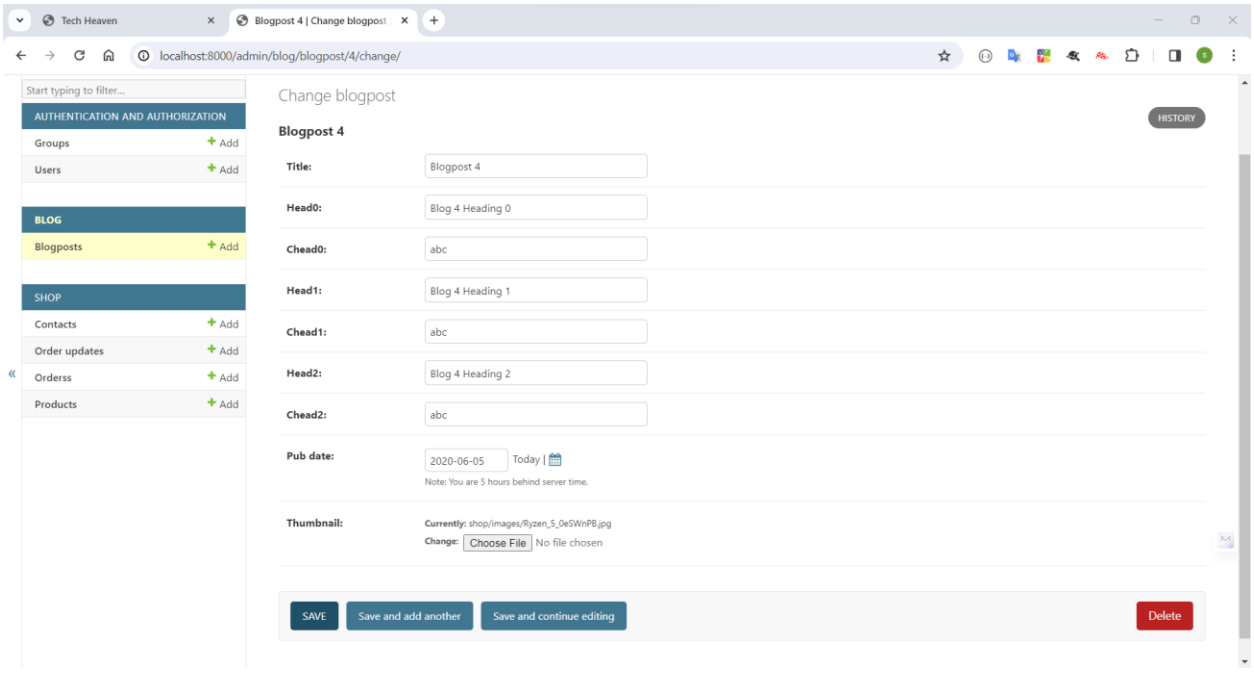


Products: This contains the products information, which is displayed in the home page





Blogpost: This contains the blogs information, which is displayed in the blogs page


Tech Heaven
Blogpost 4 | Change blogpost
localhost:8000/admin/blog/blogpost/4/change/
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
BLOG
Blogposts + Add
SHOP
Contacts + Add
Order updates + Add
Orderss + Add
Products + Add
Change blogpost
HISTORY
Blogpost 4
Title: Blogpost 4
Head0: Blog 4 Heading 0
Chead0: abc
Head1: Blog 4 Heading 1
Chead1: abc
Head2: Blog 4 Heading 2
Chead2: abc
Pub date: 2020-06-05 Today |
Note: You are 5 hours behind server time.
Thumbnail: Currently: shop/images/Ryzen_5_0eSWnPB.jpg
Change: Choose File No file chosen
SAVE
Save and add another
Save and continue editing
Delete

7. **Source Code**

```
from django.shortcuts import render

from django.http import HttpResponse

from .models import Product, Contact, Orders, OrderUpdate

from math import ceil

from django.views.decorators.csrf import csrf_exempt

from .paytm import checksum

import json

# Create your views here.

MERCHANT_KEY = 'xxxxxxxxxxxxxxxx'

def index(request):

    allprods = []



    catprods = Product.objects.values('category','id')

    cats = {item ['category'] for item in catprods}

    for cat in cats:
```

```
        prod = Product.objects.filter(category=cat)
        n = len(prod)
        nSlides = n // 4 + ceil((n / 4) - (n // 4))
        allprods.append([prod, range(1,nSlides),nSlides])
    params = {'allProds' : allprods}
    return render(request,'shop/index.html', params)
def searchMatch(query,item):
    if query in item.desc.lower() or query in item.product_name.lower() or query
in item.category.lower():
        return True
    else:
        return False


def search(request):
    query = request.GET.get('search')
    allprods = []
```

```
    catprods = Product.objects.values('category','id')
    cats = {item ['category'] for item in catprods}
    for cat in cats:
        prodtemp = Product.objects.filter(category=cat)
        prod = [item for item in prodtemp if searchMatch(query, item)]


        n = len(prod)
        nSlides = n // 4 + ceil((n / 4) - (n // 4))
        if len(prod) !=0:
            allprods.append([prod, range(1,nSlides),nSlides])
    params = {'allProds' : allprods,'msg':""}
    if len(allprods) == 0 or len(query)<4:
        params = {'msg' : "Please make sure to enter relevant search query"}
    return render(request,'shop/index.html', params)


def about(request):
```

```
    return render(request, 'shop/about.html')

def contact(request):
    if request.method =="POST":
        print(request)
        email= request.POST.get('email','')
        msg =request.POST.get('msg','')
        contact = Contact(c_email=email, c_msg=msg)
        contact.save()
        submit = True
        return render(request,'shop/contact.html',{'submit':submit})
    return render(request,'shop/contact.html')
def tracker(request):
    if request.method=="POST":
        OrderId = request.POST.get("orderId", '')
        email = request.POST.get('email', '')
        try:
```

```
            order = Orders.objects.filter(order_id=OrderId, email=email)
            if len(order)>0:
                Update = OrderUpdate.objects.filter(order_id=OrderId)
                updates = []
                for item in Update:
                    updates.append({'text':item.update_desc,'time':item.timestamp})
                    response = json.dumps({"status": "success", "updates": updates,
"itemsJson" :order[0].items_json},default=str)
                    return HttpResponse(response)
            else:
                return HttpResponse('{"status":"noitem"}')
        except Exception as e:
            return HttpResponse('{"status":"error"}')
    return render(request,'shop/tracker.html')


def productview(request,myid):
```

```
    #Fetch the product using id
    product = Product.objects.filter(id=myid)
    return render(request,'shop/prodview.html',{'product':product[0]})
def checkout(request):
    if request.method =="POST":
        print(request)
        name = request.POST.get('name','')
        amount = request.POST.get('amount','')
        items_json = request.POST.get('itemsjson','')
        email= request.POST.get('email','')
        address= request.POST.get('address1','') + " "
+request.POST.get('address2','')
        city= request.POST.get('city','')
        state= request.POST.get('state','')
        zip_code= request.POST.get('zip','')
        phone = request.POST.get('phone','')
```

```
        checkout =
Orders(items_json=items_json,name=name,email=email,address=address,city=c
ity,state=state,zip_code=zip_code,phone=phone,amount=amount)

        checkout.save()

        update = OrderUpdate(order_id=checkout.order_id,update_desc="The
Order Has Been Placed")

        update.save()

        thank = True

        id=update.order_id

        # return render(request,'shop/checkout.html',{'thank':thank,'id':id})

        #Request paytm to transfer amount to your account after payment by user

        # param_dict ={

        #     'MID': 'Your-Merchant-Id-Here',

        #     'ORDER_ID': str(checkout.order_id),

        #     'TXN_AMOUNT': str(amount),

        #     'CUST_ID': email,

        #     'INDUSTRY_TYPE_ID': 'Retail',
```

```
        #     'WEBSITE': 'WEBSTAGING',

        #     'CHANNEL_ID': 'WEB',

        #     'CALLBACK_URL':'http://127.0.0.1:8000/shop/handlerequest/',

        # }

        # param_dict['CHECKSUMHASH'] =
checksum.generate_checksum(param_dict, MERCHANT_KEY)

        # return render(request,'shop/paytm.html',{'param_dict' : param_dict})

        # handlerequest(request);

        response_dict = {}

        response_dict['RESPCODE'] = '01'

        response_dict['ORDERID'] = id;

        return render(request,'shop/paymentstatus.html',{'response':
response_dict})

        # return render(request,'shop/handlerequest/')

    return render(request,'shop/checkout.html')
```

```
@csrf_exempt
def handlerequest(request):
    response_dict['RESPCODE'] == '01'
    #Paytm will send you post request here
    # form = request.POST
    # response_dict = {}
    # for i in form.keys():
    #     response.dict[i] = form[i]
    #     if i =='CHECKSUMHASH':
    #         checksum = form[i]
    # verify = checksum.verify_checksum(response_dict, MERCHANT_KEY,
checksum)
    # if verify:
    #     if response_dict['RESPCODE'] == '01':
    print('order successful')
        # else:
```

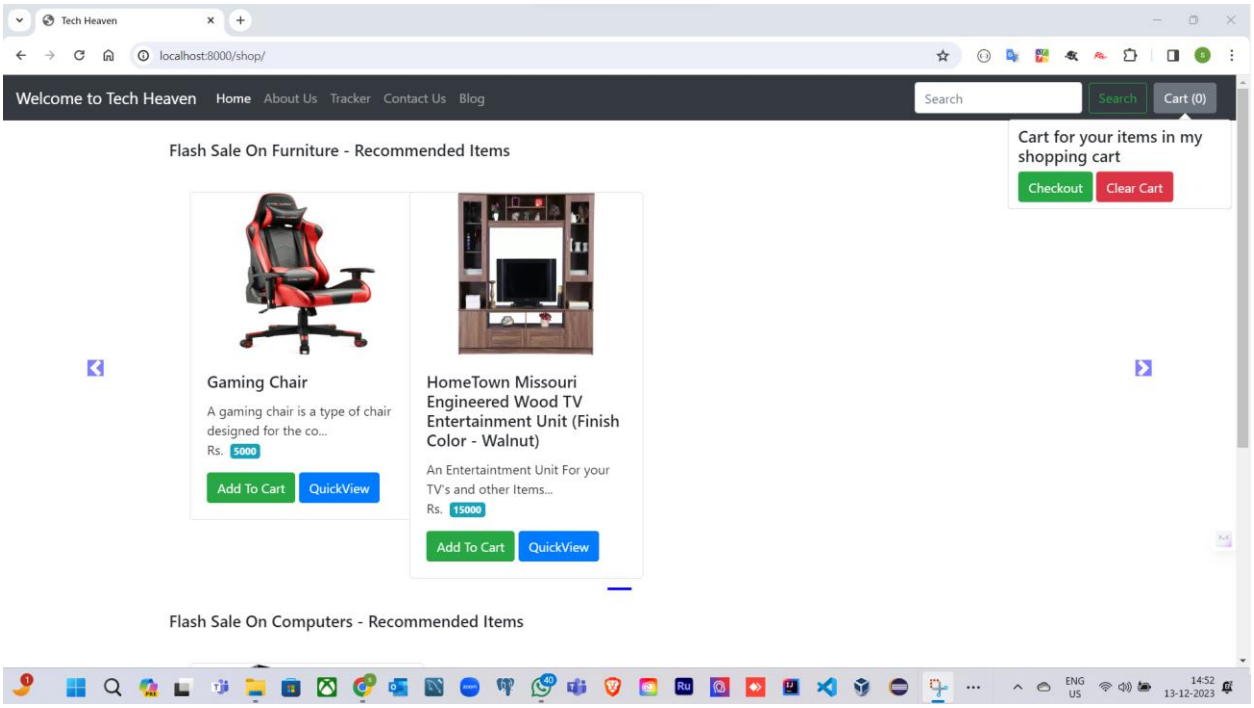
 # print('order was not successful because ' + response_dict['RESPMSG'])

return render(request,'shop/paymentstatus.html',{'response': response_dict})

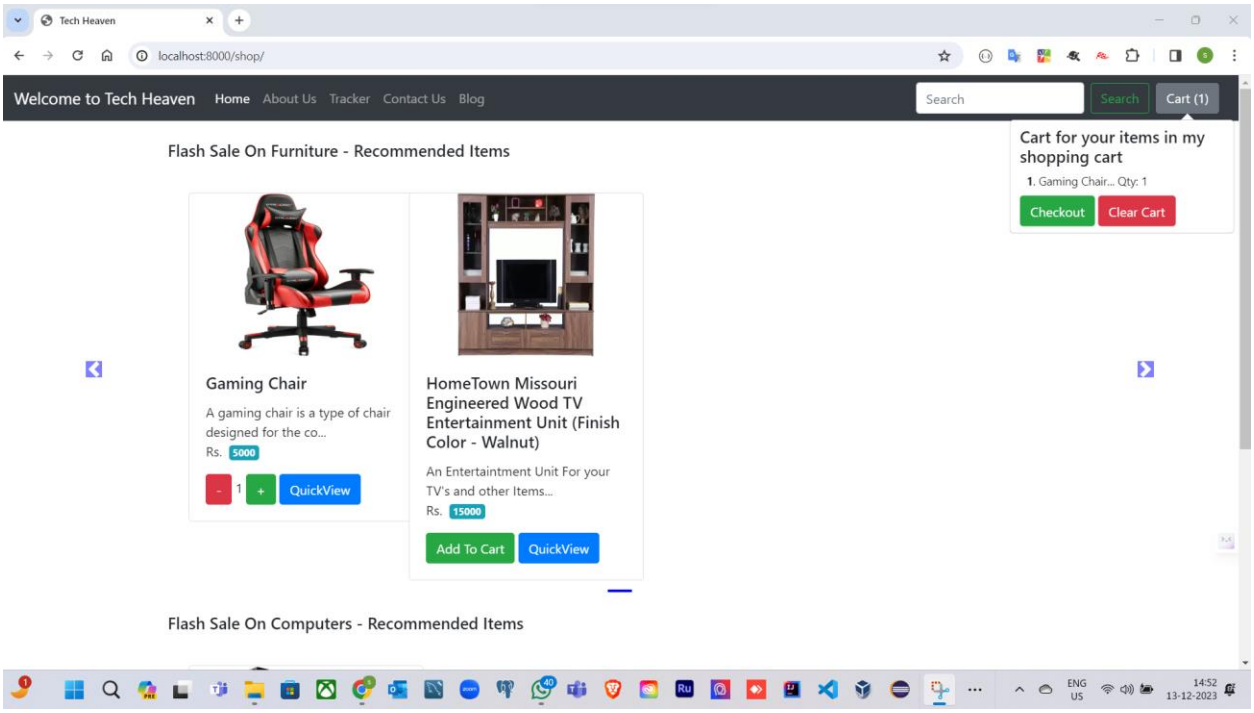
pass

8. Screenshots of the project:

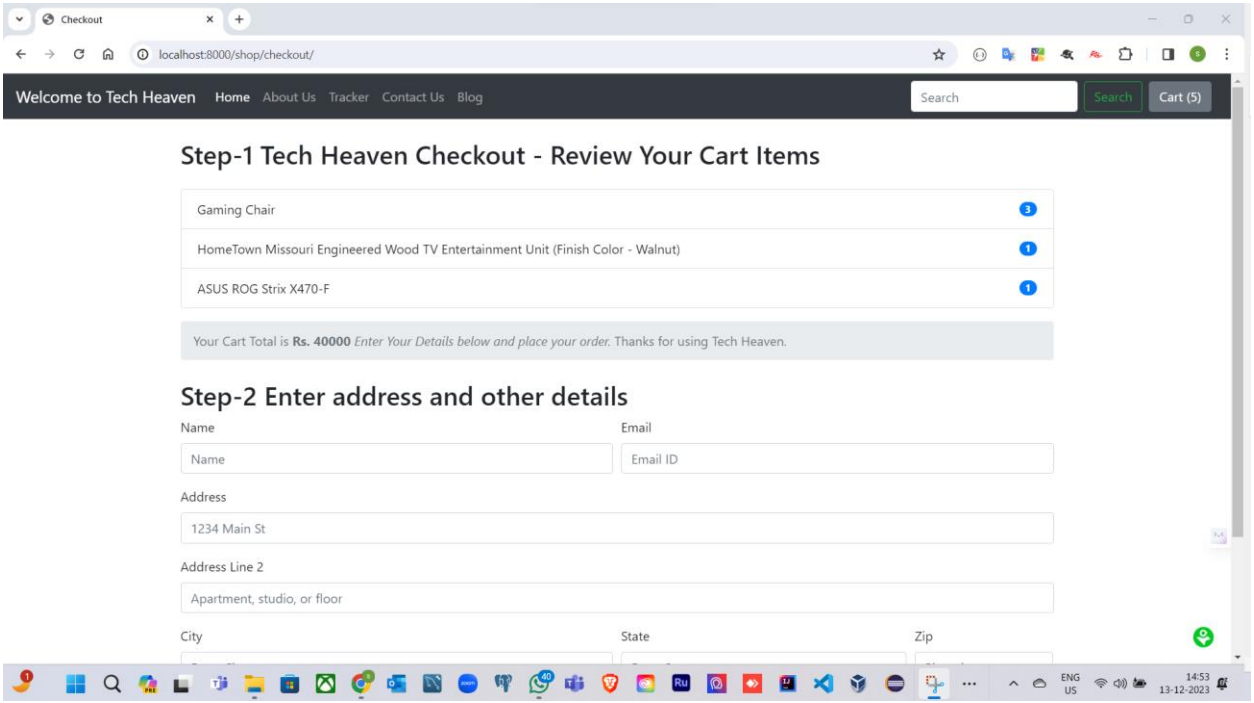
8.1 Home Page:

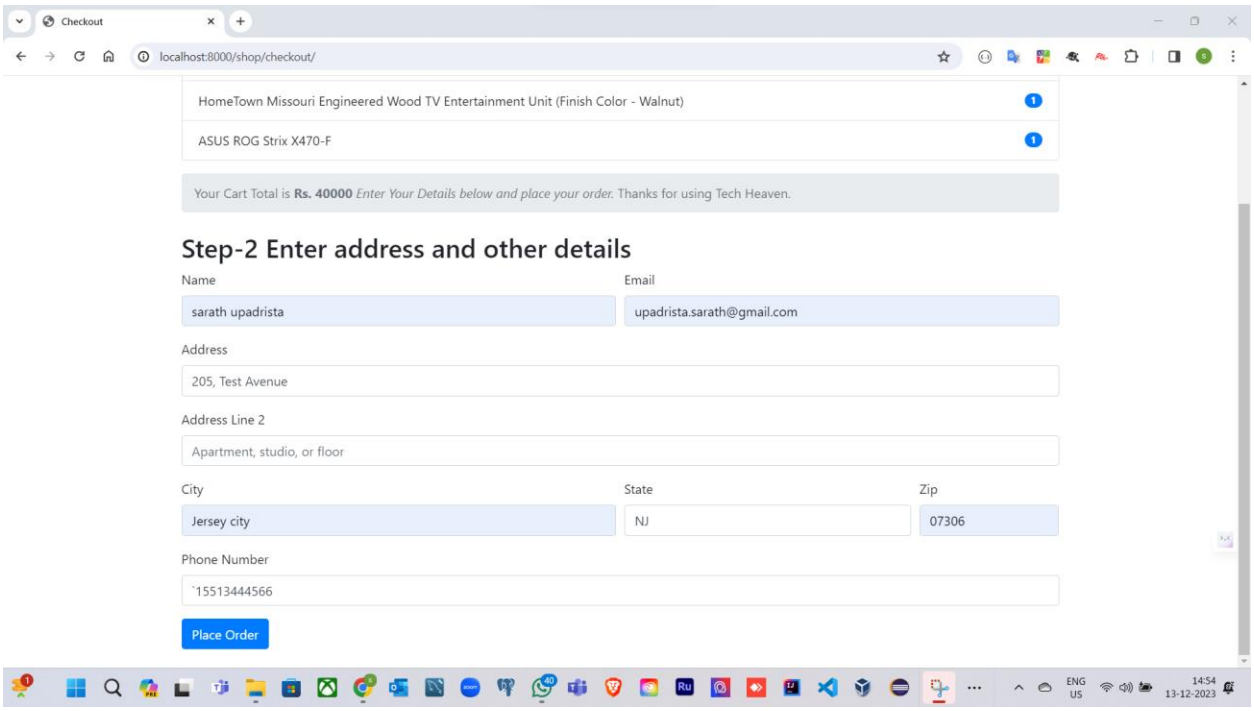


8.2 Cart screen:

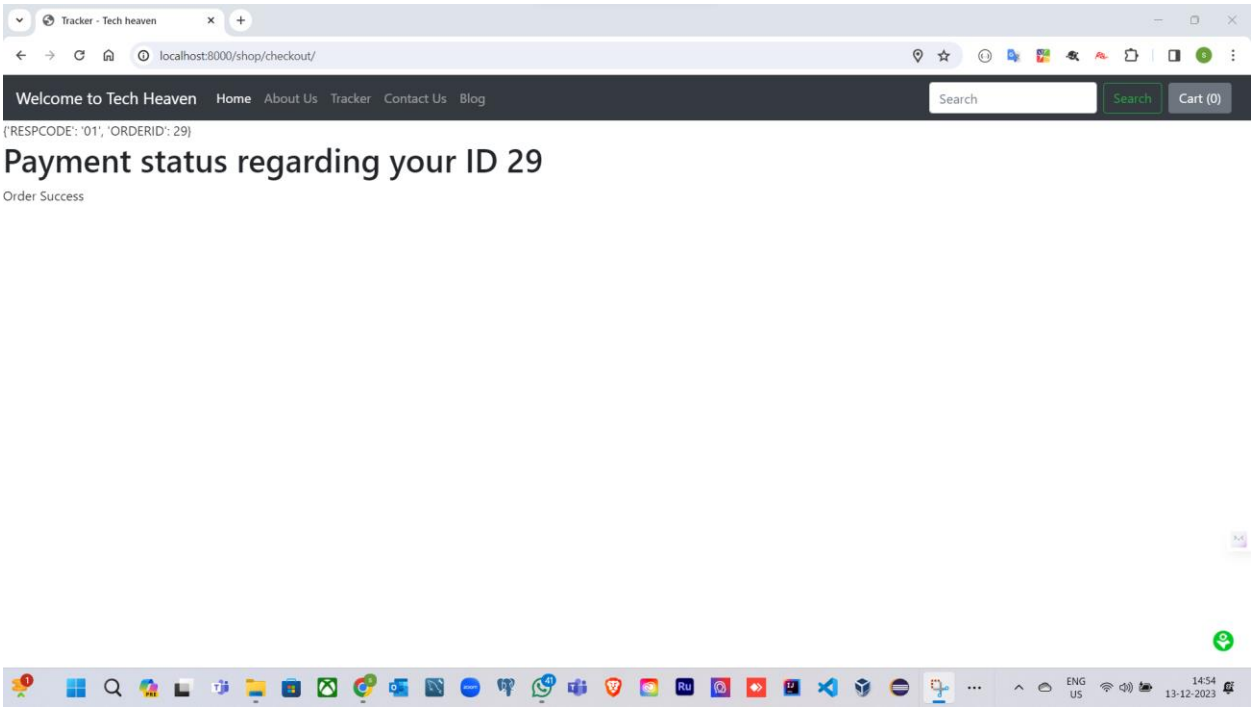


8.3 Checkout Page:

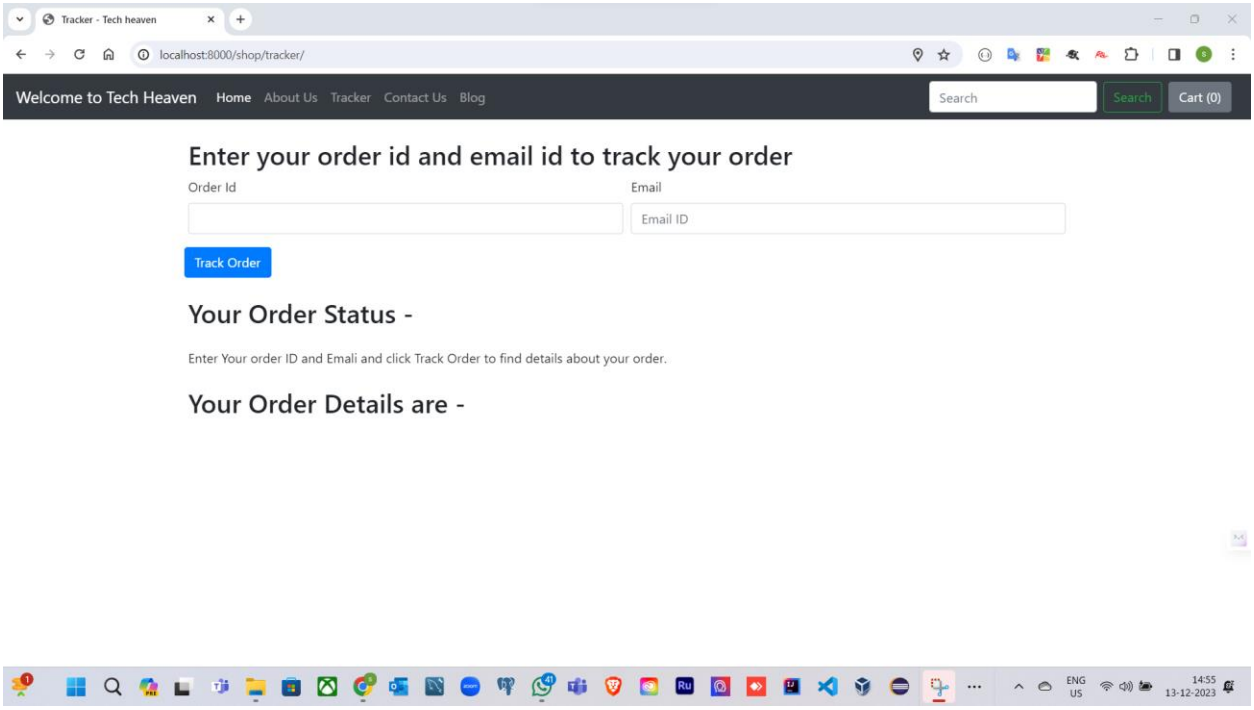


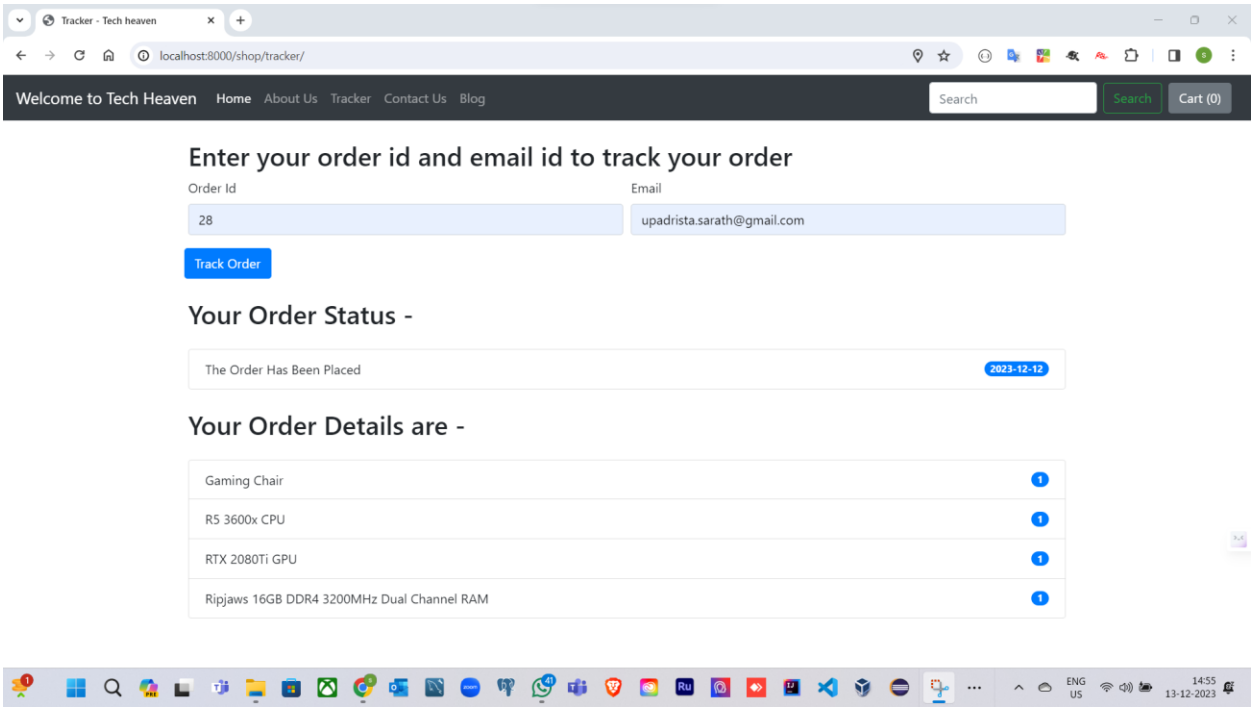


8.4 Payment success page:

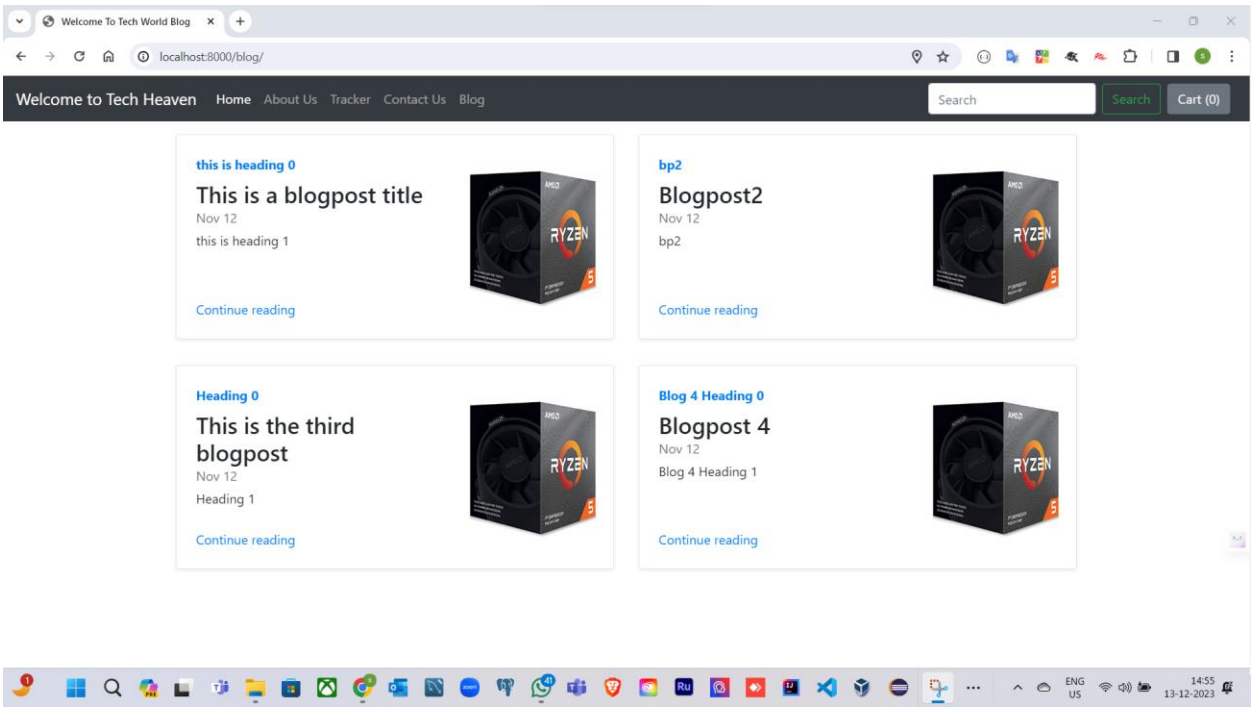


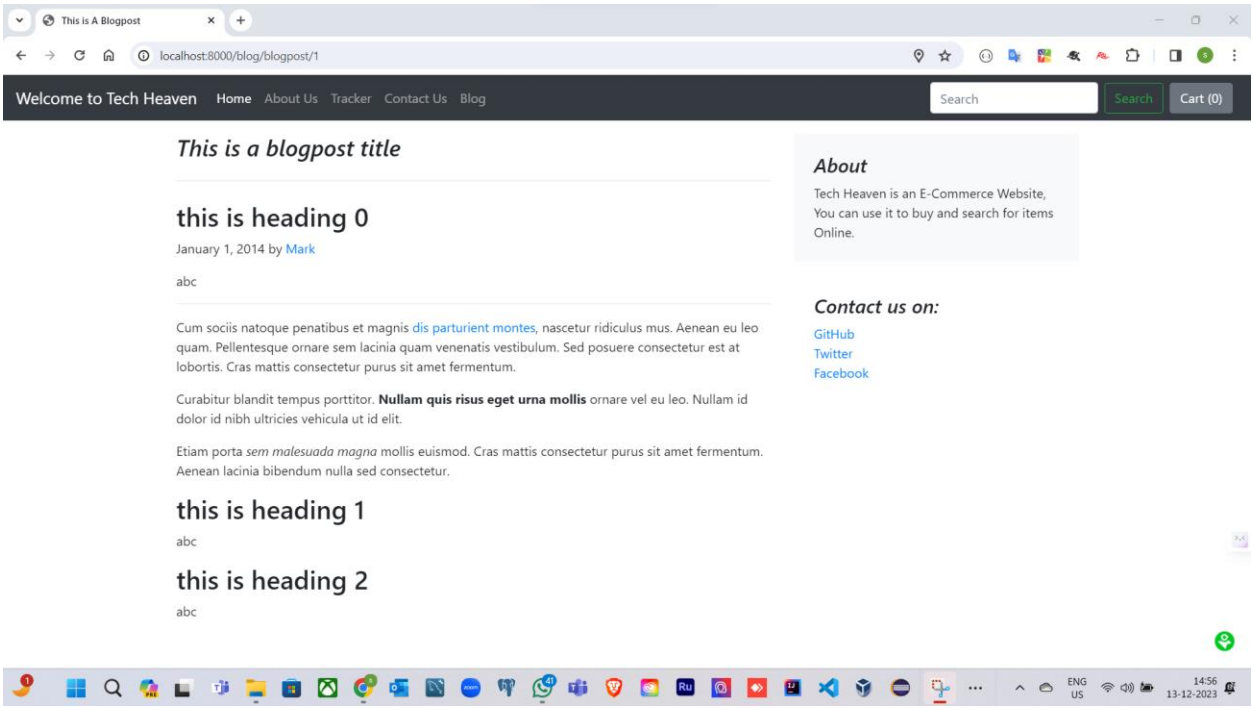
8.5 Tracker Page:



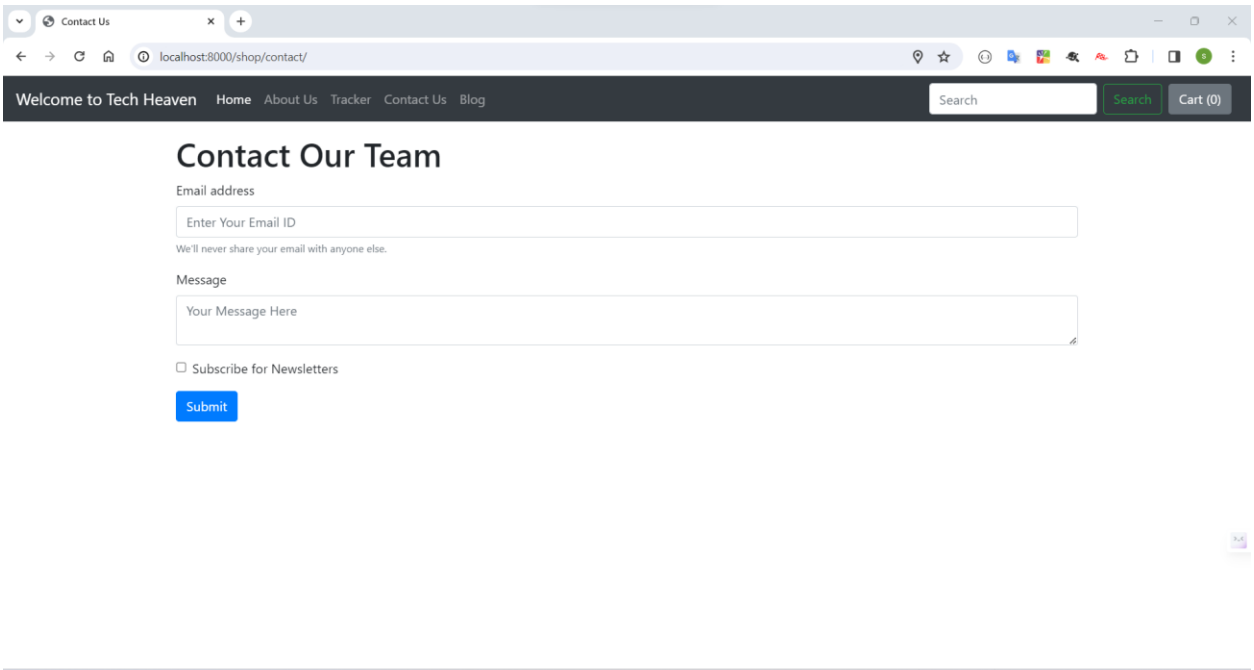


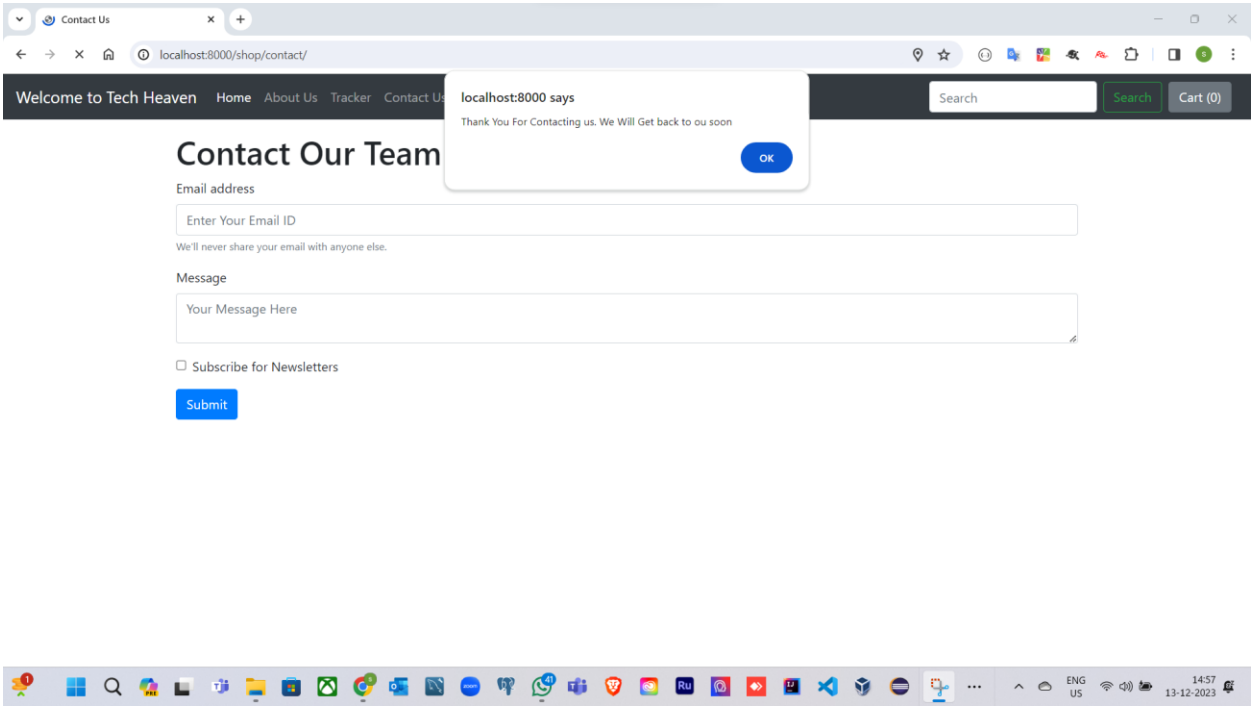
8.6 Blog Page:


Welcome To Tech World Blog
localhost:8000/blog/
Welcome to Tech Heaven
Home
About Us
Tracker
Contact Us
Blog
Search
Search
Cart (0)
this is heading 0
This is a blogpost title
Nov 12
this is heading 1
Continue reading
bp2
Blogpost2
Nov 12
bp2
Continue reading
Heading 0
This is the third blogpost
Nov 12
Heading 1
Continue reading
Blog 4 Heading 0
Blogpost 4
Nov 12
Blog 4 Heading 1
Continue reading


This is A Blogpost
localhost:8000/blog/blogpost/1
Welcome to Tech Heaven
Home
About Us
Tracker
Contact Us
Blog
Search
Search
Cart (0)
This is a blogpost title
this is heading 0
January 1, 2014 by Mark
abc
Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Sed posuere consectetur est at lobortis. Cras mattis consectetur purus sit amet fermentum.
Curabitur blandit tempus porttitor. Nullam quis risus eget urna mollis ornare vel eu leo. Nullam id dolor id nibh ultricies vehicula ut id elit.
Etiam porta sem malesuada magna mollis euismod. Cras mattis consectetur purus sit amet fermentum. Aenean lacinia bibendum nulla sed consectetur.
this is heading 1
abc
this is heading 2
abc
About
Tech Heaven is an E-Commerce Website, You can use it to buy and search for items Online.
Contact us on:
GitHub
Twitter
Facebook

8.7 Contact Us Page:





9. Enhancements:

In addition to the core functionalities outlined earlier, the E-commerce project can be further enriched with key enhancements to provide an even more comprehensive and user-friendly online shopping experience. The following features will be incorporated into the existing framework:

9.1 Payment Modules:

Integrate proper payment modules with secure payment gateways to facilitate seamless transactions. This enhancement ensures that users can securely make online payments, adding a crucial layer of convenience and trust to the shopping process.

9.2 User Authentication and Login Mechanism:

Implement a robust user authentication system with a login mechanism. This enables users to create accounts, log in, and access personalized features such as order history and saved carts. User accounts enhance security, provide a personalized experience, and streamline the checkout process.

9.3 User-Specific Cart Information:

Enhance the Cart Management module to list information based on individual users. This feature allows users to view their specific carts, add or remove items, and seamlessly resume shopping from where they left off. It contributes to a personalized and user-centric shopping **experience.**

9.4 Transaction History:

Introduce a Transaction History module to provide users with a comprehensive overview of their past purchases. This feature allows users to track and review their order history, including details such as order date, products purchased, and transaction amounts. A transaction history adds transparency and assists users in keeping track of their **spending.**

9.5 Customer Support Page:

Add a dedicated Customer Support page to the website. This page can include contact information, frequently asked questions (FAQs), and a contact form for users to submit inquiries or issues. Offering robust customer support enhances user confidence and satisfaction, addressing any concerns they may have during their shopping journey.

9.6 Payment Gateway Integration:

Explore and integrate reliable payment gateways such as Stripe, PayPal, or others based on project requirements and preferences.

9.7 User Authentication and Login:

Leverage Django's built-in authentication system to implement secure user authentication and login functionalities.

9.8 User-Specific Cart Information:

Extend the existing backend logic using Django to associate carts with specific user accounts.

9.9 Transaction History:

Utilize the database to store and retrieve transaction-related data, creating a seamless flow between the frontend and backend.

9.10 Customer Support Page:

HTML and CSS can be used to create an informative and visually appealing Customer Support page, and Django can handle the backend logic for form submissions.

10. Conclusion:

The E-commerce project represents a sophisticated and feature-rich online shopping platform designed to offer users a seamless and highly engaging shopping experience. Leveraging advanced technologies such as Python, Django, and SQLite, along with HTML and CSS for an intuitive user interface, the project encompasses essential modules and key enhancements.

The core modules, including the Home Page, Cart Management, Checkout Page, and Order Tracker, lay the foundation for a streamlined and efficient shopping journey. The Home Page serves as a user-friendly entry point, the Cart Management module ensures flexible control over selected products, the Checkout Page simplifies the payment process, and the Order Tracker enhances post-purchase transparency.

Furthermore, the project has been enriched through strategic enhancements. The integration of proper payment modules with secure gateways ensures the security of financial transactions. The implementation of a robust user authentication system, along with user-specific cart information and transaction history, enhances personalization and convenience. The addition of a dedicated Customer Support page further strengthens user trust by providing accessible assistance.

In conclusion, the E-commerce project not only meets the fundamental requirements of an online shopping platform but also goes beyond by incorporating features that contribute to user

satisfaction, transparency, and trust. The careful selection of technologies, thoughtful design, and continuous improvement through enhancements position this project as a robust and user-centric solution in the competitive landscape of e-commerce. As the digital economy evolves, this project stands ready to adapt and provide a cutting-edge shopping experience for users around the globe.