A Project Report On

ONLINE SHOPPING SYSTEM

Submitted in partial fulfillment of the

Requirements for the award of the Degree of

BACHELOR OF COMPUTER APPLICATION

By

Shivam Chaubey

AJU/190398

Under the esteemed guidance of

DR. ARUN KUMAR MARANDI

(Internal Guide)

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

ARKA JAIN UNIVERSITY, JHARKHAND

JAMSHEDPUR

2019-2022

ONLINE SHOPPING SYSTEM

A Project Report

Submitted in partial fulfillment of the

Requirements for the award of the Degree of

BACHELOR OF COMPUTER APPLICATION

By

Shivam Chaubey

AJU/190398

Under the esteemed guidance of

Dr. Arun Kumar Marandi

(Internal Guide)

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

ARKA JAIN UNIVERSITY, JHARKHAND

JAMSHEDPUR

2019-2022

ARKA JAIN UNIVERSITY, JHARKHAND

JAMSHEDPUR

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, **"ONLINE SHOPPING SYSTEM"**, is the bonafied work of **SHIVAM CHAUBEY** bearing Registration Number: **AJU/190398** submitted in partial fulfillment of the requirements for the award of the degree of BACHELOR OF COMPUTER APPLICATION from ARKA JAIN UNIVERSITY, JHARKHAND.

Internal Guide

Head of Department

External Examiner

University Seal

Date: 25/05/22

ULTRAINFO TECHNOLOGIES
ENGINEERS

Ref. No: ITD-PI-0322-011 **Date:** 07-03-2022

PROJECT INTERNSHIP LETTER

TO WHOM IT MAY CONCERN

This is to certify that Mr. **Shivam Chaubey**, a student of ARKA JAIN University, Jharkhand, Jamshedpur has successfully completed his Project Internship/Training (Virtual) with our organization as a **Full Stack Python Developer** during the period of January 2022 to February 2022.

The topic of the project was:

"Online Shopping System"

The Frontend and backend used for project development was HTML, CSS, JavaScript, BootStrap, DJANGO, PYTHON, SQLITE4.

During this period of internship/training, he was found to be honest, creative & able to perform all his duties perfectly on the project.

We wish him all the best in all his future accomplishments.

For ULTRAINFO TECHNOLOGIES

ULTRAINFO TECHNOLOGIES INDIA

(Authorized Signatory)

Address: Rourkela, Odisha, India

Website: itdeveloper.in **E-mail:** info@itdeveloper.in

ABSTRACT

Online Shopping System is a web-based project which is made for remote-shopping or shopping through Internet. As the technology is being advanced the way of life is changing accordance. Now a days we can place the order for any thing from our home. There is no need to go the shop of the things we want. The order can be placed online through Internet. The payment, the confirmation of purchasing, we can do everything we want. Now we can think that how the days have been changed with time. People had to stand in rows to wait there terms to buy a particular thing from a popular shop. But what is happening now a days, we can extremely surprise that those things can be available on the door-step in few hours. People had to suffer the rush of the market when they went for shopping. They used to think hundred times to buy anything having the sufficient money for shopping. The problem was the rush; the quarrel at the time of buying the things. But the advancement of technology brought the new way for shopping. The way of shopping has completely changed with the coming of Internet Technology. People have to fill a simple form on the internet to place their order on any popular shop or shopping-mall for the thing they want to buy. Now they can place their order from the home. This project entitled Online Shopping is an implementation of the above description. It means, it implements the E-shopping or in other wold shopping through Internet. It lets the user to place their order online for any article.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to several individuals and organization for supporting me throughout the completion of my project.

First, I wish to express my sincere gratitude to my mentor **Dr. Arun Kumar Marandi** for his enthusiasm patience, insightful comments, helpful information, practical advices and unceasing ideas that have helped me tremendously at all times in my Project and writing of these thesis. His immense knowledge, profound experience and professional expertise in Backend has enabled me to complete this project successfully. Without his support and guidance, this project would not have been possible.

I am also thankful to our respected H.O.D **Dr. Arvind Kumar Pandey** for his loving inspiration and timely guidance. I also wish to express my sincere thanks to the Department of Computer Science & Information Technology of ARKA JAIN UNIVERSITY for accepting this project.

I am thankful to all my friends who helped me out in the corrections and suggestions for the various aspects of this project.

And finally, I would like to thank my parents who always supported me at all times by their love, confession and support. Without their constant support, I couldn't have completed this project in time.

I thank all the other people who where directly or indirectly involved for the completion of this project.

DECLARATION

I hereby declare that the Project entitled **"ONLINE SHOPPING SYSTEM"** done at **ARKA JAIN UNIVERSITY** has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

This project is done in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF COMPUTER APPLICATION** to be submitted as final semester project as part of our curriculum.

Shivam Chaubey

SHIVAM CHAUBEY

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 : INTRODUCTION

1.1 OVERVIEW :

The ‘Online Shopping System’ Services department strives to provide solutions to develop and transfer easy and efficient way in the digital age and to help reduces the human pressure and time.To help support shop collections, the digital initiatives, and external partner institution digital projects, It provide services that include the digitization of analog objects, metadata management, digital preservation, and discovery and access of digital collections. “Online Shopping System” is a web application written for all operating systems, designed to help users maintain and organize shop virtually. This software is easy to use for both beginners and advanced users. It features a familiar and well thoughtout, an attractive user interface, combined with strong searching Insertion and reporting capabilities. The report generation facility of shop system helps to get a good idea of which are the various items brought by the members, makes users possible to get the product easily.

1.2 BACKGROUND STUDY:

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace.

The objective of this project is to develop a general-purpose e-commerce store where any product (such as books, CDs, computers, mobile phones, electronic items, and home appliances) can be bought from the comfort of home through the Internet. However, forimplementation purposes, this paper will deal with an online ecommerce store.

An online store is a virtual store on the Internet where customers can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction.

Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as a credit card number. An email notification is sent to the customer as soon as the order is placed.

1.3 PROJECT PLANNING:

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. The logical dependencies between tasks are defined using an activity network diagram that enables identification of the critical path. Float or slack time in the schedule can be calculated using project management software. Then the necessary resources can be estimated and costs for each activity can be allocated to each resource, giving the total project cost. At this stage, the project plan may be optimized to achieve the appropriate balance between resource usage and project duration to comply with the project objectives. Once established and agreed, the plan becomes what is known as the baseline. Progress will be measured against the baseline throughout the life of the project

1.4 OBJECTIVE:

The project is about to handle all the information of the shop regarding members. Also it manages resources which were managed and handled by manpower previously. The main purpose of the project is to integrate distinct sections of the shop into consistent manner so that complex functions can be handled smoothly. The project aims at the following matters

- Automation of product manipulation.
- Buying products.
- To manage information of different types of items.
- Consistently update information of all the item.
- Managing security by providing authorized email & password.
- Manages database efficiently.

1.5 FUTURE SCOPE:

Our designed online shopping system provides a 24×7 service, that is customers can surf the website, place orders anytime they wish to. Also, the delivery system works 24×7 hours a week. Some of the features that can be modified and added to this system in the future involve its implementation by local shopkeepers, where shops will be providing an online interface to customers for shopping and placing orders.

Then some delivery persons can perform their work. This will be adding on benefit for the customers as it will save their time, plus it adds on for the shopkeepers also, as people will continue to shop from local shops rather than preferring to supermarkets every time.

Also, since the deliveries from these local vendors will not be as time-consuming as these days Flipkart, Amazon, etc. take but rather will be delivered the same day of an order placed. Else the shopkeeper can ask the customer that the product will be available by the next day, so if he/she still wants to place the order, it can be done.

Again, return or exchange will be easy since the delivery boy can even do it as the store is nearby. Including a chatbox for public benefit is also a great idea via which people can directly have a conversation with some officials regarding any type of queries.

CHAPTER 2 : SYSTEM ANALYSIS

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is a problem solving activity that requires intensive communication between the system users and system developers.

System analysis or study is an important phase of any system development process. The system is viewed as a whole, the inputs are identified and the system is subjected to close study to identify the problem areas. The solutions are given as a proposal. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal.

2.1 EXISTING SYSTEM

The present scenario for shopping is to visit the shops and market manually and then from the available product list one needs to choose the item he or she wants and then pay for the same item mainly in cash mode is done, as not every society is well educated and aware to use net banking or card modes or wallets etc.

This system is not much user-friendly as one needs to go to the market physically and then select items only from the available list. So mostly it is difficult to get the product as per our desire. Description About the products is less available and are mostly verbal only. For this type of shopping, one needs to have an ample amount of free time.

Also, not really good markets exist everywhere, so many times good markets become out of reach for certain people.

2.2 PROPOSED SYSTEM

In the proposed system customer need not go to the shop for buying the products. He can order the product he wish to buy through the application in his Smartphone. The shop owner will be admin of the system. Shop owner can appoint moderators who will help owner in managing the customers and product orders. The system also recommends a home delivery system for the purchased products.

2.3 REQUIREMENT ANALYSIS

2.3.1 GENERAL DESCRIPTION

Product Description:

The system consists of a web application which can provide the online shopping service for the customer to access the web service from his/her smartphone/tablet/PC. Web application should be able to help the customer for selecting his item and to help the owner in managing the orders from the customers.

Problem Statement:

As online shopping became a trend nowadays the regular shops are losing their customers to online brands. Customers have effortless shopping experience and saving time through shopping online. For competing with those online brands , If shops are providing an online portal where their customers can shop through internet and get the products at their doors it will increase the number of customers.

2.3.2 SYSTEM OBJECTIVES

To provide an website for online shopping of products in an existing shop.

2.3.3 SYSTEM REQUIREMENTS

2.3.3.1 NON FUNCTIONAL REQUIREMENTS

i. EFFICIENCY REQUIREMENT

When an online shopping website implemented customer can purchase product in an efficient manner.

ii. RELIABILITY REQUIREMENT

The system should provide a reliable environment to both customers and owner. All orders should be reaching at the admin without any errors.

iii. USABILITY REQUIREMENT

The website is designed for user friendly environment and ease of use.

iv. IMPLEMENTATION REQUIREMENT

Implementation of the system using css and html in front end with python as back end, will be used for database connectivity. And the database part is developed by SQLite3 provided by Django. Responsive web designing is used for makingthe website compatible for any type of screen.

v. DELIVERY REQUIREMENT

The whole system is expected to be delivered in four months of time with a weekly evaluation by the project guide.

2.3.3.2 FUNCTIONAL REQUIREMENTS

USER

USER LOGIN

Description of feature

This feature used by the user to login into system. A user must login with his user name and password to the system after registration. If they are invalid, the user not allowed to enter the system.

Functional requirement

- Username and password will be provided after user registration is confirmed.

- Password should be hidden from others while typing it in the field

REGISTER NEW USER

Description of feature

A new user will have to register in the system by providing essential details in order to view the products in the system. The admin must accept a new user by unblocking him.

Functional requirement

- System must be able to verify and validate information.

- The system must encrypt the password of the customer to provide security.

PURCHASING AN ITEM

Description of feature

The user can add the desired product into his cart by clicking add to cart option on the product. He can view his cart by clicking on the cart button. All products added by cart can be viewed in the cart. User can remove an item from the cart by clicking remove.

After confirming the items in the cart the user can submit the cart by providing a delivery address. On

successful submitting the cart will become empty.

Functional requirement

- System must ensure that, only a registered customer can purchase items.

ADMIN

MANAGE USER

Description of feature

The administrator can add user, delete user, view user and block user.

MANAGE MODERATOR

Description of feature

The administrator can add moderator, delete moderator, block moderator and search for a moderator.

MANAGE PRODUCTS

Description of feature

The administrator can add product, delete product and view product.

MANAGE ORDERS

Description of feature

The administrator can view orders and delete orders.

Functional requirements

-The system must identify the login of the admin.

-Admin account should be secured so that only owner of the shop can access that Account

MODERATOR

Description of features

A moderator is considered as a staff who can manage orders for the time being. As a future update moderator may give facility to add and manage his own products. Moderators can reduce the work load of admin. Now moderator has all the privilege of an admin having except managing other moderators. He can manage users and manage products. He can also

check the orders and edit his profile.

Functional requirement

-The system must identify the login of a moderator.

2.3 FEASIBILITY STUDY

Whatever we think need not be feasible .It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. Here the feasibility study can be performed in two ways such as technical feasibility and Economical Feasibility.

2.3.1. Technical Feasibility

It is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well. All the resources needed for the development of the software as well as the maintenance.

2.3.2. Economic Feasibility

Development of this application is highly economically feasible .The organization needed not spend much m one for the development of the system already available. The only thing is to be done is making an environment for the development with an effective supervision. I f we are doing so , we can attain the maximum usability of the corresponding resources .Even after the development , the organization will not be in a condition to invest more in the organization. Therefore , the system is economically feasible.

2.3.3. Operational Feasibility

The proposed project is beneficial only if they can be turned into information system that will meet the organization's operating requirements. It includes the training of the user on the candidate system. Care has been taken to provides the user as much as facility possible. The screen design is very much user friendly. Data entry jobs have been kept very easy and user-friendly so one day training will be sufficient to the user. The system is an event driven application. The user does not need a special training to run this application.

2.4 HARDWARE REQUIREMENTS

Name of Component	Specification
Processor	1.6 GHz or faster processor
RAM	1 GB of RAM (1.5 GB if running on a virtual machine)
Required hard disk drive	5400 RPM hard disk drive
Required hard disk space	5 GB of available hard disk space

2.5 SOFTWARE REQUIREMENTS

Name of Component	Specification
Operating System	Windows 7 and above
Front End	HTML, CSS, JAVASCRIPT
Back End	DJANGO, PYTHON
Database	SQLITE3
Web Server	HEROKU
Web Browser	Mozilla Firefox, Google Chrome, etc

2.6 DATA FLOW DIAGRAMS (DFDS)

2.6.1. Level 0 DFD

Figure 2.6.1 Level 0 DFD

2.6.2. Level 1 DFD

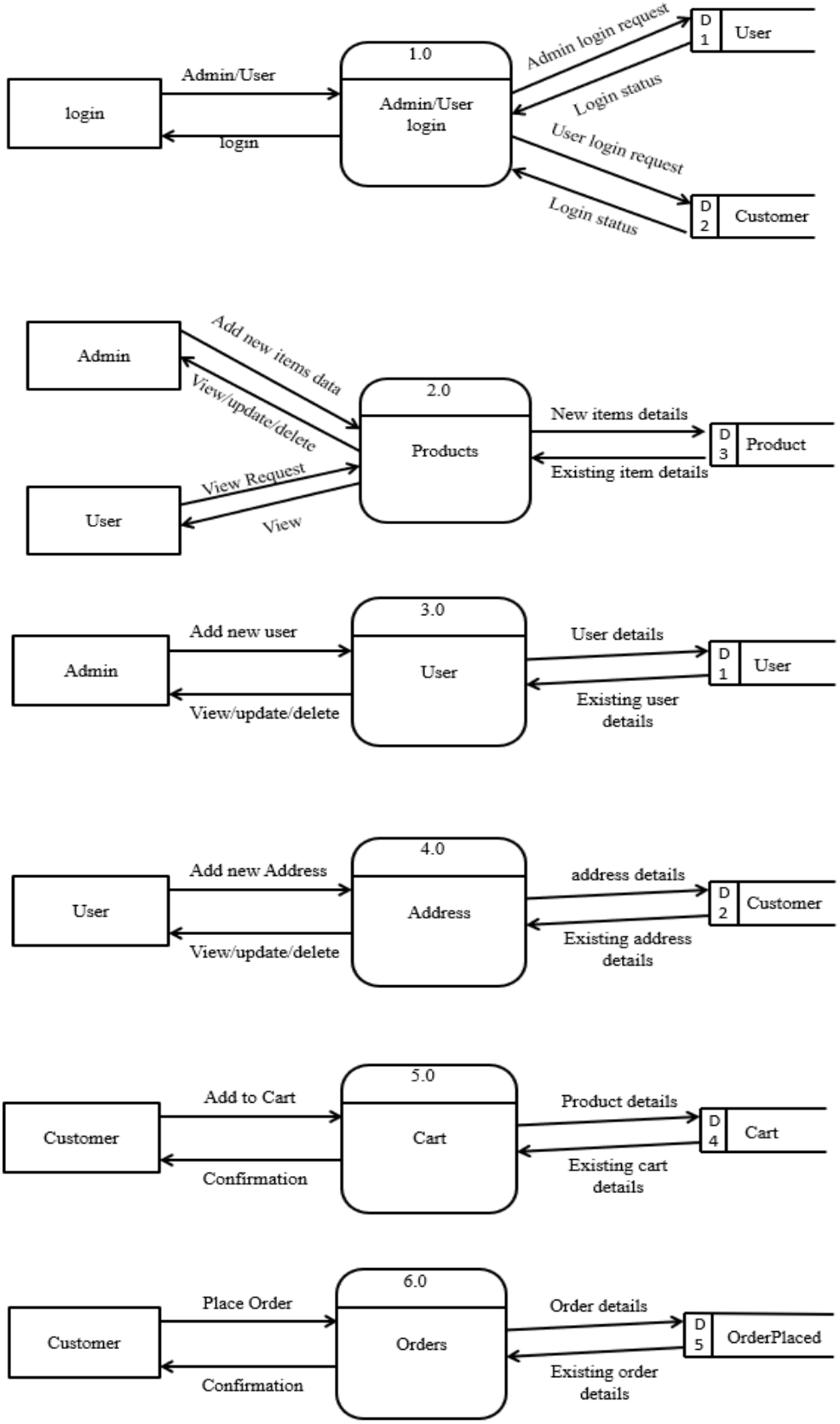
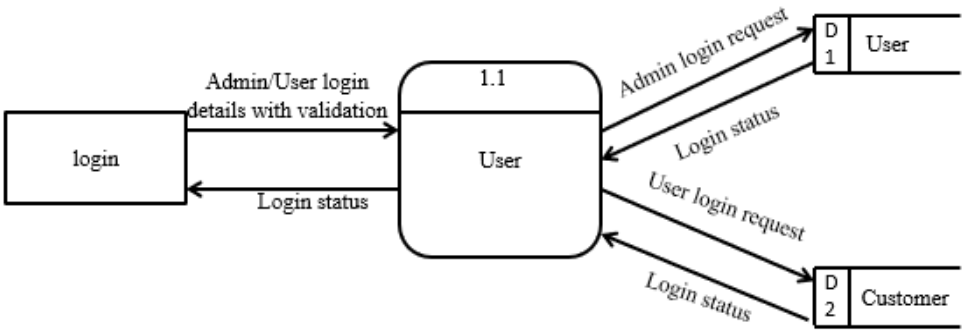
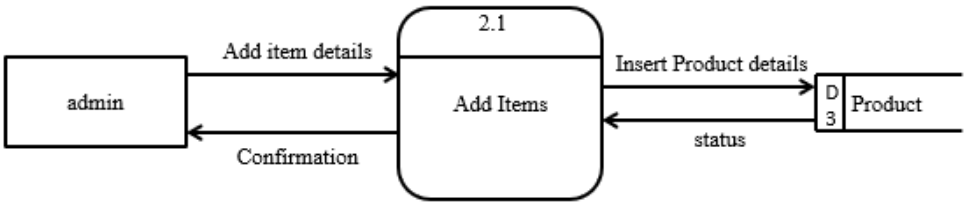
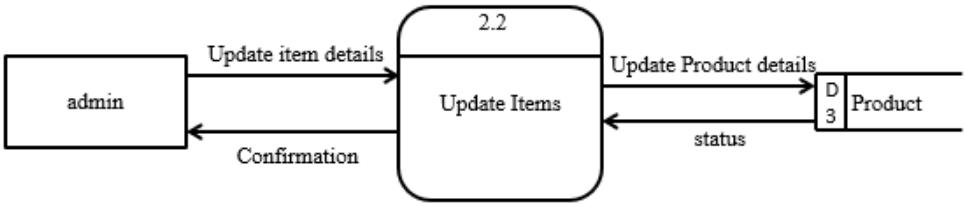


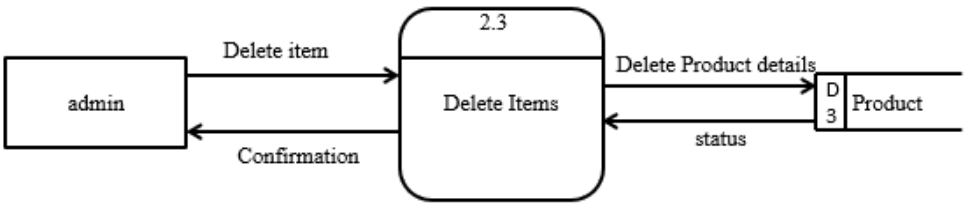
Figure 2.6.2 Leve1 1 DFD

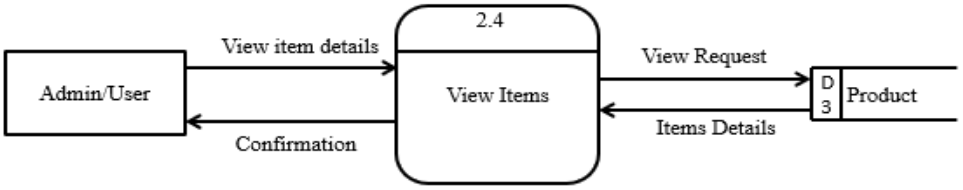
2.6.3. Level 2 DFD

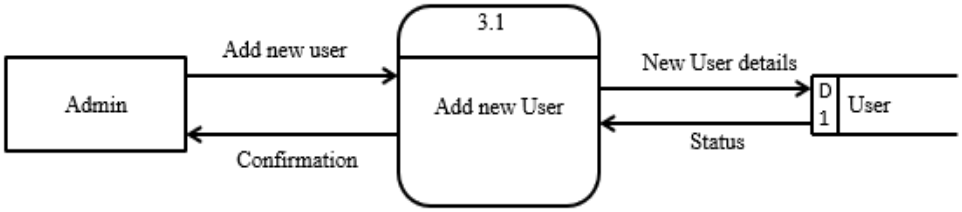


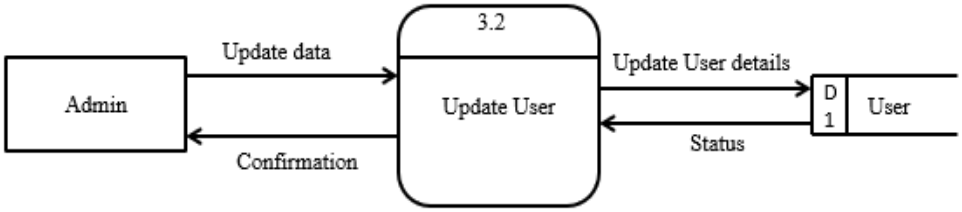


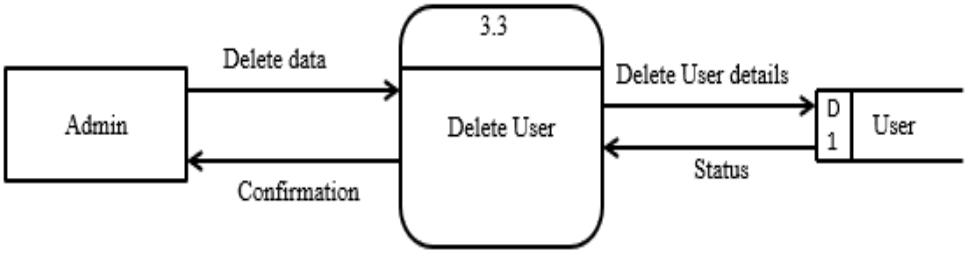


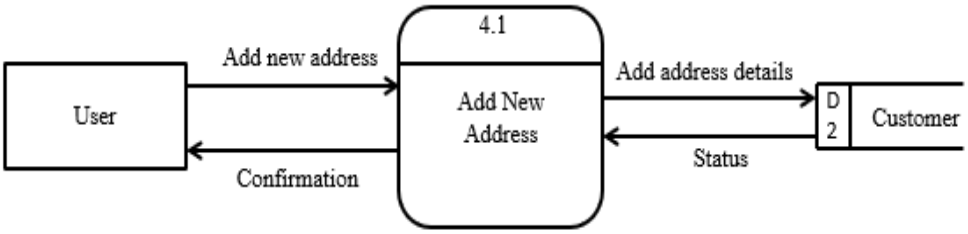


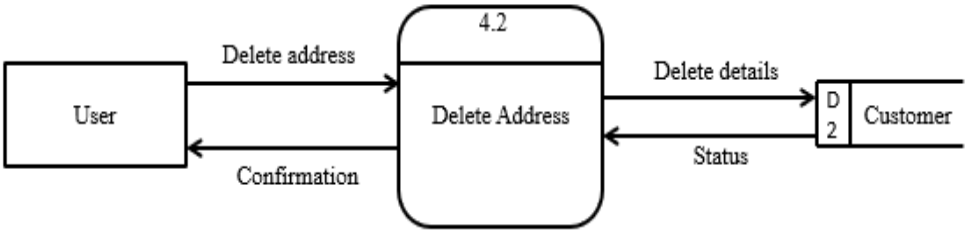



3.1
Admin
Add new user
Confirmation
Add new User
New User details
Status
D 1
User


3.2
Admin
Update data
Confirmation
Update User
Update User details
Status
D 1
User


3.3
Admin
Delete data
Confirmation
Delete User
Delete User details
Status
D 1
User


4.1
User
Add new address
Confirmation
Add New Address
Add address details
Status
D 2
Customer


4.2
User
Delete address
Confirmation
Delete Address
Delete details
Status
D 2
Customer

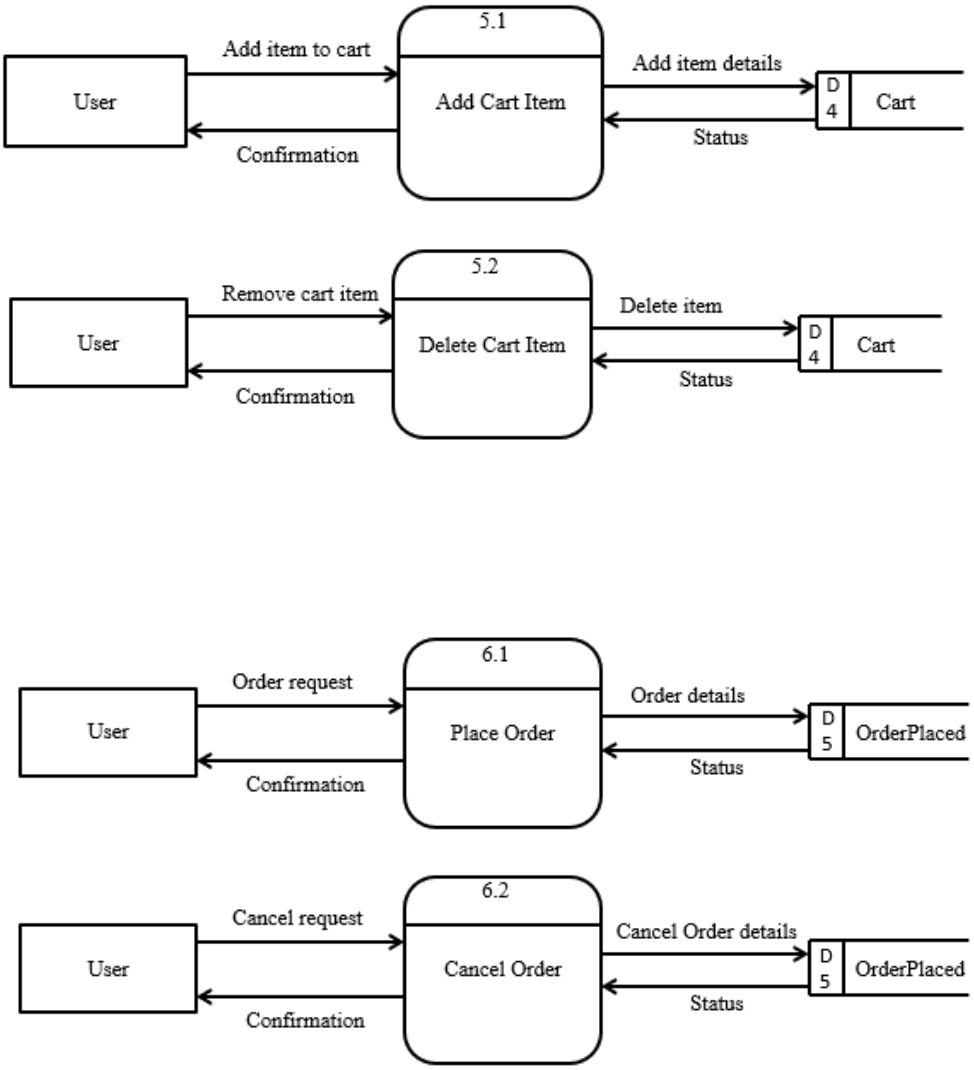


Figure 2.6.3 Level 2 DFD

2.7 JUSTIFICATION OF SELECTION OF TECHNOLOGY

With the emergence of new technologies, frameworks and tools, it has become an essential part for every business to build an online store or e-commerce platform and integrate it directly into the company website.

If we have ever thought of developing an e-commerce website, we will be juggling between choosing the right framework for our business. There are many frameworks available in the market which may convert our dream into reality, but on the other hand, we also have to check on their reliability, features, cost-effectiveness, and user adaptability.

While looking at the technologies and frameworks to build an e-commerce application, Python stands out to be a proficient and reliable programming language that could help we build a great e-commerce website. Python is the perfect solution because it's simple to understand and powerful enough to build AI systems.

Upon considering a Python for our e-commerce development choice, we will find numerous frameworks for that. Among all of them, the best-fit framework to build e-commerce applications for us is Django.

Every e-commerce website is dealt with buying and selling products along with secure payment integration. To gain users' trust and loyalty, you must develop a website which could secure the user's data and encourage them to become frequent visitors.

Python developers can integrate Oscar for seamlessly e-commerce website flow and process into various Django applications. They can even customize the open-source e-commerce framework according to day to day needs of each e-commerce app development by extending, replacing, or overriding specific classes. Also, the features of Oscar will help developers to build a custom e-commerce website with Python.

So, here are the factors that affect the application to go down in this competitive digital market.

- User experience
- Application loading time
- Security
- Various features
- Reliability

- Scalability

An e-commerce website needs some obligatory functionality to be present such as admin back-end, shopping cart, authorization, etc. Implementing them from scratch all by yourself can be a tedious and time-consuming task especially for fresher Django developers. But with the Django e-commerce packages, you can develop your e-commerce web app just like any experienced developer out there.

CHAPTER 3 : SYSTEM DESIGN

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, we need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

System Design focuses on how to accomplish the objective of the system.

It mainly focuses on:

- Systems
- Processes
- Technology

3.1 MODULES DIVISION

In this society helping system there are are three types of user one is **Admin, User** and the **visitor**.

Functionality of Admin :

Admin The administrator has all the rights to access the system. He is the one who has all rights to view the members and product details, modify those details. He can add various product based on the category. He can also set the available quantity of a product and its reasonable price. Also he can also set discount in various occasion. Admin can also view the details of a member. The admin have the power to generate the scratch card so that users can also use the recharge card to buy various product.

Functionality of User :

The user can log in to the system by using his specific email and password. User can view the products and order the products according to their own needs. He can view his profile and update his details. He can update his personal information by logging into the system. User can find various product by using search option easily. update his details. He can update his personal information by logging into the system. User can find various product by using search option easily.

3.2 DATABASE NORMALIZATION

Data normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. It was first proposed by Edgar F. codd as part of his relational model. Normalization entails organizing the columns (attributes) and tables (relations) of a database to ensure that their dependencies are properly enforced by database integrity constraints. It is accomplished by applying some formal rules either by a process of synthesis (creating a new database design) or decomposition (improving an existing database design).

3.2.1 (1NF) First Normal Form

user	
pk	id
	username
	password
	is_admin
	first_name
	address
	phone
	product_name
	product_description
	product_brand
	product_category
	product_image
	stock
	ordered_date
	order_status
	selling_price
	dicounted_price

Table 3.2.1 Level 1 Normalized Table

3.2.2 (2NF) Second Normal Form

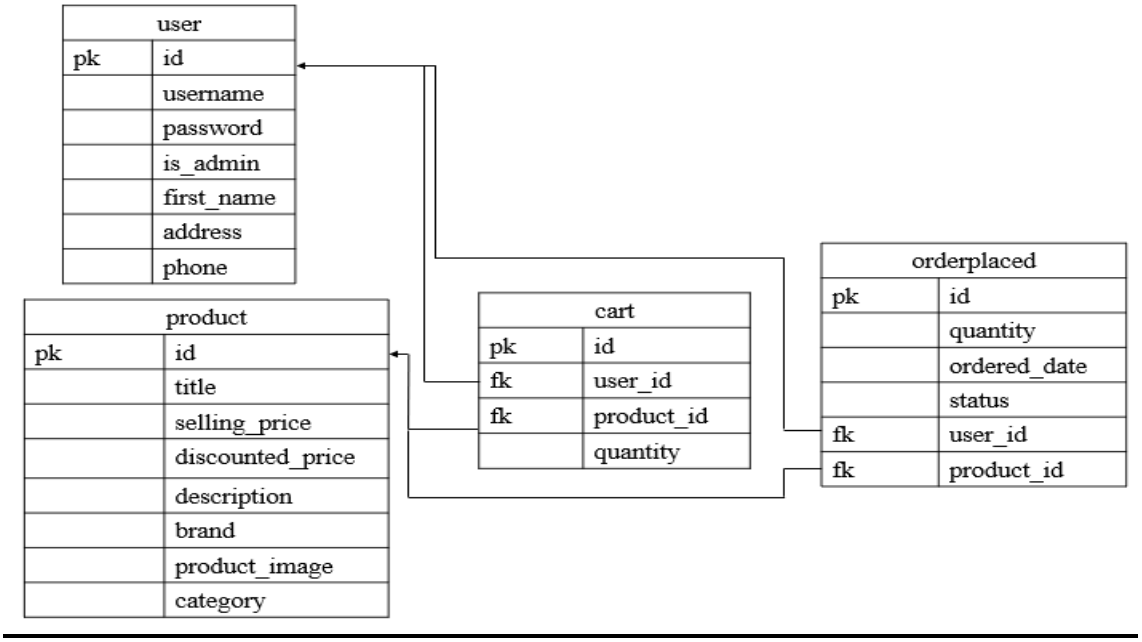


Table 3.2.2 Level 1 Normalized Table

3.2.2 (3NF) Third Normal Form

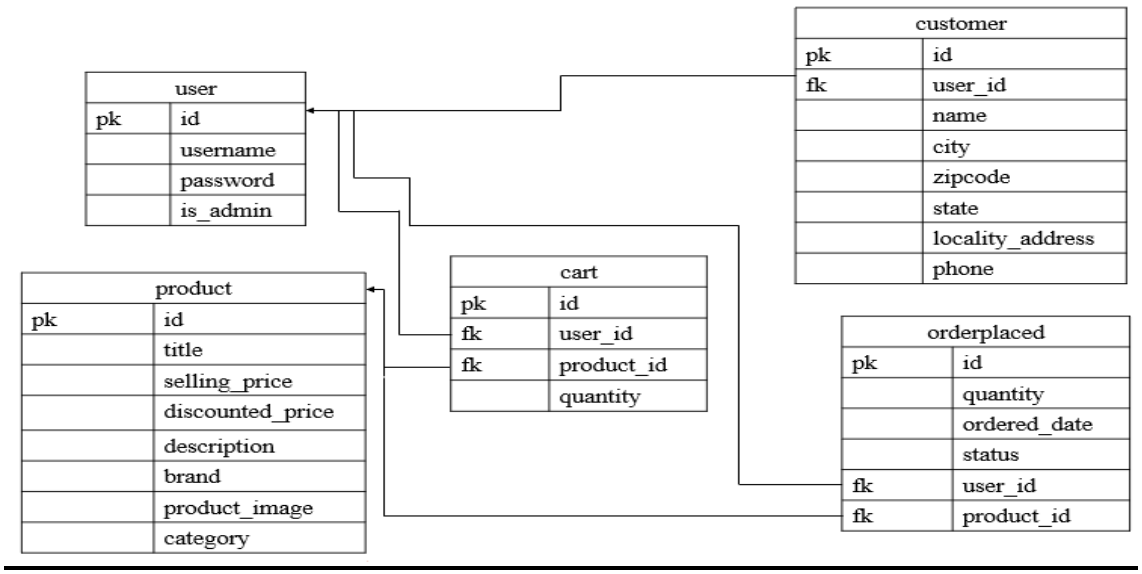


Table 3.2.3 Level 3 Normalized Table

3.3 DATA DICTIONARY

3.3.1 User Table

Variable		Data Type	Size	Description
id		INTEGER (PK)	-	To store user id for reference
first_name		VARCHAR	40	To store user first name
last_name		VARCHAR	40	To store user last name
username		VARCHAR	10	To store user username
password		VARCHAR	50	To store user password
last_login		DATETIME	-	To store user last login date and time
email		VARCHAR	100	To store user email address
date_joined		DATETIME	-	To store user registration date and time
is_active		BOOL	-	To store user login status

Table 3.3.2 User Table

3.3.2 Customer Table

Variable	Data Type	Size	Description
id	INTEGER (PK)	-	To store customer id for reference
user_id	USER (FK)	-	To store user instance
name	VARCHAR	50	To store customer name
phone	INTEGER	10	To store customer phone number
locality_address	VARCHAR	200	To store customer shipping address
city	VARCHAR	50	To store customer shipping city
state	VARCHAR	50	To store customer shipping state
zipcode	INTEGER	6	To store customer shipping postal code

Table 3.3.3 Customer Table

3.3.3 Product Table

Variable	Data Type	Size	Description
id	INTEGER (PK)	-	To store product id for reference
title	VARCHAR	100	To store product image
selling_price	REAL	-	To store product selling price
discounted_price	REAL	-	To store product discounted price
description	TEXT	1000	To store product description
brand	VARCHAR	20	To store product brand
category	VARCHAR	3	To store product category
product_image	VARCHAR	100	To store product image URL

Table 3.3.4 Product Table

3.3.4 Cart Table

Variable	Data Type	Size	Description
id	INTEGER (PK)	-	To store cart id for reference
quantity	INTEGER	-	To store cart product quantity
product_id	PRODUCT (FK)	-	To store product instance
user_id	USER (FK)	-	To store user instance

Table 3.3.5 Cart Table

3.3.5 OrderPlaced Table

Variable	Data Type	Size	Description
id	INTEGER (PK)	-	To store order id for reference
quantity	INTEGER	-	To store order quantity
order_date	DATETIME	-	To store order date and time
status	VARCHAR	50	To store order status
customer_id	CUSTOMER (FK)	-	To store customer instance
product_id	PRODUCT (FK)	-	To store product instance
user_id	USER (FK)	-	To store user instance
Txn_id	VARCHAR	20	To store transaction id of the purchase

Table 3.3.6 OrderPlaced Table

3.4 ER-DIAGRAM

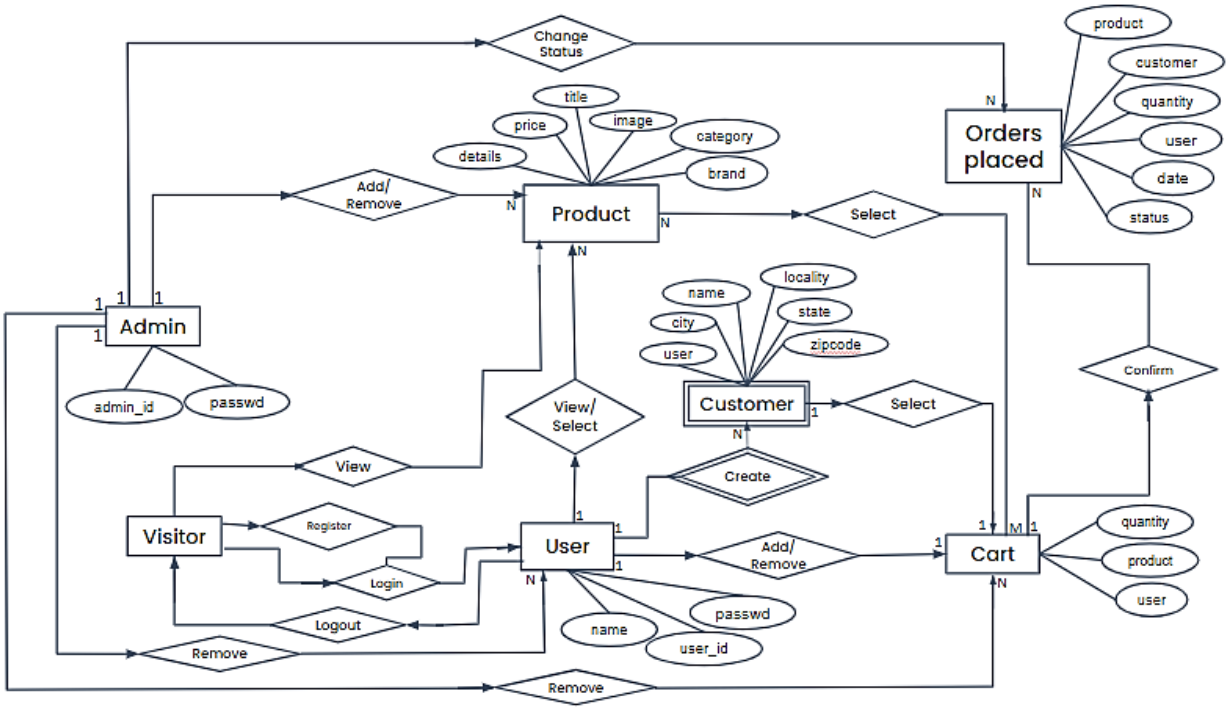


Figure 3.4.1 ER Diagram

3.5 GANTT CHART

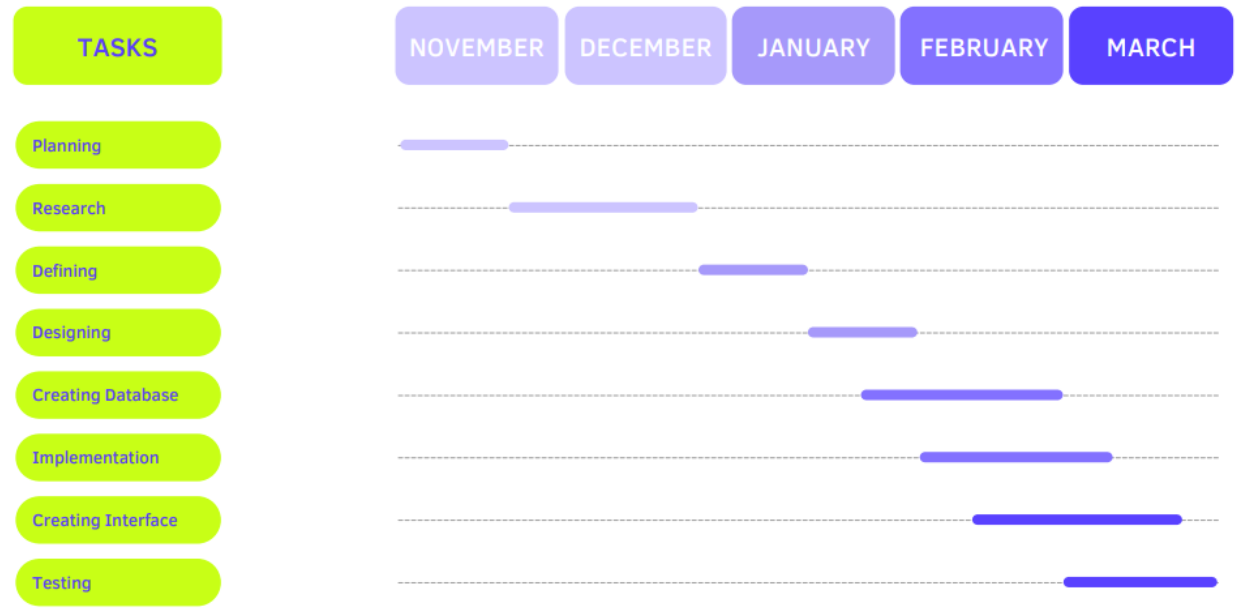


Figure 3.5.1 Gantt Chart

CHAPTER 4 : IMPLEMENTATION AND TESTING

4.1 CODING

4.1.1 Backend

Models.py

```
from django.contrib.auth.models import User
from django.db import models
from django.core.validators import MinLengthValidator, MaxLengthValidator
from django.forms import ValidationError
from django.utils.translation import gettext_lazy as _

STATE_CHOICES = (
    ('Andaman and Nicobar Islands', 'Andaman and Nicobar Islands'),
    ('Andhra Pradesh', 'Andhra Pradesh'),
    ('Arunachal Pradesh', 'Arunachal Pradesh'),
    ('Assam', 'Assam'),
    ('Bihar', 'Bihar'),
    ('Chandigarh', 'Chandigarh'),
    ('Chhattisgarh', 'Chhattisgarh'),
    ('Dadra and Nagar Haveli', 'Dadra and Nagar Haveli'),
    ('Daman and Diu', 'Daman and Diu'),
    ('Delhi', 'Delhi'),
    ('Goa', 'Goa'),
    ('Gujarat', 'Gujarat'),
    ('Haryana', 'Haryana'),
    ('Himachal Pradesh', 'Himachal Pradesh'),
    ('Jammu and Kashmir', 'Jammu and Kashmir'),
    ('Jharkhand', 'Jharkhand'),
```

```
    ('Karnataka', 'Karnataka'),
    ('Kerala', 'Kerala'),
    ('Ladakh', 'Ladakh'),
    ('Lakshadweep', 'Lakshadweep'),
    ('Madhya Pradesh', 'Madhya Pradesh'),
    ('Maharashtra', 'Maharashtra'),
    ('Manipur', 'Manipur'),
    ('Meghalaya', 'Meghalaya'),
    ('Mizoram', 'Mizoram'),
    ('Nagaland', 'Nagaland'),
    ('Odisha', 'Odisha'),
    ('Puducherry', 'Puducherry'),
    ('Punjab', 'Punjab'),
    ('Rajasthan', 'Rajasthan'),
    ('Sikkim', 'Sikkim'),
    ('Tamil Nadu', 'Tamil Nadu'),
    ('Telangana', 'Telangana'),
    ('Tripura', 'Tripura'),
    ('Uttar Pradesh', 'Uttar Pradesh'),
    ('Uttarakhand', 'Uttarakhand'),
    ('West Bengal', 'West Bengal')
)

CATEGORY_CHOICES = (
    ('RAM', 'RAM'),
    ('SSD', 'Solid State Drive'),
    ('HDD', 'Hard Disk Drive'),
    ('MB', 'Mother Board'),
    ('PSU', 'Power Supply Unit'),
    ('CB', 'Cabinet'),
    ('UPS', 'UPS'),
    ('PND', 'Pendrive'),
    ('MOU', 'Mouse'),
    ('KB', 'Keyboard'),
```

```
)

def validate_phone_length(value):
    str_value = str(value)
    if len(str_value)!=10:
        raise ValidationError(_('%(value)s is not invalid'))

def validate_zipcode_length(value):
    str_value = str(value)
    if len(str_value)!=6:
        raise ValidationError(_('%(value)s is not invalid'))

class Customer(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=200)
    phone = models.IntegerField(validators=[validate_phone_length])
    locality_address = models.CharField(max_length=200)
    city = models.CharField(max_length=50)
    state = models.CharField(choices=STATE_CHOICES, max_length=50)
    zipcode = models.IntegerField(validators=[validate_zipcode_length])

    def __str__(self):
        return str(self.id)

class Product(models.Model):
    title = models.CharField(max_length=100)
    selling_price = models.FloatField()
    discounted_price = models.FloatField()
    description = models.TextField()
    brand = models.CharField(max_length=100)
    category = models.CharField(choices=CATEGORY_CHOICES, max_length=3)
    product_image = models.ImageField(upload_to='productimg')

    def __str__(self):
```

```
        return str(self.id)

class Cart(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)

    def __str__(self):
        return str(self.id)
    @property
    def total_price(self):
        return self.quantity * self.product.discounted_price

STATUS_CHOICES = (
    ('Accepted', 'Accepted'),
    ('Packed', 'Packed'),
    ('On The Way', 'On The Way'),
    ('Delivered', 'Delivered'),
    ('Cancel', 'Cancel')
)

class OrderPlaced(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)
    ordered_date = models.DateTimeField(auto_now_add=True)
    status = models.CharField(choices=STATUS_CHOICES, max_length=50, default='Pending')
    txn_id = models.CharField(max_length=20)

    def __str__(self):
        return str(self.id)
    @property
    def total_cost(self):
```

```
        return self.quantity * self.product.discounted_price
```

admin.py

```
from django.contrib import admin
from django.utils.html import format_html
from django.urls import reverse
from .models import (Customer, Product, Cart, OrderPlaced)

@admin.register(Customer)
class CustomerModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'user', 'name', 'phone', 'locality_address', 'city', 'state', 'zipcode']

@admin.register(Product)
class ProductModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'title', 'selling_price', 'discounted_price', 'description', 'brand', 'category',
'product_image']

@admin.register(Cart)
class CartModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'user', 'product', 'product_info', 'quantity']

    def product_info(self, obj):
        link = reverse("admin:app_product_change", args=[obj.product.pk])
        return format_html('<a href="{}">{}</a>', link, obj.product.title)

@admin.register(OrderPlaced)
class OrderPlacedModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'user', 'customer', 'customer_info', 'product', 'product_info', 'quantity',
'ordered_date', 'status', 'txn_id']

    def customer_info(self, obj):
        link = reverse("admin:app_customer_change", args=[obj.customer.pk])
        return format_html('<a href="{}">{}</a>', link, obj.customer.name)
```

```
    def product_info(self, obj):
        link = reverse("admin:app_product_change", args=[obj.product.pk])
        return format_html('<a href="{}">{}</a>', link, obj.product.title)
```

Appys.py

```
from django.apps import AppConfig
class AppConfig(AppConfig):
    name = 'app'
```

context_processors.py

```
def cart_count(request):
  counter = 0
  from app.models import Cart
  try: counter = Cart.objects.filter(user=request.user).count()
  except: counter = 0
  return {'cart_counter': counter}
```

urls.py

```
from django.urls import path
from app import views
from django.conf import settings
from django.conf.urls.static import static
from django.contrib.auth import views as auth_views
from .forms import LoginForm, UserPasswordChangeForm, MyPasswordResetForm,
MySetPasswordForm

urlpatterns = [
    path('', views.ProductSneekPeak.as_view(), name='home'),
    # Basic urls
    path('product-detail/<int:pk>',
         views.ProductDetailView.as_view(), name='product-detail'),
```

```
path('ram/<slug:data>', views.ram, name='ramdata'),
path('ram/', views.ram, name='ram'),
path('solidstatedrive/<slug:data>', views.solidstatedrive, name='solidstatedrivedata'),
path('solidstatedrive/', views.solidstatedrive, name='solidstatedrive'),
path('harddiskdrive/<slug:data>', views.hdd, name='harddiskdrivedata'),
path('harddiskdrive/', views.hdd, name='harddiskdrive'),
path('motherboard/<slug:data>', views.motherboard, name='motherboarddata'),
path('motherboard/', views.motherboard, name='motherboard'),
path('keyboard/<slug:data>', views.keyboard, name='keyboarddata'),
path('keyboard/', views.keyboard, name='keyboard'),
path('psu/<slug:data>', views.psu, name='psudata'),
path('psu/', views.psu, name='psu'),
path('cabinet/<slug:data>', views.cabinet, name='cabinetdata'),
path('cabinet/', views.cabinet, name='cabinet'),
path('ups/<slug:data>', views.ups, name='upsdata'),
path('ups/', views.ups, name='ups'),
path('pendrive/<slug:data>', views.pendrive, name='pendrivedata'),
path('pendrive/', views.pendrive, name='pendrive'),
path('mouse/<slug:data>', views.mouse, name='mousedata'),
path('mouse/', views.mouse, name='mouse'),
path('address/', views.AddressView.as_view(), name='address'),
path('profile/', views.ProfileView.as_view(), name='profile'),
# Cart urls
path('add-to-cart/', views.add_to_cart, name='add-to-cart'),
path('cart/', views.view_cart, name='view_cart'),
path('pluscartitem/', views.plus_cart_item, name='pluscartitem'),
path('minuscartitem/', views.minus_cart_item, name='minuscartitem'),
path('removecartitem/', views.remove_cart_item, name='removecartitem'),
# checkout related
path('checkout/', views.checkout, name='checkout'),
path('paymentdone/', views.payment_done, name='paymentdone'),
path('buy/<int:pk>', views.buy_now, name='buy-now'),
path('buynowcheckout/', views.buynowcheckout, name='buynowcheckout'),
path('buynowpaymentdone/', views.buy_now_payment_done, name='buynowpaymentdone'),
```

```
    path('orders/', views.orders, name='orders'),
    # url for delete address record
    path('address/<int:id>', views.delete_customer, name='delete_customer'),

    # Auth
    path('registration/', views.CustomerRegistrationView.as_view(),
         name='customerregistration'),
    path('accounts/login/', auth_views.LoginView.as_view(template_name='app/login.html',
         authentication_form=LoginForm, next_page='/'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(next_page='login'), name='logout'),
    # Change password
    path('changepasswordsuccess/', auth_views.PasswordChangeDoneView.as_view(
        template_name='app/changepasswordsuccess.html'), name='changepasswordsuccess'),
    path('changepassword/',
auth_views.PasswordChangeView.as_view(template_name='app/changepassword.html',
         form_class=UserPasswordChangeForm, success_url='/changepasswordsuccess/'),
name='changepassword'),
    # Reset password
    path('resetpassword/',
auth_views.PasswordResetView.as_view(template_name='app/reset_password.html',
         form_class=MyPasswordResetForm), name="resetpassword"),
    path('resetpassword/done/', auth_views.PasswordResetDoneView.as_view(
        template_name='app/reset_password_done.html'), name="password_reset_done"),
    path('resetpasswordconfirm/<uidb64>/<token>/',
auth_views.PasswordResetConfirmView.as_view(
        template_name='app/reset_password_confirm.html', form_class=MySetPasswordForm),
name="password_reset_confirm"),
    path('resetpasswordsuccess', auth_views.PasswordResetCompleteView.as_view(
        template_name='app/reset_password_complete.html'), name="password_reset_complete"),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Views.py

```
from django.http import JsonResponse
from django.shortcuts import redirect, render
from django.views import View
from django.contrib import messages
from django.db.models import Q
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator
import os
from .models import Cart, Customer, OrderPlaced, Product, CATEGORY_CHOICES
from .forms import CustomerRegistrationForm, CustomerProfileForm
from .custom_logger import logger

########## Helper Functions #############
def calculateAmounts(cart):
    # Calculate total amounts based on cart objects
    if cart:
        amounts = []
        shipping = 50.0
        for i in cart:
            quantity = i.quantity
            price = i.product.discounted_price * quantity
            amounts.append(price)
        total_amt = sum(amounts)
        if total_amt >= 500: shipping = 0.0
        total_amt += shipping
        final_amounts = {
            'shippingamount': shipping, 'finalamount': total_amt, 'totalamount': sum(amounts),
        }
        return final_amounts
    else: return


###### Basic Page Renderers ########
```

```
@login_required
def orders(request):
    orders = OrderPlaced.objects.filter(
        user=request.user).order_by('-ordered_date')
    return render(request, 'app/orders.html', {'order_placed': orders})
@method_decorator(login_required, name='dispatch')
class ProfileView(View):
    # for profile page
    def get(self, request):
        return render(request, 'app/profile.html', {'active': 'btn-primary'})

class CustomerRegistrationView(View):
    # for register page
    def get(self, request):
        form = CustomerRegistrationForm()
        return render(request, 'app/customerregistration.html', {'form': form})
    def post(self, request):
        form = CustomerRegistrationForm(request.POST)
        if form.is_valid():
            messages.success(request, 'Account created Successfully!')
            form.save()
        return render(request, 'app/customerregistration.html', {'form': form})

class ProductSneekPeak(View):
    # for home page
    def get(self, request):
        # Fetch first 3 product objects according to their categories
        categories = {}
        for i in CATEGORY_CHOICES:
            cat_code = i[0]
            cat_name = ''.join(i[1].strip().lower().split())
            cat_product = Product.objects.filter(category=cat_code)[:5]
            categories[cat_name] = cat_product
        return render(request, 'app/home.html', categories)
```

```
class ProductDetailView(View):
    # for product page
    def get(self, request, pk):
        product = Product.objects.get(pk=pk)
        cart_state = False
        try: cart_state = Cart.objects.filter(Q(product=product.id) & Q(user=request.user)).exists()
        except: pass
        return render(request, 'app/productdetail.html', {'product': product, 'cart_state': cart_state})

######Category page renderers ########
def ram(request, data=None):
    # for ram page
    if data is not None:  data = " ".join(data.split("_"))
    if data == None:  rams = Product.objects.filter(category='RAM')
    elif str(data).lower() == 'corsair' or str(data).lower() == 'crucial':
        rams = Product.objects.filter(category='RAM').filter(brand=data)
    elif str(data) == 'below2000':
        rams = Product.objects.filter(
            category='RAM').filter(discounted_price__lt=2000)
    elif str(data) == 'above2000':
        rams = Product.objects.filter(
            category='RAM').filter(discounted_price__gt=2000)
    return render(request, 'app/categories/ram.html', {'rams': rams})

def solidstatedrive(request, data=None):
    # for ssd page
    if data is not None:  data = " ".join(data.split("_"))
    if data == None:  solidstatedrives = Product.objects.filter(category='SSD')
    elif str(data).lower() == 'samsung' or str(data).lower() == 'western digital':
        solidstatedrives = Product.objects.filter(category='SSD').filter(brand=data)
    elif str(data) == 'below4000':
        solidstatedrives = Product.objects.filter(category='SSD').filter(discounted_price__lt=4000)
    elif str(data) == 'above4000':
```

```
        solidstatedrives =
Product.objects.filter(category='SSD').filter(discounted_price__gt=4000)
    return render(request, 'app/categories/solidstatedrive.html', {'solidstatedrives':
solidstatedrives})

def cabinet(request, data=None):
    # for cabinet page
    if data is not None:data = " ".join(data.split("_"))
    if data == None:cabinets = Product.objects.filter(category='CB')
    elif str(data).lower() == 'samsung' or str(data).lower() == 'wd':
        cabinets = Product.objects.filter(category='CB').filter(brand=data)

    elif str(data) == 'below4000':
        cabinets = Product.objects.filter(category='CB').filter(discounted_price__lt=4000)

    elif str(data) == 'above4000':
        cabinets = Product.objects.filter(category='CB').filter(discounted_price__gt=4000)
    return render(request, 'app/categories/cabinet.html', {'cabinets': cabinets})

def pendrive(request, data=None):
    # for pendrive page
    if data is not None: data = " ".join(data.split("_"))
    if data == None:  pendrives = Product.objects.filter(category='PND')
    elif str(data).lower() == 'samsung' or str(data).lower() == 'wd':
        pendrives = Product.objects.filter(category='PND').filter(brand=data)
    elif str(data) == 'below4000':
        pendrives = Product.objects.filter(category='PND').filter(discounted_price__lt=4000)
    elif str(data) == 'above4000':
        pendrives = Product.objects.filter(category='PND').filter(discounted_price__gt=4000)
    return render(request, 'app/categories/pendrive.html', {'pendrives': pendrives})

def ups(request, data=None):
    # for ups page
    if data is not None:  data = " ".join(data.split("_"))
```

```
    if data == None:  upss = Product.objects.filter(category='UPS')
    elif str(data).lower() == 'samsung' or str(data).lower() == 'wd':
        upss = Product.objects.filter(category='UPS').filter(brand=data)
    elif str(data) == 'below4000':
        upss = Product.objects.filter(category='UPS').filter(discounted_price__lt=4000)
    elif str(data) == 'above4000':
        upss = Product.objects.filter(category='UPS').filter(discounted_price__gt=4000)
    return render(request, 'app/categories/ups.html', {'upss': upss})

def keyboard(request, data=None):
    # for keyboard page
    if data is not None: data = " ".join(data.split("_"))
    if data == None:  keyboards = Product.objects.filter(category='KB')
    elif str(data).lower() == 'samsung' or str(data).lower() == 'wd':
        keyboards = Product.objects.filter(category='KB').filter(brand=data)
    elif str(data) == 'below4000':
        keyboards = Product.objects.filter(  category='KB').filter(discounted_price__lt=4000)
    elif str(data) == 'above4000':
        keyboards = Product.objects.filter(  category='KB').filter(discounted_price__gt=4000)
    return render(request, 'app/categories/keyboard.html', {'keyboards': keyboards})

def hdd(request, data=None):
    # for hdd page
    if data is not None:  data = " ".join(data.split("_"))
    if data == None:  hdds = Product.objects.filter(category='HDD')
    elif str(data).lower() == 'samsung' or str(data).lower() == 'western digital':
        hdds = Product.objects.filter(category='HDD').filter(brand=data)
    elif str(data) == 'below4000':
        hdds = Product.objects.filter(category='HDD').filter(discounted_price__lt=4000)
    elif str(data) == 'above4000':
        hdds = Product.objects.filter(category='HDD').filter(discounted_price__gt=4000)
    return render(request, 'app/categories/hdd.html', {'harddiskdrives': hdds})

def psu(request, data=None):
```

```
    # for psu page
    if data is not None:  data = " ".join(data.split("_"))
    if data == None:   psus = Product.objects.filter(category='PSU')
    elif str(data).lower() == 'samsung' or str(data).lower() == 'wd':
        psus = Product.objects.filter(category='PSU').filter(brand=data)
    elif str(data) == 'below4000':
        psus = Product.objects.filter(category='PSU').filter(discounted_price__lt=4000)
    elif str(data) == 'above4000':
        psus = Product.objects.filter(category='PSU').filter(discounted_price__gt=4000)
    return render(request, 'app/categories/psu.html', {'psus': psus})

def motherboard(request, data=None):
    # for motherboard page
    if data is not None:data = " ".join(data.split("_"))
    if data == None:motherboards = Product.objects.filter(category='MB')
    elif str(data).lower() == 'samsung' or str(data).lower() == 'wd':
        motherboards = Product.objects.filter(category='MB').filter(brand=data)
    elif str(data) == 'below4000':
        motherboards = Product.objects.filter(category='MB').filter(discounted_price__lt=4000)
    elif str(data) == 'above4000':
        motherboards = Product.objects.filter(category='MB').filter(discounted_price__gt=4000)
    return render(request, 'app/categories/motherboard.html', {'motherboards': motherboards})

def mouse(request, data=None):
    # for mouse page
    if data is not None: data = " ".join(data.split("_"))
    if data == None:  mouses = Product.objects.filter(category='MOU')
    elif str(data).lower() == 'samsung' or str(data).lower() == 'wd':
        mouses = Product.objects.filter(category='MOU').filter(brand=data)
    elif str(data) == 'below4000':
        mouses = Product.objects.filter(category='MOU').filter(discounted_price__lt=4000)
    elif str(data) == 'above4000':
        mouses = Product.objects.filter(category='MOU').filter(discounted_price__gt=4000)
    return render(request, 'app/categories/mouse.html', {'mouses': mouses})
```

```
######## Cart Related ##############
@login_required
def add_to_cart(request):
    # for add to cart in db
    user = request.user
    product_id = request.GET.get("product_id")
    if product_id is not None:
        cart = Cart.objects.filter(user=user, product=product_id)
        if cart:
            logger.error("product already exists! Skipping addition to cart")
        else:
            # add product only if its not present
            product = Product.objects.get(id=product_id)
            Cart(user=user, product=product).save()
    return redirect("/cart")

@login_required
def view_cart(request):
    # for cart page
    if request.user.is_authenticated:
        cart = Cart.objects.filter(user=request.user)
        if cart:
            final_amounts = calculateAmounts(cart)
            return render(request, 'app/addtocart.html', {'carts': cart, 'amounts': final_amounts,
'cartempty': False})
        else:    return render(request, 'app/addtocart.html', {'cartempty': True})

@login_required
def plus_cart_item(request):
    # for plus button in cart page
    if request.method == 'GET':
        product_id = request.GET['product_id']
        cart_product = Cart.objects.get(Q(product=product_id) & Q(user=request.user))
```

```
        cart_product.quantity += 1
        cart_product.save()
        cart = Cart.objects.filter(user=request.user)
        if cart:
            data = calculateAmounts(cart)
            data['quantity'] = cart_product.quantity
            return JsonResponse(data)
        else:return JsonResponse({'empty': True})

@login_required
def minus_cart_item(request):
    # for minus button in cart page
    if request.method == 'GET':
        product_id = request.GET['product_id']
        cart_product = Cart.objects.get(Q(product=product_id) & Q(user=request.user))
        cart_product.quantity -= 1
        if cart_product.quantity < 1:
            logger.error("cart quantity was below 1, setting to 1")
            cart_product.quantity = 1
        cart_product.save()

        cart = Cart.objects.filter(user=request.user)
        if cart:
            data = calculateAmounts(cart)
            data['quantity'] = cart_product.quantity
            return JsonResponse(data)
        else:  return JsonResponse({'empty': True})


@login_required
def remove_cart_item(request):
    # for delete button in cart page
    if request.method == 'GET':
        product_id = request.GET['product_id']
```

```
        cart_product = Cart.objects.get(Q(product=product_id) & Q(user=request.user))
        cart_product.delete()

        cart = Cart.objects.filter(user=request.user)
        if cart:
            data = calculateAmounts(cart)
            data['quantity'] = cart_product.quantity
            return JsonResponse(data)
        else:  return JsonResponse({'empty': True})


@login_required
def checkout(request):
    user = request.user
    addresses = Customer.objects.filter(user=user)
    carts = Cart.objects.filter(user=user)
    final_amounts = calculateAmounts(carts)

    # get the PayPal Client ID from OS environment variables
    client_id = os.environ.get('PAYPAL-CLIENTID')

    # Calculating USD from INR for Payment
    inr_amount = final_amounts['finalamount']
    usd_amount = round(inr_amount/76.88, 2)
    logger.critical('Converting INR ' + str(inr_amount) +  ' to USD ' + str(usd_amount))
    return render(request, 'app/checkout.html', {'addresses': addresses, 'cartitems': carts,
'amounts': final_amounts, 'paypal_clientid': client_id, 'usd_amount': usd_amount})

@login_required
def payment_done(request):
    # called when order is placed through cart
    user = request.user
    custid = request.GET.get('custid')
    customer = Customer.objects.get(id=custid)
```

```
    cart = Cart.objects.filter(user=user)
    txn_id = request.GET.get('txn_id')
    for c in cart:
        OrderPlaced(user=user, customer=customer, product=c.product, quantity=c.quantity,
txn_id=txn_id).save()
        c.delete()
    return redirect('orders')

######### Buy Now Related ######
@login_required
def buy_now(request, pk):
    # Called when buy now button is clicked
    product = Product.objects.get(id=pk)
    quantity = 1
    total_amt = quantity * product.discounted_price
    shipping = 50 if total_amt < 500 else 0
    final_amt = total_amt + shipping

    final_amounts = {
        'shippingamount': shipping, 'finalamount': final_amt, 'totalamount': total_amt,
    }

    # get the PayPal Client ID from OS environment variables
    client_id = os.environ.get('PAYPAL-CLIENTID')

    # Calculating USD from INR for Payment
    usd_amount = round(final_amt/76.88, 2)
    logger.critical('Converting INR ' + str(final_amt) +  ' to USD ' + str(usd_amount))
    return render(request, 'app/buynow.html', {'amounts': final_amounts, 'paypal_clientid':
client_id, 'product': product, 'quantity': quantity, 'usd_amount': usd_amount})

@login_required
def buynowcheckout(request):
    # Called when order is placed from buy now page
```

```
    user = request.user
    product_id = request.POST.get('prod_id')
    quantity = int(request.POST.get('prod_quant'))
    product = Product.objects.get(id=product_id)
    addresses = Customer.objects.filter(user=user)

    total_amt = quantity * product.discounted_price
    shipping = 50 if total_amt < 500 else 0
    final_amt = total_amt + shipping

    final_amounts = {
        'shippingamount': shipping, 'finalamount': final_amt, 'totalamount': total_amt,
    }
    # get the PayPal Client ID from OS environment variables
    client_id = os.environ.get('PAYPAL-CLIENTID')

    # Calculating USD from INR for Payment
    usd_amount = round(final_amt/76.88, 2)
    logger.critical('Converting INR ' + str(final_amt) +' to USD ' + str(usd_amount))
    return render(request, 'app/buynowcheckout.html', {'addresses': addresses, 'paypal_clientid':
client_id, 'amounts': final_amounts, 'product': product, 'quantity': quantity, 'usd_amount':
usd_amount})

@login_required
def buy_now_payment_done(request):
    # called when payment has been done after buy now functionality
    user = request.user
    custid = request.GET.get('custid')
    product_id = request.GET.get('prod_id')
    quantity = request.GET.get('prod_quant')
    txn_id = request.GET.get('txn_id')

    customer = Customer.objects.get(id=custid)
    product = Product.objects.get(id=product_id)
```

```
    OrderPlaced(user=user, customer=customer, product=product, quantity=quantity,
txn_id=txn_id).save()
    return redirect('orders')


######### Address and Customer Related #######
@method_decorator(login_required, name='dispatch')
class AddressView(View):
    # for address page
    def get(self, request):
        form = CustomerProfileForm()
        address = Customer.objects.filter(user=request.user)
        return render(request, 'app/address.html', {'form': form, 'address': address, 'active': 'btn-
primary'})

    def post(self, request):
        form = CustomerProfileForm(request.POST)
        if form.is_valid():
            user = request.user
            name = form.cleaned_data['name']
            phone = form.cleaned_data['phone']
            locality_address = form.cleaned_data['locality_address']
            city = form.cleaned_data['city']
            state = form.cleaned_data['state']
            zipcode = form.cleaned_data['zipcode']
            reg = Customer(user=user, name=name, phone=phone,
                        locality_address=locality_address, city=city, state=state, zipcode=zipcode)
            reg.save()
        return redirect('address')

@login_required
def delete_customer(request, id):
    # for deleting customer address
```

```
    ob = Customer.objects.get(id=id)
    ob.delete()
    return redirect('address')
```

settings.py

```
from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = os.environ['SECRET_KEY']

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = False

ALLOWED_HOSTS = []

# Application definition

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

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

```
]

ROOT_URLCONF = 'ShopOnline.urls'

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',
                'app.context_processors.cart_count'
            ],
        },
    },
]

WSGI_APPLICATION = 'ShopOnline.wsgi.application'

DATABASES = {
    'default': {'ENGINE': 'django.db.backends.sqlite3', 'NAME': BASE_DIR / 'db.sqlite3'}
}

AUTH_PASSWORD_VALIDATORS = [
    { 'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator'},
    {'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator'},
    {'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator'},
    {'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator'},
]
```

```
LANGUAGE_CODE = 'en-us'

# TIME_ZONE = 'UTC'
TIME_ZONE = 'Asia/Kolkata'

USE_I18N = True
USE_TZ = True

STATIC_URL = 'static/'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'
LOGIN_REDIRECT_URL = '/profile/'
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
SESSION_ENGINE = "django.contrib.sessions.backends.db"
```

4.1.2 Frontend

Addtocart.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block title %}Cart{% endblock title %}
{% block main-content %}
<div class="container my-5"><div class="row">
  <h1 class="text-center mb-5">Shopping Cart</h1>
  <div class="col-sm-8"><div class="card">
   <div class="card-body"><h3>Cart</h3><div class="row">
    <div class="col-sm-3 text-center align-self-center"><img src="{% static
'app/images/product/1.jpg'%}" alt="" srcset="" class="img-fluid img-thumbnail shadow-sm"
height="150" width="150"> </div>
    <div class="col-sm-9"><div>
     <h5>Product 1</h5>
     <p class="mb-2 text-muted small">Description: Lorem ipsum dolor sit amet consectetur
adipisicing elit. Rerum quas, nihil vel velit sed, quos consequatur maiores odio libero eius est in
mollitia quo minus dolorum culpa consectetur, ipsam unde!</p>
     <div class="my-3">
      <label for="quantity">Quantity:</label>
      <a class="minus-cart btn"><i class="fas fa-minus-square fa-lg"></i></a>
       <span id="quantity">3</span>
      <a class="plus-cart btn"><i class="fas fa-plus-square fa-lg"></i></a>
     </div>
     <div class="d-flex justify-content-between">
      <a href="#" class="btn btn-sm btn-secondary mr-3">Remove item </a>
      <p class="mb-0"><span><strong>Rs. 110.00</strong></span></p>
     </div></div></div></div>
   <hr class="text-muted">
   <div class="row my-5">
    <div class="col-sm-3 text-center align-self-center"> <img src="{% static
'app/images/product/2.jpg'%}" alt="" srcset="" class="img-fluid img-thumbnail shadow-sm"
height="150" width="150"></div>
```

```
    <div class="col-sm-9"><div><h5>Product 2</h5>
      <p class="mb-2 text-muted small">Description: Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio corrupti repudiandae quas tenetur? Vel harum iste impedit ad cupiditate sint soluta consequuntur sed, enim, eligendi labore molestiae! Mollitia, ad exercitationem!</p>
      <div class="my-3">
       <label for="quantity">Quantity:</label>
       <a class="minus-cart btn"><i class="fas fa-minus-square fa-lg"></i></a>
         <span id="quantity">2</span>
       <a class="plus-cart btn"><i class="fas fa-plus-square fa-lg"></i></a>
      </div>
      <div class="d-flex justify-content-between">
       <a href="#" class="btn btn-sm btn-secondary mr-3">Remove item </a>
       <p class="mb-0"><span><strong>Rs. 320.00</strong></span></p></div>
   </div> </div></div></div></div></div>


  <div class="col-sm-4">
   <div class="card">
    <div class="card-body">
     <h3>The Total Amount of</h3>
     <ul class="list-group">
      <li class="list-group-item d-flex justify-content-between align-items-center border-0 px-0 pb-0">Amount<span>Rs. 430.00</span></li>
      <li class="list-group-item d-flex justify-content-between align-items-center px-0">Shipping<span>Rs. 70.00</span></li>
      <li class="list-group-item d-flex justify-content-between align-items-center border-0 px-0 mb-3">
       <div><strong>Total</strong> <small>(including VAT)</small></div>
        <span><strong>Rs. 500.00</strong></span> </li></ul>
     <div class="d-grid"><a href="{% url 'checkout' %}" class="btn btn-primary">Place Order</a></div>
   </div> </div></div></div></div>
<div class="container">
 <div class="row"><div class="col-sm-8"> <div class="card">
   <div class="card-body"><h5 class="mb-4">We accept</h5>
```

```
    <img src="{% static 'app/images/payment.png' %}" alt="" class="img-fluid">
  </div></div></div></div></div>
{% endblock main-content %}
```

Base.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en"><head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<link rel="shortcut icon" type="image/png" href="{% static 'favicon.ico' %}" />
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1"
crossorigin="anonymous" />

<link rel="preconnect" href="https://fonts.googleapis.com" />
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@100;300&display=swap"
rel="stylesheet">
<link rel="stylesheet" href="{% static 'app/css/all.min.css' %}" />
<link rel="stylesheet" href="{% static 'app/css/style.css' %}" />
<title>ShopOnline | {% block title %} {% endblock title %}</title>
</head><body>

  <div id="page-container"><div id="content-wrap">
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
<div class="container">
<a class="navbar-brand" href="/">ShopOnline</a>
<button class="navbar-toggler"
type="button" data-bs-toggle="collapse"
data-bs-target="#navbarSupportedContent"
aria-controls="navbarSupportedContent"
```

```
aria-expanded="false" aria-label="Toggle navigation" >
<span class="navbar-toggler-icon"></span> </button>
<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav me-auto mb-2 mb-lg-0">
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="/">Home</a></li>

<li class="nav-item dropdown">
<a
class="nav-link dropdown-toggle text-white"
href="#" id="electronicsDropdown"
role="button" data-bs-toggle="dropdown"
aria-expanded="false" >
Electronics </a><ul
class="dropdown-menu" aria-labelledby="electronicsDropdown">
<li><a class="dropdown-item" href="{% url 'mobile' %}">Mobile</a>
</li><li><a class="dropdown-item" href="#">Laptop</a></li>
</ul></li><li class="nav-item dropdown">
<a class="nav-link dropdown-toggle text-white"
href="#" id="fashionDropdown" role="button"
data-bs-toggle="dropdown" aria-expanded="false" >
Fashion </a><ul class="dropdown-menu" aria-labelledby="fashionDropdown">
<li><a class="dropdown-item" href="#">Top Wear</a></li>
<li><a class="dropdown-item" href="#">Bottom Wear</a></li>
</ul></li></ul><form class="d-flex">
<input class="form-control me-2" type="search"
placeholder="Search" aria-label="Search" />
<button class="btn btn-warning" type="submit">S</button>
</form><div>
<ul class="navbar-nav me-auto mb-2 mb-lg-0">
{% if request.user.is_authenticated %}
<li class="nav-item dropdown mx-2">
<a class="nav-link dropdown-toggle text-white"
href="#" id="profileDropdown"
```

```
role="button" data-bs-toggle="dropdown" aria-expanded="false" >
Sonam </a> <ul class="dropdown-menu" aria-labelledby="profileDropdown">
<li> <a class="dropdown-item" href="{% url 'profile' %}">Profile</a>
</li><li><a class="dropdown-item" href="{% url 'orders' %}">Orders</a>
</li><li><a
class="dropdown-item" href="{% url 'changepassword' %}" >Change Password</a>
</li><li><a class="dropdown-item" href="{% url 'logout' %}">Logout</a></li>
</ul></li><li class="nav-item mx-2">
<a href="{% url 'add-to-cart' %}"
class="nav-link text-white"><span class="badge bg-danger">4</span> Cart
</a> </li>

{% else %}
<li class="nav-item mx-2">
<a href="{% url 'login' %}" class="nav-link text-white">Login</a>
</li><li class="nav-item mx-2">
<a href="{% url 'customerregistration' %}"
class="nav-link text-white" >Registration</a>
</li>
{% endif %}
</ul></div></div></div></nav>
{% block banner_slider %}
{% endblock banner_slider %}
{% block livesale%}
{% endblock livesale %}
{% block main-content %}
{% endblock main-content %}
</div>
<footer class="container-fluid bg-dark text-center p-2 mt-5">
<small class="text-white" >Copyright &copy;
<script> document.write(new Date().getFullYear()); </script>
   Designed By Shivam Chaubey</small>
<img
src="{% static 'app/images/payment.png' %}"
```

```
alt="" srcset="" class="img-fluid" height="2px" />
</footer></div>

<script
src="https://code.jquery.com/jquery-3.5.1.min.js"
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/t7LECLbyPA2x65Kgf80OJFdroafW"
crossorigin="anonymous"></script>

<script src="{% static 'app/js/all.min.js' %}"></script>
</body></html>
```

changepassword.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block title %}Change Password{%endblock title %}
{% block main-content %}
<div class="container my-5">
  <div class="row">
    <h3>Welcome Sonam</h3>
    <div class="col-sm-2 border-end">
      <ul class="list-unstyled">
        <li class="d-grid">
          <a href="{% url 'profile' %}" class="btn btn-primary"
            >Change Password</a >  </li> </ul>
    </div>
    <div class="col-sm-9 offset-sm-1">
      <h3>Change Password</h3>
```

```
    <hr>
    <form action="" method="post" novalidate class="shadow-sm p-5">
      {% csrf_token %}
      {% for fm in form %}
        <div class="form-group">
          {{fm.label_tag}} {{fm}} <small class="text-danger">{{fm.errors|striptags}}</small>
        </div>
      {% endfor %}
      <input type="submit" class="btn btn-primary mt-4" value="Save">
      {% if form.non_field_errors %}
        {% for error in form.non_field_errors %}
          <p class="alert alert-danger my-3">{{error}}</p>
        {% endfor %}
      {% endif %}
    </form>
   </div>
  </div>
</div>
{% endblock main-content %}
```

customerregistration.html

```
{% extends 'app/base.html' %} {% load static %} {% block title %}Customer
Registration{% endblock title %} {% block main-content %}
<div class="container">
  <div class="row my-3">
    <div class="col-sm-6 offset-sm-3">
      <h3>Customer Registration</h3>
      <hr />

      <form action="" method="post" novalidate class="shadow p-5">
      {% if messages %}
        {% for message in messages %}
```

```
          <p {% if message.tags %} class="alert alert-{{message.tags}} mb-5" {% endif
%}>{{message}}</p>
        {% endfor %}
      {% else %}
      {% endif %}
      {% csrf_token %}
        {% for fm in form %}

        <div class="form-group mb-3">
          {{fm.label_tag}} {{fm}}
          <small class="text-danger">{{fm.errors|striptags}}</small>
        </div>
        {% endfor %}

        <input type="submit" value="Submit" class="btn btn-primary" />

        <br />
        <div class="text-center text-primary fw-bold">
          <small
            >Existing User ?
            <a href="{% url 'login' %}" class="text-danger">Login Now</a>
          </small>
        </div>

        {% if form.non_field_errors %} {% for error in form.non_field_errors %}
        <p class="alert alert-danger my-3">{{error}}</p>
        {% endfor %} {% endif %}
      </form>
    </div>
  </div>
</div>
{% endblock main-content %}
```

Home.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block title %}Home{% endblock title %}
{% block banner_slider %}
<div id="carouselExampleControls" class="carousel slide" data-bs-ride="carousel">
   <div class="carousel-inner">
      <div class="carousel-item active">
         <img src="{% static 'app/images/banner/b1.jpg' %}" class="d-block w-100" alt="...">
      </div>
      <div class="carousel-item">
         <img src="{% static 'app/images/banner/b2.jpg' %}" class="d-block w-100" alt="...">
      </div>
      <div class="carousel-item">
         <img src="{% static 'app/images/banner/b3.jpg' %}" class="d-block w-100" alt="...">
      </div>
      <div class="carousel-item">
         <img src="{% static 'app/images/banner/b4.jpg' %}" class="d-block w-100" alt="...">
      </div>
   </div>
   <a class="carousel-control-prev" href="#carouselExampleControls" role="button" data-bs-
slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Previous</span>
   </a>
   <a class="carousel-control-next" href="#carouselExampleControls" role="button" data-bs-
slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Next</span>
   </a>
</div>
{% endblock banner_slider %}

{% block livesale %}
```

```
<div class="container">
   <div class="row bg-danger text-center p-5 text-white border-bottom shadow">
      <h1>SALE IS LIVE NOW</h1>
      <span>5% Instant Discount on Axis Bank Credit and Debit Card</span>
      <small class="fw-lighter">Term and Condition Applied (For details visit Bank's official
Website)</small>
   </div>
</div>
{% endblock livesale %}

{% block main-content %}
<!-- Start Category: Mobiles -->
<div class="m-3">
   <h2>Mobiles</h2>
   <div class="owl-carousel" id="slider1">
      {% for b in mobiles %}
      <a href="{% url 'product-detail' b.id %}" class="btn">
         <div class="item"><img src="{{b.product_image.url}}" alt="" height="200px"><span
               class="fw-bold">{{b.title}}</span><br><span class="fs-5">₹
{{b.discounted_price}}</span></div>
      </a>
      {% endfor %}
   </div>
</div>
<div class="container my-5">
   <div class="row">
      <div class="col-sm-3">
         <div class="card mb-3">
            <div class="card-body">
               <img src="{% static 'app/images/payavail/cc.jpg' %}" alt="" srcset="" class="img-
fluid">
            </div>
         </div>
      </div>
```

```
        <div class="col-sm-3">
          <div class="card mb-3">
            <div class="card-body">
              <img src="{% static 'app/images/payavail/upi.jpg' %}" alt="" srcset="" class="img-fluid">
            </div>
          </div>
        </div>
        <div class="col-sm-3">
          <div class="card mb-3">
            <div class="card-body">
              <img src="{% static 'app/images/payavail/nb.jpg' %}" alt="" srcset="" class="img-fluid">
            </div>
          </div>
        </div>
        <div class="col-sm-3">
          <div class="card mb-3">
            <div class="card-body">
              <img src="{% static 'app/images/payavail/bj.jpg' %}" alt="" srcset="" class="img-fluid">
            </div>
          </div>
        </div>
      </div>
    </div>
    {% endblock main-content %}
```

Login.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block title %}Login{% endblock title %}
{% block main-content %}
```

```
<div class="container">
  <div class="row my-3">
    <div class="col-sm-6 offset-sm-3">
      <h3>Login</h3>
      <hr />
      <form action="" method="post" novalidate class="shadow p-5">
        {% csrf_token %}
        {% for fm in form %}
          <div class="form-group">
            {{fm.label_tag}} {{fm}}
            <small class="text-danger">{{fm.errors|striptags}}</small><br/>
          </div>
        {% endfor %}
        {% comment %} <small><a href="{% url 'password_reset' %}">Forgot Password
?</a></small> <br /> {% endcomment %}
        <input type="submit" class="btn btn-primary mt-4" value="Login" />
        <br />
        <div class="text-center text-primary fw-bold">
          <small>New to ShopOnline ?
            <a href="{% url 'customerregistration' %}" class="text-danger">Create an Account</a>
          </small></div>
        {% if form.non_field_errors %}
          {% for error in form.non_field_errors %}
            <p class="alert alert-danger my-3">{{error}}</p>
          {% endfor %}
        {% endif %}
      </form></div></div></div>
{% endblock main-content %}
```

Orders.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block custom_css %}
```

```
  <!-- Custom CSS -->
{% endblock custom_css %}
{% block title %}Orders{% endblock title %}

{% block main-content %}
  <div class="container">
    <div class="mt-5">
      <div class="col-sm-12">
        <h4>Orders</h4>
        <hr>
          <div class="my-2 row pt-2">
            {% for order in order_placed %}
              <div class="col-sm-2">
                <img src="{{order.product.product_image.url}}" alt="" class="img-fluid"
height="150" width="150">
              </div><div class="col-sm-7"><h6 class="fw-light">
            <a href="{% url 'product-detail' order.product.id %}">{{order.product.title}}</a></h6>
                <span><span class="fw-bold">Quantity:</span> {{order.quantity}}</span><br>
                <span><span class="fw-bold">Order Date:</span>
{{order.ordered_date}}</span><br>
                <span><span class="fw-bold">Ordered By:</span>
{{order.customer.name}}</span><br>
                <span><span class="fw-bold">Status:</span> {{order.status}}</span><br>
                <span><span class="fw-bold">Price:</span> {{order.total_cost}}</span><br>
                <span><span class="fw-bold">Txn ID:</span> {{order.txn_id}}</span><br>
              </div>
              <div class="col-sm-3 pt-5">
                <p>Order Status: <span class="fw-bold">{{order.status}}</span></p>
                <div class="progress">
                  {% if order.status == 'Pending' %}
                    <div style="width: 1%;" class="progress-bar bg-secondary" role="progressbar"
aria-valuenow="5" aria-valuemin="0" aria-valuemax="100"></div>
                  {% endif %}
                  {% if order.status == 'Accepted' %}
```

```
                <div style="width: 10%;" class="progress-bar" role="progressbar" aria-valuenow="10" aria-valuemin="0" aria-valuemax="100"></div>
                {% endif %}
                {% if order.status == 'Packed' %}
                <div style="width: 30%;" class="progress-bar bg-info" role="progressbar" aria-valuenow="30" aria-valuemin="0" aria-valuemax="100"></div>
                {% endif %}

                {% if order.status == 'On The Way' %}
                <div style="width: 70%;" class="progress-bar bg-warning" role="progressbar" aria-valuenow="70" aria-valuemin="0" aria-valuemax="100"></div>
                {% endif %}

                {% if order.status == 'Delivered' %}
                <div style="width: 100%;" class="progress-bar bg-success" role="progressbar" aria-valuenow="10" aria-valuemin="0" aria-valuemax="100"></div>
                {% endif %}

                {% if order.status == 'Cancel' %}
                <div style="width: 100%;" class="progress-bar bg-danger" role="progressbar" aria-valuenow="10" aria-valuemin="0" aria-valuemax="100"></div>
                {% endif %}
              </div></div><hr class="my-5">
          {% endfor %}
        </div></div></div></div>
{% endblock main-content %}
```

Productdetail.html

```
{% extends 'app/base.html' %} {% load static %} {% block title %}{{product.title}}{% endblock title %} {% block main-content %}
<style>
   pre{
      font-family: 'Roboto', sans-serif;
```

```
    }
</style>
<div class="container my-5">
  <div class="row">
    <div class="col-sm-6 text-center align-self-center">
      <img
        src="{{product.product_image.url}}"
        alt=""
        class="img-fluid img-thumbnail"
      />
    </div>
    <div class="col-sm-5 offset-sm-1">
      <h2>{{product.title}}</h2>
      <hr /><p><pre>{{product.description}}</pre></p><br /><h4>
        ₹ {{product.discounted_price}}
        <small class="fw-light text-decoration-line-through"
          >{{product.selling_price}}</small>
      </h4><br />
      <a href="{% url 'add-to-cart' %}" class="btn btn-primary shadow px-5 py-2"
        >Add to Cart</a>
      <a href="{% url 'buy-now' %}" class="btn btn-danger shadow px-5 py-2 ms-4"
        >Buy Now</a>
      <h5 class="mt-5">Available Offers</h5>
      <ul>
        <li>Bank Offer 5% Unlimited Cashback on Flipkart Axis Bank Credit</li>
        <li>Special Price Get extra ₹3000 off (price inclusive of discount)</li>
        <li>No cost EMI ₹1,667/month. Standard EMI also available</li>
        <li>
          Partner Offer ₹2000 Flipkart Gift Card on Every 1000th Transaction
          with a new Visa Debit/Credit Card
        </li></ul></div></div></div>
{% endblock main-content %}
```

Profile.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block title %}Profile{% endblock title %}
{% block main-content %}
<div class="container my-5">
 <div class="row">
  <h3>Welcome Sonam</h3>
  <div class="col-sm-2 border-end">
   <ul class="list-unstyled">
    <li class="d-grid"><a href="{% url 'profile' %}" class="btn btn-primary">Profile</a></li>
    <li class="d-grid"><a href="{% url 'address' %}" class="btn">Address</a></li>
   </ul>
  </div>
  <div class="col-sm-8 offset-sm-1">
   <form action="" method="post">
    {% csrf_token %}
    <div class="col-12">
      <label for="inputName" class="form-label">Name</label>
      <input type="text" class="form-control" id="inputEmail">
    </div>
    <div class="col-12">
      <label for="inputAddress" class="form-label">Address</label>
      <input type="text" class="form-control" id="inputAddress" placeholder="1234 Main St">
    </div>
    <div class="col-12">
      <label for="inputAddress2" class="form-label">Address 2</label>
      <input type="text" class="form-control" id="inputAddress2" placeholder="Apartment,
studio, or floor">
    </div>
    <div class="col-12">
      <label for="inputCity" class="form-label">City</label>
      <input type="text" class="form-control" id="inputCity">
    </div>
```

```
    <div class="col-md-4">
      <label for="inputState" class="form-label">State</label>
      <select id="inputState" class="form-select">
        <option selected>Choose...</option>
        <option>...</option>
      </select>
    </div>
    <div class="col-md-2">
      <label for="inputZip" class="form-label">Zip</label>
      <input type="text" class="form-control" id="inputZip">
    </div>
    <div class="col-12 mt-3">
      <button type="submit" class="btn btn-primary">Submit</button>
    </div></form> </div></div></div>
{% endblock main-content %}
```

Ram.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block title %} RAM {% endblock title %} {% block main-content %}
<div class="container my-5">
    <div class="row">
        <div class="col-sm-3">
            <div class="list-group">
                <a href="{% url 'ram' %}" class='list-group-item list-group-item-action' aria-current='true'>All RAMs</a>

                <a href="{% url 'ramdata' 'CORSAIR' %}" class='list-group-item list-group-item-action' aria-current='true'>Corsair</a>

                <a href="{% url 'ramdata' 'CRUCIAL' %}" class='list-group-item list-group-item-action' aria-current='true'>Crucial</a>
```

```
            <a href="{% url 'ramdata' 'below2000' %}" class='list-group-item list-group-item-action' aria-current='true'>Below 2000</a>

            <a href="{% url 'ramdata' 'above2000' %}" class='list-group-item list-group-item-action' aria-current='true'>Above 2000</a>
        </div>
    </div>

    <div class="col-sm-8">
        <div class='row'>
            {% for product in rams %}
                <div class='col-sm-4 text-center mb-4'>
                    <a href="{% url 'product-detail' product.id %}" class='btn'>
                        <div class="item">
                            <img src="{{product.product_image.url}}" height="300">
                            <div class="fw-bold">{{product.title}}</div>
                            <div class="fw-bold">₹ {{product.discounted_price}}</div>
                            <div class="fw-light text-decoration-line-through">₹ {{product.selling_price}}</div>
                        </div></a></div>
            {% endfor %}
        </div>  </div></div></div>
{% endblock main-content %}
```

cabinet.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block custom_css %}
  <!-- Custom CSS -->
  <link rel="stylesheet" href="{% static 'app/css/category.css' %}" />
{% endblock custom_css %}
{% block title %} Cabinet {% endblock title %}
{% block main-content %}
```

```
  <!-- Start Main Container -->
  <div class="main-container">
    <div class="container section">
      <div class="row">

        <!-- Start Sidebar filter -->
        <div class="col-sm-2">
          <div class="list-group">
            <a href="{% url 'cabinet' %}" class='list-group-item list-group-item-action' aria-current='true'>All Cabinets</a>

            <a href="{% url 'cabinetdata' 'SAMSUNG' %}" class='list-group-item list-group-item-action' aria-current='true'>Samsung</a>

            <a href="{% url 'cabinetdata' 'WD' %}" class='list-group-item list-group-item-action' aria-current='true'>Western Digital</a>

            <a href="{% url 'cabinetdata' 'below4000' %}" class='list-group-item list-group-item-action' aria-current='true'>Below 4000</a>

            <a href="{% url 'cabinetdata' 'above4000' %}" class='list-group-item list-group-item-action' aria-current='true'>Above 4000</a>
          </div>
        </div>
        <!-- End Sidebar filter -->

        <!-- Start Product list -->
        <div class="col-sm-10">
          <div class='row'>
            {% for product in cabinets %}
              <div class="product-card col">
                <a href="{% url 'product-detail' product.id %}" class="product-card-btn">
                  <div class="item">
                    <div class="item-img">
```

```
                <img src="{{product.product_image.url}}" alt="" />
              </div>
              <div class="fw-bold title">
                {{product.title}}
              </div>
              <div class="fs-5">₹ {{product.discounted_price}}</div>
              <div class="fs-10 fw-light text-decoration-line-through">
                ₹ {{product.selling_price}}
              </div>
              <div class="btn-container">
                <a href="#" type="button" class="btn btn-primary shadow-none">
                  <span>Add to Cart</span>  
                  <i class="fas fa-shopping-cart"></i>
                </a> </div> </div>  </a>  </div>
          {% endfor %}
        </div> </div>
      <!-- End Product list -->
    </div>  </div>  </div>
  <!-- End Main Container -->
{% endblock main-content %}
```

Buynow.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block custom_css %}
  <!-- Custom CSS -->
  <link rel="stylesheet" href="{% static 'app/css/cart.css' %}" />
{% endblock custom_css %}
{% block title %}Buy{% endblock title %}

{% block main-content %}
  <div class="container">
    <div class="row mt-5">
```

```
    <h4 class="text-center mb-5">Order Details</h4>
    <div class="col-sm-8">
      <div class="row">
        <div class="col-sm-3 text-center align-self-center">
          <a href="{% url 'product-detail' product.id %}">
            <img src="{{product.product_image.url}}" alt="" srcset="" class="img-fluid img-
thumbnail border-0" height="150" width="150">
          </a></div>
        <div class="col-sm-9">
          <div>
            <h5>{{product.title}}</h5>
            <p class="mb-2 text-muted small">
              Description: <pre>{{product.description}}</pre>
            </p>
            <div class="my-3">
              <label for="quantity">Quantity:</label>

              <button class="minus-item btn shadow-none">
                <i class="fas fa-minus-square fa-lg"></i>
              </button>

              <span id="quantity">{{quantity}}</span>

              <button class="plus-item btn shadow-none">
                <i class="fas fa-plus-square fa-lg"></i>
              </button></div>
            <div class="d-flex justify-content-between">

              <p class="mb-0">
                <span><strong>₹ <span id="product-
price">{{product.discounted_price}}</span></strong></span>
              </p></div></div></div></div></div>
    <div class="col-sm-4">
      <div class="card">
```

```
        <div class="card-body">
          <h3>The Total Amount of</h3>
          <ul class="list-group">
            <li class="list-group-item d-flex justify-content-between align-items-center border-0
px-0 pb-0">
              Amount<span>Rs. <span id="total-
amount">{{amounts.totalamount}}</span></span>
            </li>
            <li class="list-group-item d-flex justify-content-between align-items-center px-0
border-0">
              Shipping<span>Rs. <span id="shipping-
amount">{{amounts.shippingamount}}</span></span>
            </li>
            <li class="list-group-item d-flex justify-content-between align-items-center border-0
px-0 mb-3">
              <div>
                <strong>Total</strong> <small>(including VAT)</small>
              </div>
              <span><strong>Rs. <span id="final-
amount">{{amounts.finalamount}}</span></strong>
            </li>
          </ul>
          <form action="/buynowcheckout/" method="POST" id="checkout-form">
            {% csrf_token %}

            <input name="prod_id" id="product_id" type="text" class="d-none bg-warning"
value="{{product.id}}">
            <input name="prod_quant" id="prod_quant" type="text" class="d-none bg-warning"
value="{{quantity}}">

            <div class="d-grid">
              <button type="submit" class="btn btn-primary">Place Order</button>
            </div>
          </form>
```

```
        </div></div></div></div></div>
{% endblock main-content %}
{% block custom_js %}
  <script>
    var quantInput = document.getElementById("prod_quant");
    var minusBtn = document.getElementsByClassName("minus-item")[0];
    var plusBtn = document.getElementsByClassName("plus-item")[0];
    var quantDisplay = document.getElementById("quantity");

    minusBtn.addEventListener("click", ()=> {
      var quantity = parseInt(quantDisplay.innerText);
      quantity -= 1;
      if (quantity<1) quantity = 1;
      updateQuantity(quantity);
    });

    plusBtn.addEventListener("click", ()=> {
      var quantity = parseInt(quantDisplay.innerText);
      quantity += 1;
      if (quantity>10) quantity = 10;
      updateQuantity(quantity);
    });

    function updateQuantity(quantity) {
      quantDisplay.innerText = quantity.toString();
      quantInput.value = quantity.toString();

      shipAmtEl = document.getElementById("shipping-amount");
      totalAmtEl = document.getElementById("total-amount");
      finalAmtEl = document.getElementById("final-amount");
      priceEl = document.getElementById("product-price");

      shipAmt = parseFloat(shipAmtEl.innerText);
```

```
    totalAmt = 0.0;
    finalAmt = 0.0;
    price = parseFloat(document.getElementById("product-price").innerText);

    totalAmt = price * quantity
    if (totalAmt >= 500) shipAmt = 0.0;
    finalAmt = totalAmt + shipAmt;
    shipAmtEl.innerText = shipAmt;
    totalAmtEl.innerText = totalAmt;
    finalAmtEl.innerText = finalAmt;
   }
  </script>
{% endblock custom_js %}
```

Buynowcheckout.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block custom_css %}
  <!-- Custom CSS -->
  <link rel="stylesheet" href="{% static 'app/css/cart.css' %}" />
{% endblock custom_css %}
{% block title %}Checkout{% endblock title %}
{% block main-content %}
  <div class="container">
    <div class="row mt-5">
      <div class="col-sm-6">
        <h4>Order Summary</h4>
        <hr>
          <div class="card mb-2">
            <div class="row">
              <div class="col-sm-3 text-center align-self-center">
                <a href="{% url 'product-detail' product.id %}">
```

```
            <img src="{{product.product_image.url}}" alt="" srcset="" class="img-fluid img-thumbnail border-0" height="150" width="150">
            </a></div>
           <div class="card-body col-sm-9">
            <h6 class="fw-light">
              <a href="{% url 'product-detail' product.id %}">{{product.title}}</a>
            </h6>
            <p>Quantity: {{quantity}}</p>
            <p class="fw-bold">Price: {{product.discounted_price}}</p>
           </div></div>
          <hr>
          <div class="card-body">
            <h6 class="fw-bold">Total Amount: </h6>
            <ul class="list-group">
              <li class="list-group-item d-flex justify-content-between align-items-center border-0 px-0 pb-0">
                Amount<span>Rs. <span id="total-amount">{{amounts.totalamount}}</span></span>
              </li>
              <li class="list-group-item d-flex justify-content-between align-items-center border-0 px-0">
                Shipping<span>Rs. <span id="shipping-amount">{{amounts.shippingamount}}</span></span>
              </li>
              <li class="list-group-item d-flex justify-content-between align-items-center border-0 px-0 mb-3">
                <div>
                  <strong>Total</strong> <small>(including VAT)</small>  </div>
                <span><strong>Rs. <span id="final-amount">{{amounts.finalamount}}</span>
          </strong> </li></ul></div>  </div>
      <small>Term and Condition: <br> Lorem ipsum dolor sit amet consectetur adipisicing elit. Mollitia, ullam saepe! Iure optio repellat dolor velit, minus rem. Facilis cumque neque numquam laboriosam, accusantium adipisci nisi nihil in et quis?</small>
    </div>
```

```
    <div class="col-sm-4 offset-sm-1">
      <h4>Select Shipping Address</h4>
      <hr>
      <form action="/buynowpaymentdone" id="checkout-form">
        <!-- Start Hidden form fields -->
        <input name="prod_id" id="product_id" type="text" class="d-none bg-warning"
value="{{product.id}}">
        <input name="prod_quant" id="prod_quant" type="text" class="d-none bg-warning"
value="{{quantity}}">
        <input name="txn_id" id="txn_id" type="text" class="d-none bg-warning" value="txn
id">
        <!-- End Hidden form fields -->
        {% for address in addresses %}
         <div class="card">
           <div class="card-body">
            <div class="form-check mt-2 mb-2 d-inline-block">
              <input name="custid" id="custaddress{{forloop.counter}}" class="form-check-
input" type="radio" value="{{address.id}}">
              <label class="form-check-label fw-bold" for=""></label>
            </div>
            <h4 class="fw-bold fs-10 d-inline-block mb-0">Address {{forloop.counter}}</h4>
            <span class="fs-13 d-block"><span class="fw-bold">Name:</span>
{{address.name}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">Phone:</span>
{{address.phone}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">Address:</span>
{{address.locality_address}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">City:</span>
{{address.city}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">State:</span>
{{address.state}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">Postal Code:</span>
{{address.zipcode}}</span>
          </div>
```

```
          </div>

        {% empty %}
          <div>There are no Customer addresses added! Go to <strong>Profile</strong> and
create a new address</div>
        {% endfor %}
          <div class="text-end my-4">
            <!-- <button type="submit" class="btn btn-warning mt-3 px-5 my-5 fw-
bold">Continue</button> -->
            <!-- PayPal Integration -->
            <!-- Set up a container element for the button -->
            <div id="paypal-button-container"></div>

          </div>
        </form></div></div></div>
{% endblock main-content %}
{% block custom_js %}
  <script src="{% static 'app/js/cart.js' %}"></script>
  <script>
    document.getElementById("custaddress1").checked = true;
  </script>
{% endblock custom_js %}
{% block payment-gateway %}
  <!-- Include the PayPal JavaScript SDK -->
  <script src="https://www.paypal.com/sdk/js?client-
id={{paypal_clientid}}&currency=USD"></script>

  <!-- PayPal Payment Gateway Functionality -->
  <script>
    // Render the PayPal button into #paypal-button-container
    paypal.Buttons({
        // Set up the transaction
        createOrder: function(data, actions) {
            return actions.order.create({
```

```
            purchase_units: [{amount: {value: '{{usd_amount}}'}}]});},

        // Finalize the transaction
        onApprove: function(data, actions) {
          return actions.order.capture().then(function(orderData) {
            // Successful capture! For demo purposes:
            console.log('Capture result', orderData, JSON.stringify(orderData, null, 2));
            var transaction = orderData.purchase_units[0].payments.captures[0];
            alert('Transaction '+ transaction.status + ': ' + transaction.id + '\n\nSee console for all
available details');

            document.getElementById("txn_id").value = transaction.id;
            document.getElementById("checkout-form").submit();

            // Replace the above to show a success message within this page, e.g.
            // const element = document.getElementById('paypal-button-container');
            // element.innerHTML = '';
            // element.innerHTML = '<h3>Thank you for your payment!</h3>';
            // Or go to another URL:  actions.redirect('thank_you.html');
          });
        }
    }).render('#paypal-button-container');
  </script>
{% endblock payment-gateway %}
```

Checkout.html

```
{% extends 'app/base.html' %}
{% load static %}
{% block custom_css %}
  <!-- Custom CSS -->
  <link rel="stylesheet" href="{% static 'app/css/cart.css' %}" />
{% endblock custom_css %}
{% block title %}Checkout{% endblock title %}
```

```
{% block main-content %}
  <div class="container">
    <div class="row mt-5">
      <div class="col-sm-6 mb-4">
        <h4>Order Summary</h4>
        <div class="card my-4">
          {% for cartitem in cartitems %}
            <div class="row">
              <div class="col-sm-3 text-center align-self-center">
                <a href="{% url 'product-detail' cartitem.product.id %}">
                  <img src="{{cartitem.product.product_image.url}}" alt="" srcset="" class="img-
fluid img-thumbnail border-0 my-5" height="150" width="150">
                </a>
              </div>
              <div class="card-body col-sm-9">
                <h6 class="fw-light">{{cartitem.product.title}}</h6>
                <p>Quantity: {{cartitem.quantity}}</p>
                <p class="fw-bold">Price: {{cartitem.total_price}}</p>
              </div>  </div>
          {% endfor %}
          <hr>
          <div class="card-body">
            <h6 class="fw-bold">Total Amount: </h6>
            <ul class="list-group">
              <li class="list-group-item d-flex justify-content-between align-items-center border-0
px-0 pb-0">
                Amount<span>Rs. <span id="total-
amount">{{amounts.totalamount}}</span></span>
              </li>
              <li class="list-group-item d-flex justify-content-between align-items-center border-0
px-0">
                Shipping<span>Rs. <span id="shipping-
amount">{{amounts.shippingamount}}</span></span>
```

```
            </li>
            <li class="list-group-item d-flex justify-content-between align-items-center border-0
px-0 mb-3">
              <div>
                <strong>Total</strong> <small>(including VAT)</small>
              </div>
              <span><strong>Rs. <span id="final-
amount">{{amounts.finalamount}}</span></strong>
            </li></ul></div></div>

      <small>Term and Condition: <br> Lorem ipsum dolor sit amet consectetur adipisicing elit.
Mollitia, ullam saepe! Iure optio repellat dolor velit, minus rem. Facilis cumque neque
numquam laboriosam, accusantium adipisci nisi nihil in et quis?</small>
    </div>
    <div class="col-sm-4 offset-sm-1">
      <h4>Select Shipping Address</h4>
      <form action="/paymentdone" id="checkout-form" class="my-4">
        <!-- Start Hidden form fields -->
        <input name="txn_id" id="txn_id" type="text" class="d-none bg-warning" value="txn
id">
        <!-- End Hidden form fields -->

        {% for address in addresses %}
          <div class="card mb-2">
            <div class="card-body">
              <div class="form-check mt-2 mb-2 d-inline-block">

                <input name="custid" id="custaddress{{forloop.counter}}" class="form-check-
input" type="radio" value="{{address.id}}">
                <label class="form-check-label fw-bold" for=""></label>

              </div>
              <h4 class="fw-bold fs-10 d-inline-block mb-0">Address {{forloop.counter}}</h4>
```

```
            <span class="fs-13 d-block"><span class="fw-bold">Name:</span>
{{address.name}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">Phone:</span>
{{address.phone}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">Address:</span>
{{address.locality_address}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">City:</span>
{{address.city}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">State:</span>
{{address.state}}</span>
            <span class="fs-13 d-block"><span class="fw-bold">Postal Code:</span>
{{address.zipcode}}</span>
          </div>
         </div>

        {% empty %}
         <div>There are no Customer addresses added! Go to <strong>Profile</strong> and
create a new address</div>
        {% endfor %}

         <div class="text-end my-4">
          <!-- <button type="submit" class="btn btn-warning my-3 px-5 fw-
bold">Continue</button> -->
           <!-- PayPal Integration -->
           <!-- Set up a container element for the button -->
           <div id="paypal-button-container"></div></div></form></div></div></div>
{% endblock main-content %}

{% block custom_js %}
  <script src="{% static 'app/js/cart.js' %}"></script>
  <script>
    document.getElementById("custaddress1").checked = true;
  </script>
{% endblock custom_js %}
```

```
{% block payment-gateway %}
  <!-- Include the PayPal JavaScript SDK -->
  <script src="https://www.paypal.com/sdk/js?client-
id={{paypal_clientid}}&currency=USD"></script>

  <!-- PayPal Payment Gateway Functionality -->
  <script>
    // Render the PayPal button into #paypal-button-container
    paypal.Buttons({
        // Set up the transaction
        createOrder: function(data, actions) {
            return actions.order.create({
                purchase_units: [{amount: {value: '{{usd_amount}}'}  }]});},
        // Finalize the transaction
        onApprove: function(data, actions) {
            return actions.order.capture().then(function(orderData) {
                // Successful capture! For demo purposes:
                console.log('Capture result', orderData, JSON.stringify(orderData, null, 2));
                var transaction = orderData.purchase_units[0].payments.captures[0];
                alert('Transaction '+ transaction.status + ': ' + transaction.id + '\n\nSee console for all
available details');

                document.getElementById("txn_id").value = transaction.id;
                document.getElementById("checkout-form").submit();

                // Replace the above to show a success message within this page, e.g.
                // const element = document.getElementById('paypal-button-container');
                // element.innerHTML = '';
                // element.innerHTML = '<h3>Thank you for your payment!</h3>';
                // Or go to another URL:  actions.redirect('thank_you.html');
            });
        }
    }).render('#paypal-button-container');</script>
{% endblock payment-gateway %}
```

base.css

```
/* Global Properties */
:root {
  /* Light theme color palette */
  --primary-light: #9053c7; /* theme color */
  --secondary-light: #fff; /* secondary item background and text color for buttons */
  --tertiary-light: #57b846; /* button backgrounds */
  --quaternary-light: #000; /* text color */
  --quinary-light: #e9e9e9;
  --senary-light: #333333;
  --septenary-light: #666666;}

/********** Fonts **********/
@font-face {
  font-family: Poppins-Regular;
  src: local(Poppins-regular), url("../fonts/poppins/Poppins-Regular.ttf");}
@font-face {
  font-family: Poppins-Bold;
  src: local(Poppins-bold), url("../fonts/poppins/Poppins-Bold.ttf");}
@font-face {
  font-family: Poppins-Medium;
  src: local(Poppins-medium), url("../fonts/poppins/Poppins-Medium.ttf");}
@font-face {
  font-family: Montserrat-Bold;
  src: local(Montserrat-bold), url("../fonts/montserrat/Montserrat-Bold.ttf");}
@font-face {
  font-family: Fira-Sans-Regular;
  src: local(Fira-sans-regular), url("../fonts/fira-sans/FiraSans-Regular.ttf");}
@font-face {
  font-family: Fira-Sans-Bold;
  src: local(Fira-sans-bold), url("../fonts/fira-sans/FiraSans-Bold.ttf");}
@font-face {
  font-family: Fira-Sans-Medium;
  src: local(Fira-sans-medium), url("../fonts/fira-sans/FiraSans-Medium.ttf");}
```

```
@font-face {
  font-family: Fira-Sans-Light;
  src: local(Fira-sans-light), url("../fonts/fira-sans/FiraSans-Light.ttf");}
/********* Universal Styles **********/
* {margin: 0px;padding: 0px;box-sizing: border-box;}
body,
html { height: 100%; font-family: Poppins-Regular, sans-serif;}
body {transition: background-color 0.3s;}
main { width: 100%; min-height: 100vh; margin: 0 auto; padding-top: 65px;}

/********** Navigation Bar **********/
.navbar {
  padding: 0 10px; padding-right: 60px; z-index: 100;
  background: var(--primary-light);
  background: -webkit-linear-gradient(-135deg, #c850c0, #4158d0);
  background: -o-linear-gradient(-135deg, #c850c0, #4158d0);
  background: -moz-linear-gradient(-135deg, #c850c0, #4158d0);
  background: linear-gradient(-135deg, #c850c0, #4158d0);
  opacity: 90%;  backdrop-filter: blur(5px);}
.navbar-brand {
  font-family: Poppins-Bold; font-size: 1.3rem;
  line-height: 1.2; display: inline; padding: 20px 20px;
  color: rgba(10, 60, 150, 0.8);  text-shadow: rgb(245 245 245 / 60%) 0px 1px 10px;}
.search-btn {
  font-family: Montserrat-Bold;  font-size: 15px; line-height: 1.5;
  color: var(--secondary-light);  text-transform: uppercase; border-radius: 50px;
  background: var(--tertiary-light);  display: -webkit-box; display: -webkit-flex;
  display: -moz-box; display: -ms-flexbox; display: flex;  justify-content: center; align-items:
center;
  -webkit-transition: all 0.4s;  -o-transition: all 0.4s;
  -moz-transition: all 0.4s;  transition: all 0.4s;}
.search-btn:hover {
  background: var(--senary-light);  color: var(--quinary-light);}
.nav-link {
```

```
  display: flex; align-items: center;}
@media (max-width: 1199px) {
  .navbar-brand { padding-top: 20px; padding-left: 0px; padding-right: 0;}
  .navbar { padding-left: 10px; padding-bottom: 0px; padding-right: 10px;}
  .dropdown-menu { background-color: rgba(255, 255, 255, 0); border: none;}
  .dropdown-menu li a { color: #e6e6e6;}
  .navbar-nav.account {  padding-top: 30px;}}
@media (max-width: 991px) {
  .navbar-brand {  padding-top: 20px;  padding-left: 0px;  padding-right: 0;}
  .navbar {  padding-left: 10px;  padding-bottom: 0px;  padding-right: 10px;}
  .dropdown-menu {  background-color: rgba(255, 255, 255, 0);  border: none;}
  .dropdown-menu li a {   color: #e6e6e6; }  .navbar-nav.account {  padding-top: 30px;}}
```

Dark.css

```
/* Global Properties */
:root {  /* Dark theme color palette */
  --primary-dark: #162447;  --secondary-dark: #1f4068;
  --tertiary-dark: #1b1b2f;  --quaternary-dark: #e43f5a;
  --txt-color-dark: antiquewhite;}

/* containers */
.dark body {
  background-color: var(--primary-dark);  color: var(--txt-color-dark);}
.dark .main-container {
  background: none;  background-color: var(--primary-dark) !important;}

/* Form page elements */
.dark .card-container {
  background-color: var(--secondary-dark);}
.dark .card-container .form-title {
  color: var(--txt-color-dark);}
/* Product card */
.dark .product-card,.dark .product-card .item,.dark .product-card .item-img {
```

```
  background-color: var(--secondary-dark);}
.dark .product-card a {  color: var(--txt-color-dark);}
/* Footer elements */
.dark footer {  background-color: var(--tertiary-dark);}
.dark footer svg,.dark footer i {  color: var(--txt-color-dark) !important;  font-size: 100px;}
/* Buttons */
.dark a.btn {  background-color: var(--quaternary-dark);}
/* dark mode toggler button */
.dark .moon-icon {  transform: scale(1);}
.dark .sun-icon {  transform: scale(0);}
```

Form-page.css

```
/* Form Styles */
.form {  width: 290px;}
.form-error {  padding-bottom: 10px;}

.form-title {
  font-family: Poppins-Bold;  font-size: 24px;  color: var(--senary-light);  line-height: 1.2;
  text-align: center;  width: 100%;  display: block;  padding-bottom: 54px;}
.form-subtitle {
  font-family: Poppins-Bold;  font-size: 20px;  color: var(--senary-light);  line-height: 1.2;
  text-align: center;  width: 100%;  display: block;  padding-bottom: 10px;}

.wrap-input {  position: absolute;  width: 100%;  z-index: 1;  margin-bottom: 10px;
  /* border: 4mm ridge rgba(211, 220, 50, .6); */}

.form-input {
  font-family: Poppins-Medium;  font-size: 15px;  line-height: 1.5;  color: var(--septenary-light);
  width: 100%;  background: var(--quinary-light);  height: 50px;  border-radius: 25px;  padding:
0 30px 0 68px;}

.focus-input {
  display: block;  position: absolute;  border-radius: 25px;  bottom: 0;  left: 0;
```

```
  z-index: -1;  width: 100%;  height: 100%;  box-shadow: 0px 0px 0px 0px;
  color: var(--tertiary-light);  opacity: 50%;}

.form-input:focus + .focus-input {
  -webkit-animation: anim-shadow 0.5s ease-in-out forwards;
  animation: anim-shadow 0.5s ease-in-out forwards;}

@-webkit-keyframes anim-shadow {
  to {    box-shadow: 0px 0px 70px 25px;    opacity: 0;  }}
@keyframes anim-shadow {
  to {    box-shadow: 0px 0px 70px 25px;    opacity: 0;  }}

.symbol-input {
  font-size: 15px;  display: -webkit-box;  display: -webkit-flex;  display: -moz-box;  display: -
ms-flexbox;
  display: flex;  align-items: center;  position: absolute; border-radius: 25px;  bottom: 0;
  left: 0;  width: 100%;  height: 100%;  padding-left: 35px;  pointer-events: none;  color: var(--
septenary-light);
  -webkit-transition: all 0.4s;  -o-transition: all 0.4s;  -moz-transition: all 0.4s;  transition: all
0.4s;}
.form-input:focus + .focus-input + .symbol-input {
  color: var(--tertiary-light);  padding-left: 28px;}
```

home.css

```
/* Containers */
.main-container {  width: 100%;  min-height: 100vh;  padding: 15px;}
.section {  width: 100%;  padding: 50px 15px 50px;}

/* Carousel */
.carousel-item-img {  width: 100%;  max-height: 560px;  overflow: hidden;}
.carousel-item-img video {  width: 100%;  min-height: 100%;  object-fit: contain;}

/* Section styles */
```

```
.section .view-all-link {  text-align: center;  padding-top: 20px;}
.section .view-all-link a {  text-decoration: none;  font-size: 20px;}

/* Product List Display */
.product-list-title {  font-family: Montserrat-bold;  display: flex;  align-items: center;}
.product-list-title .form-switch {  display: inline;  margin-left: auto;  padding-left: 90px;  text-
align: center;}

.product-list {
  text-align: center;  overflow-x: scroll;  white-space: nowrap;  padding: 10px;  scroll-behavior:
smooth;}
.product-list .product-card {
  display: inline-block;  margin: 10px 2px 20px;  transition: 0.2s transform ease, 0.2s box-
shadow ease;
  box-shadow: rgba(17, 17, 26, 0.1) 0px 4px 16px,    rgba(17, 17, 26, 0.05) 0px 8px 32px;}
.product-list .product-card:hover {
  box-shadow: rgba(100, 100, 111, 0.2) 0px 7px 29px 0px;  transform: scale(1.02);}
.product-list .product-card-btn {  display: inline-block;  color: var(--quaternary-light);}
.product-card .item {  background-color: var(--secondary-light);  padding: 20px;  border-radius:
5px;}
.product-card .item-img {  width: 250px;  height: 200px;  overflow: hidden;}
.product-card .item-img img {  width: 100%;  min-height: 100%;  object-fit: contain;}
.item .btn-container {  text-align: center;}
.item .title {  padding-top: 0.5em;  line-height: 1.5em;  height: 3.5em;
  white-space: normal;  overflow: hidden;  text-overflow: ellipsis;  width: 230px;  display: -
webkit-box;
  -webkit-line-clamp: 2;  -webkit-box-orient: vertical;}
.product-card .btn-primary {
  font-family: Montserrat-Bold;  font-size: 15px;  line-height: 1.5;  color: var(--secondary-light);
  text-transform: uppercase;  height: 40px;  border-radius: 25px;  background: var(--tertiary-
light);
  display: -webkit-box;  display: -webkit-flex;  display: -moz-box;  display: -ms-flexbox;
display: flex;
```

```
  justify-content: center;  align-items: center;  padding: 0 25px;  border: none;  -webkit-
transition: all 0.4s;
  -o-transition: all 0.4s;  -moz-transition: all 0.4s;  transition: all 0.4s;}
.product-card .btn-primary:hover {  background: var(--senary-light);}

/********** Toggle button **********/
.form-switch .form-check-input:checked {  border: 10px;  background-color: var(--primary-
light);}
.form-switch .form-check-input:focus:not(:checked) {  background-color: rgb(241, 241, 241);}

/********** Scroll bar **********/
.product-list::-webkit-scrollbar {  width: 5px;  height: 5px;}
.product-list::-webkit-scrollbar-track {  background-color: rgba(255, 255, 255, 0.1);  border-
radius: 10px;}
.product-list::-webkit-scrollbar-thumb {  background-color: #11171a;  border-radius: 10px;}
```

Profile.css

```
.btn {  box-shadow: none !important;  margin: 2px;  border: none !important;}
.btn:hover {  background-color: var(--quinary-light);  border: none !important;}
.btn:hover .fa-trash {  color: black;}
.btn-primary {  background-color: var(--tertiary-light);  border: none;  box-shadow: none
!important;}
.btn-primary:active,.btn-primary:hover {  background-color: var(--senary-light);}

.card {
  display: inline-block;  margin: 0px 4px 7px;  width: max-content;  transition: 0.2s transform
ease, 0.2s box-shadow ease;
  box-shadow: rgba(17, 17, 26, 0.1) 0px 4px 16px,    rgba(17, 17, 26, 0.05) 0px 8px 32px;
color: var(--quaternary-light);
  border: none;  padding: 20px;}
.card span {  overflow: visible;  word-wrap: break-word;}
.card:hover {  box-shadow: rgba(100, 100, 111, 0.2) 0px 7px 29px 0px;  transform:
scale(1.02);}
```

Product.css

```
/* Global Properties */
:root {
  /* Light theme color palette */
  --cart-btn-light: #006eff;  --buy-btn-light: #f30029;}

.main-container {  width: 100%;  min-height: 100vh;  padding: 15px;}
.product-detail h4,h5,li,span {  font-family: Poppins-Regular;}
pre {  font-family: Poppins-Regular;  font-size: 1.1rem;}

/* Product image style */
.product-img img {  display: block;  margin-bottom: 50px;  margin-top: 50px;  border: none;}

/* Product detail section */
.product-detail {  padding: 20px;  font-family: Poppins-Bold;}
@media (max-width: 576px) {
  .product-detail {    padding-top: 50px;  }}

/* Button styles */
.product-btn {  margin-top: 5px;  margin-bottom: 5px;
  font-family: Montserrat-Bold;  font-size: 12px;  line-height: 1.5;  max-width: fit-content;
color: var(--secondary-light);
  text-transform: uppercase;  height: 40px;  border-radius: 25px;  display: -webkit-inline-box;
display: -webkit-inline-flex;
  display: -moz-inline-box;  display: -ms-inline-flexbox;  display: inline-flex;  justify-content:
center;
  align-items: center;  padding: 0 25px;  -webkit-transition: all 0.4s;  -o-transition: all 0.4s;  -
moz-transition: all 0.4s;
  transition: all 0.4s;}
.product-btn.cart {  background: var(--cart-btn-light);}
.product-btn.buy {  margin-left: 10px;  background: var(--buy-btn-light);}
.product-btn:hover {  background: var(--senary-light);  color: var(--secondary-light);}

/* Read More styles in description */
```

```
#productDescription pre {  white-space: pre-wrap;
  white-space: -moz-pre-wrap;  white-space: -pre-wrap;  white-space: -o-pre-wrap;  word-wrap:
break-word;}
#productDescription.collapse:not(.show) {  display: block;  height: 4.4rem;  overflow: hidden;
white-space: nowrap;  text-overflow: ellipsis;}
#productDescription.collapsing {  height: 4.4rem;}
```

Cart.js

```
$('.plus-cart').click(function () {
  var id = $(this).attr("pid");
  var quantity_el = this.parentNode.children[2];
  var ship_amt_el = document.getElementById('shipping-amount');
  var total_amt_el = document.getElementById('total-amount');
  var final_amt_el = document.getElementById('final-amount');

  $.ajax({
    type: "GET",    url: "/pluscartitem",    data: {      product_id: id,    },
    success: function (data) {
      quantity_el.innerText = data.quantity;
      ship_amt_el.innerText = data.shippingamount;
      total_amt_el.innerText = data.totalamount;
      final_amt_el.innerText = data.finalamount;
    }  })})

$('.minus-cart').click(function () {
  var id = $(this).attr("pid");
  var quantity_el = this.parentNode.children[2];
  var ship_amt_el = document.getElementById('shipping-amount');
  var total_amt_el = document.getElementById('total-amount');
  var final_amt_el = document.getElementById('final-amount');

  $.ajax({    type: "GET",    url: "/minuscartitem",    data: {      product_id: id,    },
    success: function (data) {
```

```
      quantity_el.innerText = data.quantity;
      ship_amt_el.innerText = data.shippingamount;
      total_amt_el.innerText = data.totalamount;
      final_amt_el.innerText = data.finalamount;
    }  })})

$('.delete-cart').click(function () {
  var id = $(this).attr("pid");
  var el = this;
  var ship_amt_el = document.getElementById('shipping-amount');
  var total_amt_el = document.getElementById('total-amount');
  var final_amt_el = document.getElementById('final-amount');
  var cart_badge = document.getElementById('cart_badge');

  $.ajax({    type: "GET",    url: "/removecartitem",    data: {      product_id: id,    },
    success: function (data) {
      if (!data.empty){
        ship_amt_el.innerText = data.shippingamount;
        total_amt_el.innerText = data.totalamount;
        final_amt_el.innerText = data.finalamount;
        el.parentNode.parentNode.parentNode.parentNode.remove();
        cart_badge.innerText = parseInt(cart_badge.innerHTML) - 1;
      } else {        location.reload();      }    }  })})
```

Homescript.js

```
// Pause auto scroll when mouse enters the product list div
function pauseScroll(number, state) {
  var checbox = document.getElementById(`scroll-checkbox${number}`);  checbox.checked =
!state;}

// Auto scroll list
function scrollDiv() {
  var productList = document.getElementsByClassName("product-list");
```

```
  var checkBoxes = document.getElementsByClassName("autoscroll-checkbox");
  for (let i = 0; i < productList.length; i++) {
    if (checkBoxes[i].checked === false) {      continue;    }
    const scroll_width = productList[i].scrollWidth;
    const off_width = productList[i].offsetWidth;
    const scrollPos = productList[i].scrollLeft;
    const scrollBy = 50;
    productList[i].scrollLeft += scrollBy;
    if (scrollPos === scroll_width - off_width) {      productList[i].scrollLeft = 0;    }  }}
window.onload = () => {  setInterval(scrollDiv, 500);};
```

4.2 TESTING APPROACH

As websites grow they become harder to test manually. Not only is there more to test, but, as interactions between components become more complex, a small change in one area can impact other areas, so more changes will be required to ensure everything keeps working and errors are not introduced as more changes are made. One way to mitigate these problems is to write automated tests, which can easily and reliably be run every time you make a change. This tutorial shows how to automate unit testing of your website using Django's test framework.

4.2.1 TYPES OF TESTING

There are numerous types, levels, and classifications of tests and testing approaches. The most important automated tests are:

- UNIT TESTS

Verify functional behavior of individual components, often to class and function level.

- REGRESSION TESTS

Tests that reproduce historic bugs. Each test is initially run to verify that the bug has been fixed, and then re-run to ensure that it has not been reintroduced following later changes to the code.

- INTEGRATION TESTS

Verify how groupings of components work when used together. Integration tests are aware of the required interactions between components, but not necessarily of the internal operations of each component. They may cover simple groupings of components through to the whole website.

4.2.2 BLACKBOX TESTING

Black box testing is also known as opaque technique, behavioral testing, functional testing, and closed-box testing is a type of software testing. When we enter a topic to search on the search engine, we type out the topic and enter search. The result is obtained thereafter without looking at the internal structure or working.

4.2.2.1 UNIT TESTING

Django provides a test framework with a small hierarchy of classes that build on the Python standard unittest library. Despite the name, this test framework is suitable for both unit and integration tests. The Django framework adds API methods and tools to help test web and Django-specific behavior. These allow we to simulate requests, insert test data, and inspect wer application's output. Django also provides an API (LiveServerTestCase) and tools for using different testing frameworks, for example we can integrate with the popular Selenium framework to simulate a user interacting with a live browser.

Local library Tests

- **Models**

Consider the Customer model below. Here we should test the labels for all the fields, because even though we haven't explicitly specified most of them, we have a design that says what these values should be. If we don't test the values, then we don't know that the field labels have their intended values. Similarly while we trust that Django will create a field of the specified length, it is worthwhile to specify a test for this length to ensure that it was implemented as planned.

Here we'll see that we first import TestCase and derive our test class (CustomerModelTest) from it, using a descriptive name so we can easily identify any failing tests in the test output. We then call setUpTestData() to create an author object that we will use but not modify in any of the tests.

If we created the Customer model in the models it is quite likely that we will get an error for the date_of_death label as shown below. The test is failing because it was written expecting the label definition to follow Django's convention of not capitalizing the first letter of the label (Django does this for you).

This is a very minor bug, but it does highlight how writing tests can more thoroughly check any assumptions you may have made.

- **Forms**

The philosophy for testing our forms is the same as for testing our models; we need to test anything that we've coded or our design specifies, but not the behavior of the underlying framework and other third party libraries.

Generally this means that we should test that the forms have the fields that we want, and that these are displayed with appropriate labels and help text. We don't need to verify that Django validates the field type correctly (unless we created our own custom field and validation) — i.e. we don't need to test that an email field only accepts emails. However we would need to test any additional validation that we expect to be performed on the fields and any messages that our code will generate for errors.

That's all for forms; we do have some others, but they are automatically created by our generic class-based editing views, and should be tested there! Run the tests and confirm that our code still passes!

- **Views**

To validate our view behavior we use the Django test Client. This class acts like a dummy web browser that we can use to simulate GET and POST requests on a URL and observe the response. We can see almost everything about the response, from low-level HTTP (result headers and status codes) through to the template we're using to render the HTML and the context data we're passing to it. We can also see the chain of redirects (if any) and check the URL and status code at each step. This allows us to verify that each view is doing what is expected.

As this is a generic list view almost everything is done for us by Django. Arguably if you trust Django then the only thing you need to test is that the view is accessible at the correct URL and can be accessed using its name. However if you're using a test-driven development process you'll start by writing tests that confirm that the view displays all Authors, paginating them in lots of 10.

Test case for Register Page.

Test Case Id -TC001

S No.	Input/ Action	Expected Result	Actual Result	Remark
1	Leave text/ Field empty.	Will show error message “Enter First Name” “Enter Last Name” “Enter Username” “Enter Email” “Enter Password” “Enter Confirm Password”	Error message “Enter First Name” “Enter Last Name “ “Enter Username” “Enter Email” “Enter Password” “Enter Confirm Password”	Pass
2	Entered Valid username.	Will accept the data.	Data accepted	Pass

Test case for Login Page.

Test Case Id -TC002

S No.	Input/ Action	Expected Result	Actual Result	Remark
1	Leave text/ Field empty.	Will show error message “Enter Username" “Enter Password "	Error message “Enter Username" “Enter Password "	Pass
2	Entered Invalid username or password.	Will show error message “Please enter a correct username and password. Note that both fields may be case-sensitive.”	Error message “Please enter a correct username and password. Note that both fields may be case-sensitive.”	Pass
3	Entered Valid username.	Will accept the data.	Data accepted	Pass

4.2.2.2 INTEGRATION TESTING

Integration tests are at the opposite end of the spectrum. These tests usually cover multiple different facets of the application working together to produce a result. They ensure that data flow is right & often handle multiple user interactions.

Testing is a very challenging task and needs a lot of planning before you start the testing. Also while testing there are numerous things to be taken care of in order to identify the defects, which improves the quality of the system.

“Testing is the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements, or to identify differences between expected and actual results." IEEE

Testing methodology that we have considered is V&V Model. In this model, testing team works in parallel with the development team. This provides the testing team sufficient time for preparation in order to deliver the application with high quality.

Validation: Ensures the building of right product

Verification: Ensures that the product is right

4.2.2.3 REGRESSION TESTING

Regression testing is a software testing practice that ensures an application still functions as expected after any code changes, updates, or improvements.

Regression testing is responsible for the overall stability and functionality of the existing features. Whenever a new modification is added to the code, regression testing is applied to guarantee that after each update, the system stays sustainable under continuous improvements.

Changes in the code may involve dependencies, defects, or malfunctions. Regression testing targets to mitigate these risks, so that the previously developed and tested code remains operational after new changes.

Generally, an application goes through multiple tests before the changes are integrated into the main development branch. Regression testing is the final step, as it verifies the product behaviors as a whole.

4.2.2.4 SMOKE TESTING

Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as “Build Verification Testing” or “Confidence Testing.”

In simple terms, we are verifying whether the important features are working and there are no showstoppers in the build that is under testing. It is a mini and rapid regression test of major functionality. It is a simple test that shows the product is ready for testing. This helps determine if the build is flawed as to make any further testing a waste of time and resources.

4.2.3 WHITE BOX TESTING

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

4.2.3.1 SECURITY TESTING

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss. Security testing of any system is focuses on finding all possible loopholes and weaknesses of the system which might result into the loss of information or repute of the organization.

4.2.3.2 MUTATION TESTING

Mutation Testing is a type of software testing in which certain statements of the source code are changed/mutated to check if the test cases are able to find errors in source code. The goal of Mutation Testing is ensuring the quality of test cases in terms of robustness that it should fail the mutated source code.

The changes made in the mutant program should be kept extremely small that it does not affect the overall objective of the program. Mutation Testing is also called Fault-based testing strategy as it involves creating a fault in the program and it is a type of White Box Testing which is mainly used for Unit Testing.

4.2.4 ALPHA TESTING

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the user acceptance testing.

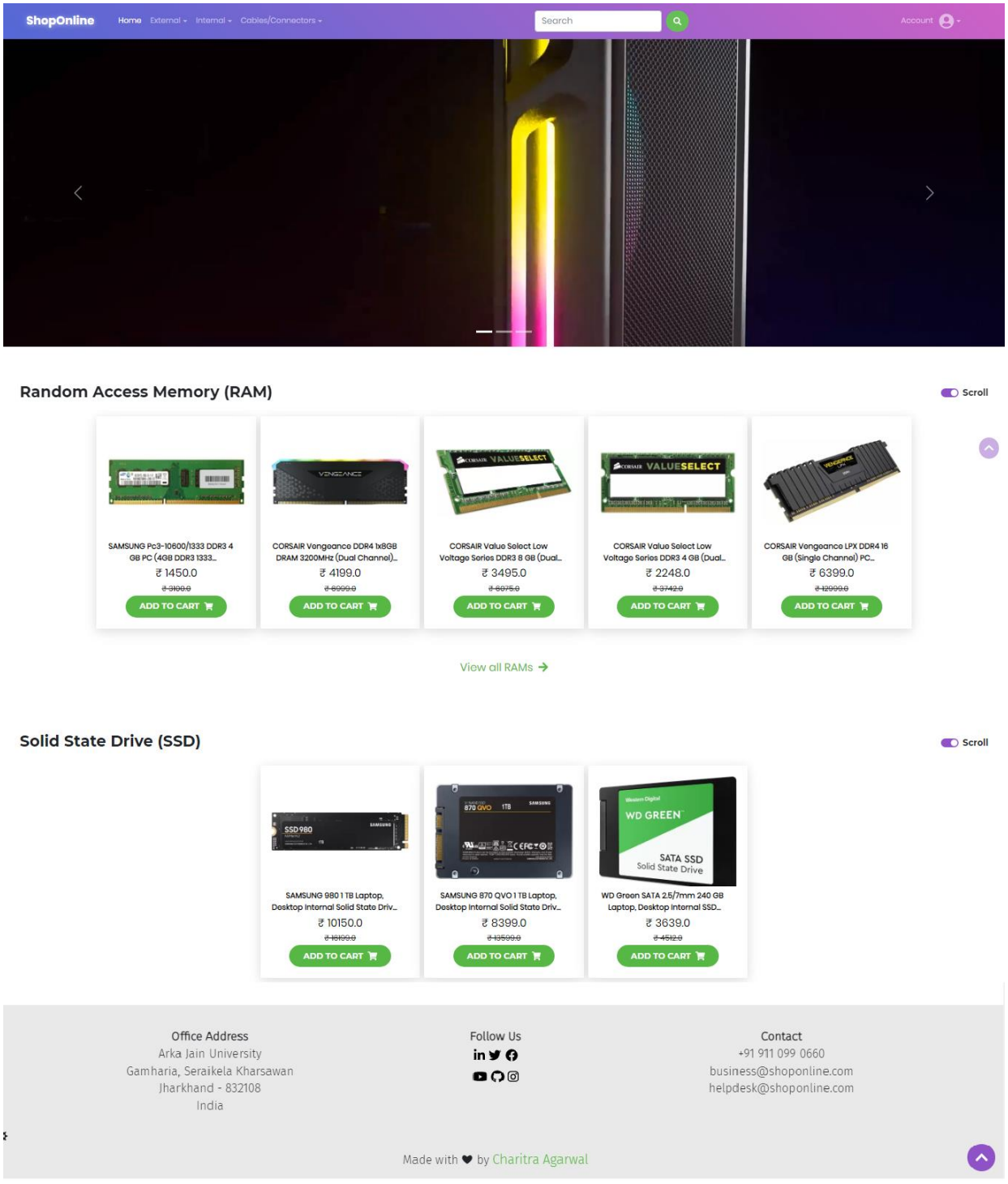
This is referred to as an alpha testing only because it is done early on, near the end of the development of the software. Alpha testing is commonly performed by homestead software engineers or quality assurance staffs. It is the last testing stage before the software is released into the real world.

4.2.5 BETA TESTING

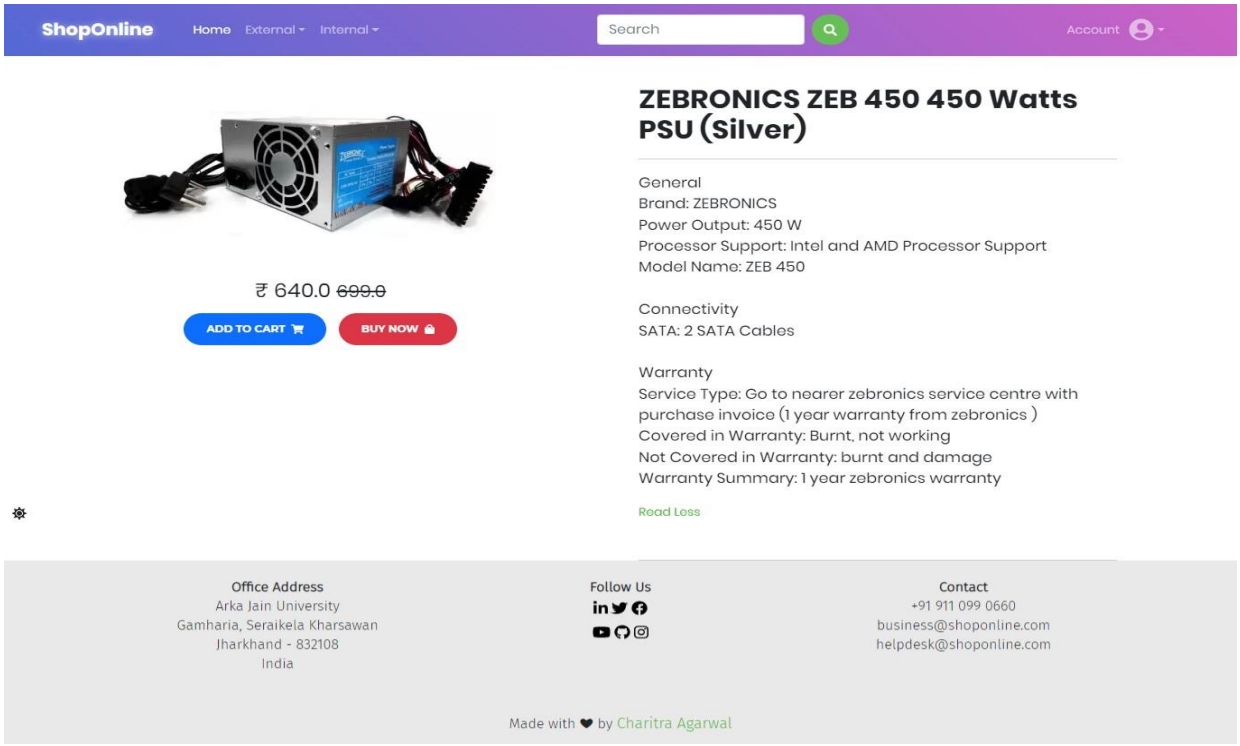
Beta testing is a phase of acceptance testing that involves having end-users test the product before its official product launch. The group of end-users that act as beta testers would generally be smaller than the number of customers a full release would reach. However, we'll cover the different types of beta tests (including their scopes) in the next section.

A beta program gives developers a realistic perspective on how good their product is. That's because developers get the chance to hear about the product's quality from end-users, whose perspective is fundamentally different from the devs' own. Beta testers provide feedback on the app to its developers. That feedback then gets processed and turned into guidelines for improvements to the app.

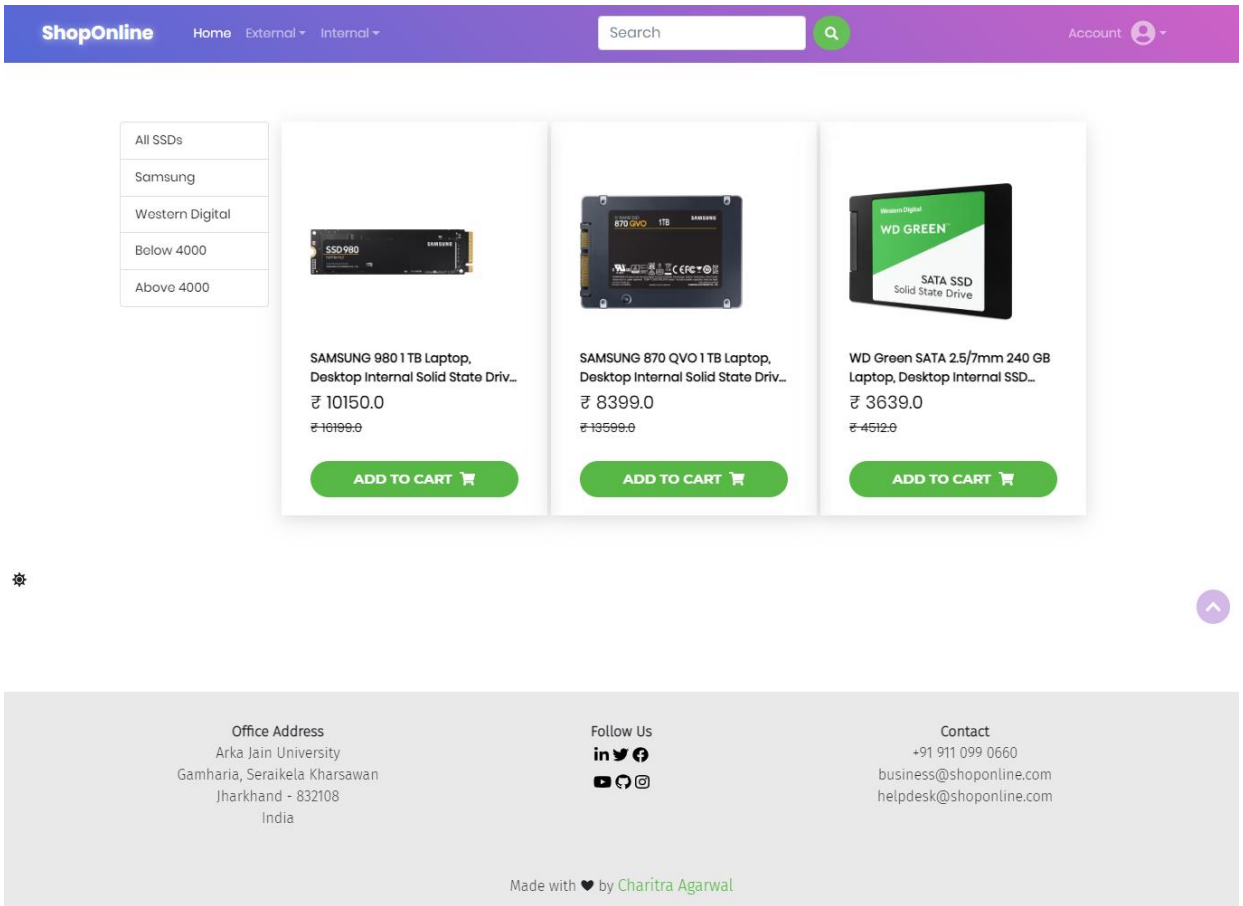
CHAPTER 5 : RESULT AND DISCUSSION



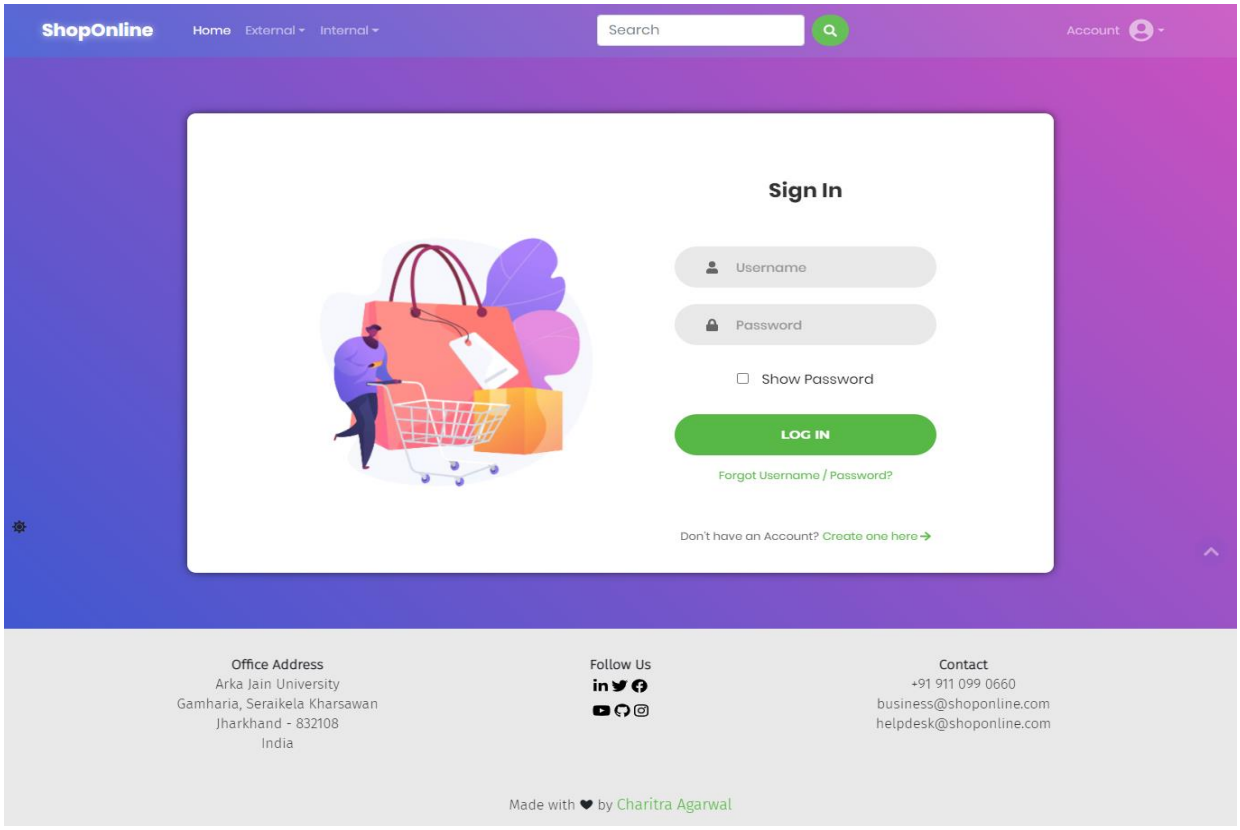
Home Page



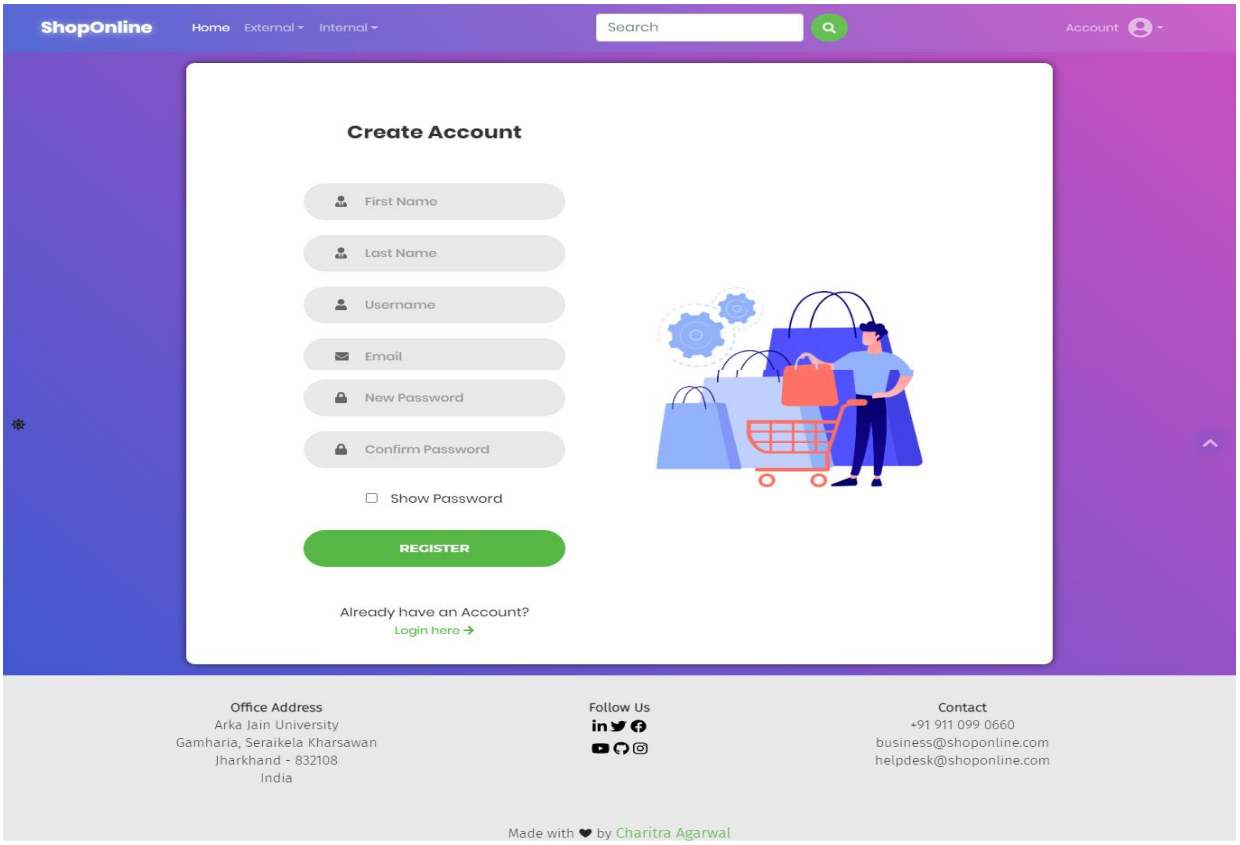
Product Page



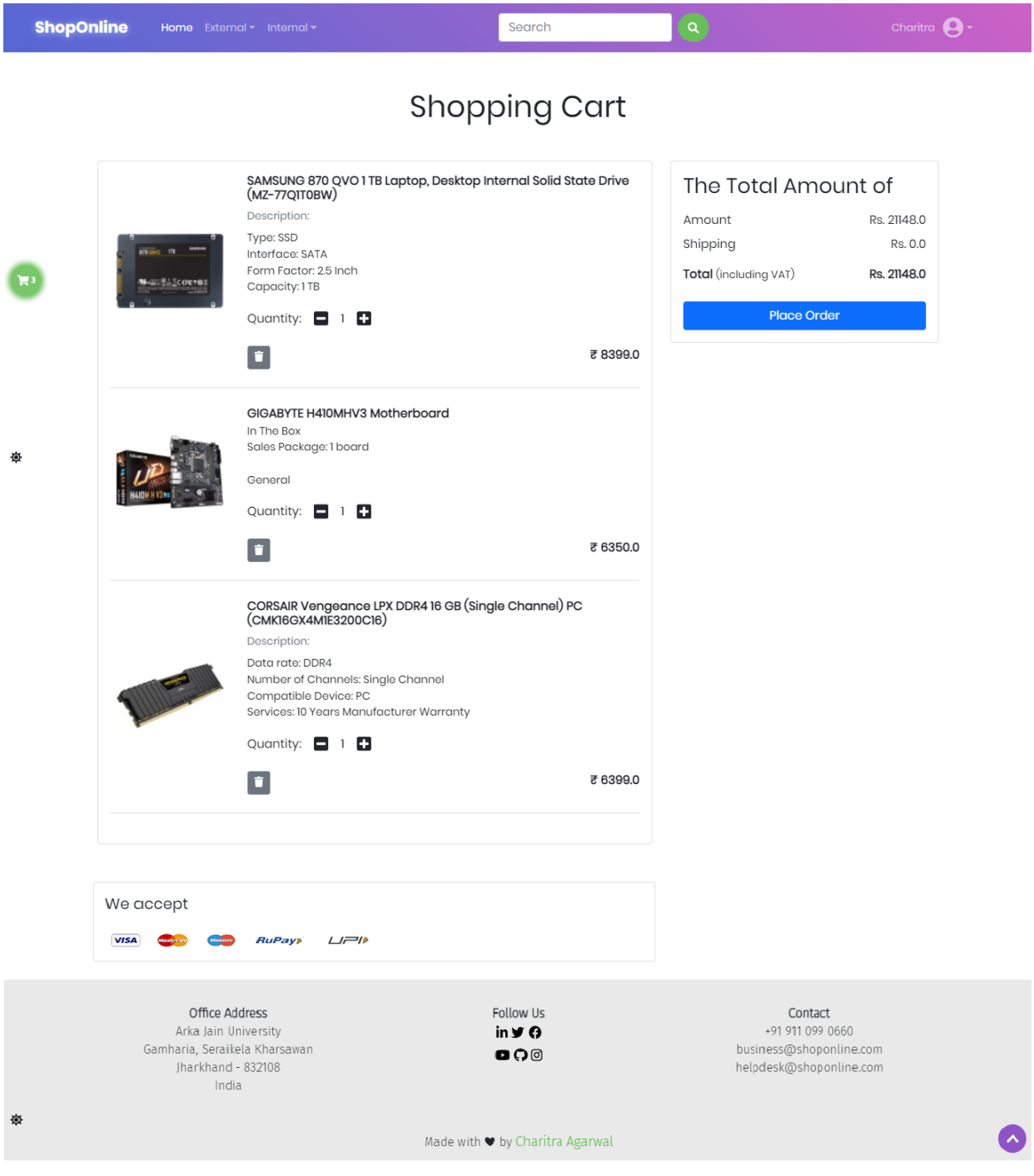
Solid State Drive Page



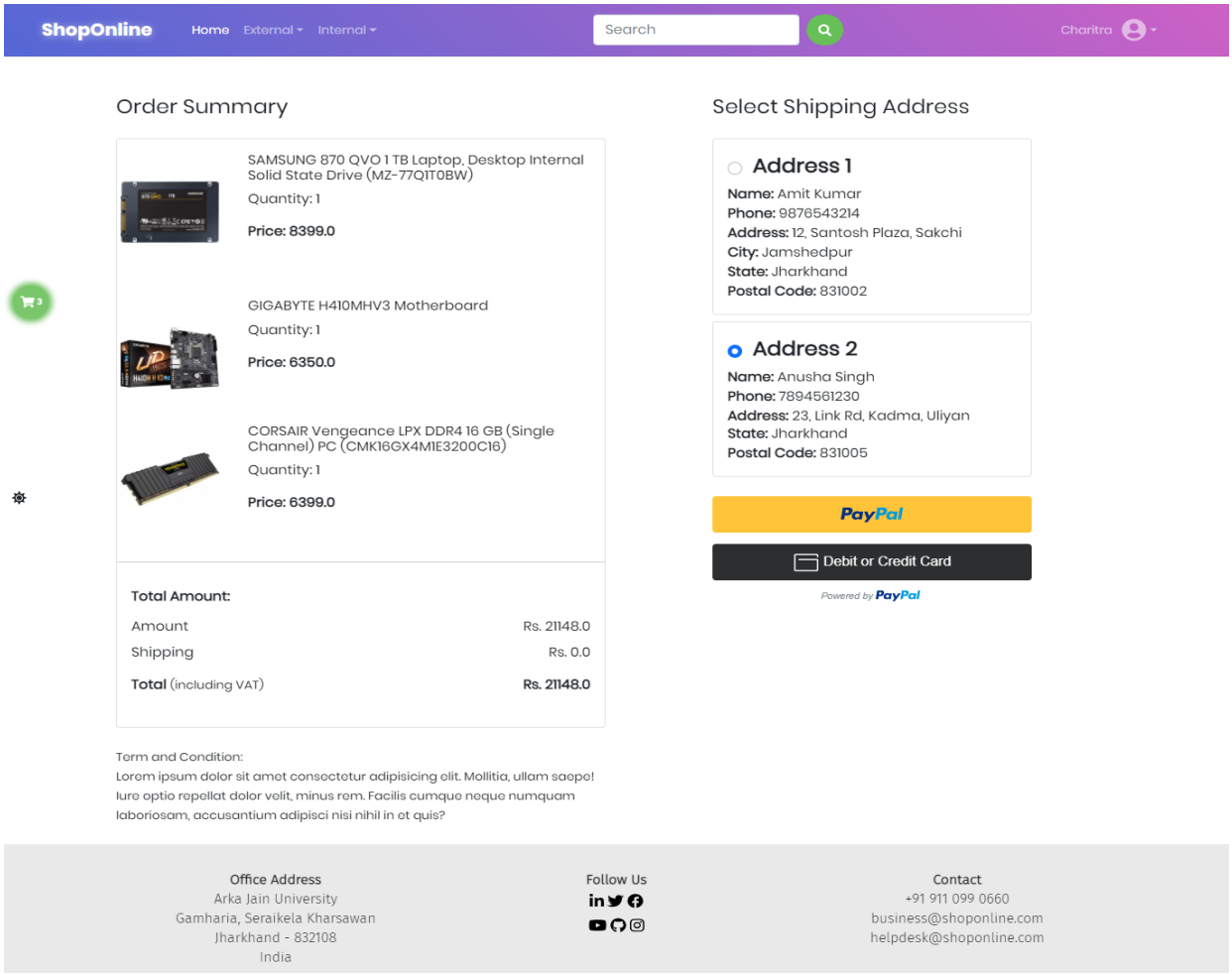
Login Page



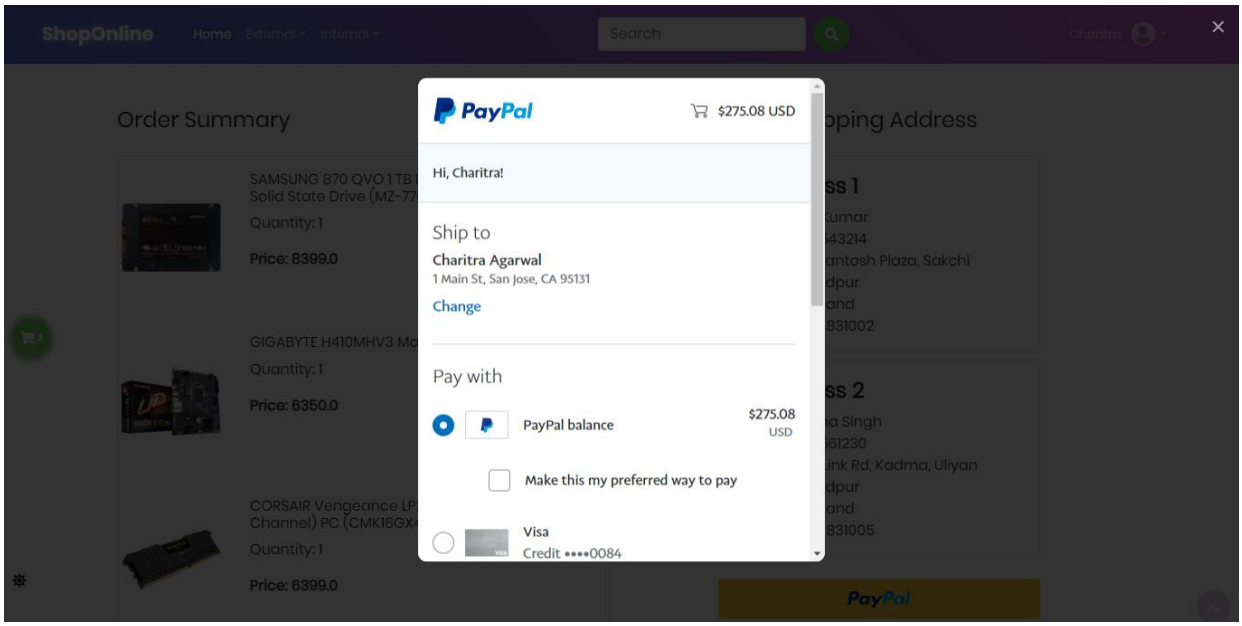
Register Page


ShopOnline
Home
External
Internal
Search
Charitra
Shopping Cart
SAMSUNG 870 QVO 1 TB Laptop, Desktop Internal Solid State Drive (MZ-77Q1T0BW)
Description:
Type: SSD
Interface: SATA
Form Factor: 2.5 Inch
Capacity: 1 TB
Quantity: 1
₹ 8399.0
GIGABYTE H410MHV3 Motherboard
In The Box
Sales Package: 1 board
General
Quantity: 1
₹ 6350.0
CORSAIR Vengeance LPX DDR4 16 GB (Single Channel) PC (CMK16GX4M1E3200C16)
Description:
Data rate: DDR4
Number of Channels: Single Channel
Compatible Device: PC
Services: 10 Years Manufacturer Warranty
Quantity: 1
₹ 6399.0
The Total Amount of
Amount Rs. 21148.0
Shipping Rs. 0.0
Total (including VAT) Rs. 21148.0
Place Order
We accept
Office Address
Arka Jain University
Gamharia, Seraikela Kharsawan
Jharkhand - 832108
India
Follow Us
Contact
+91 911 099 0660
business@shoponline.com
helpdesk@shoponline.com
Made with ♥ by Charitra Agarwal

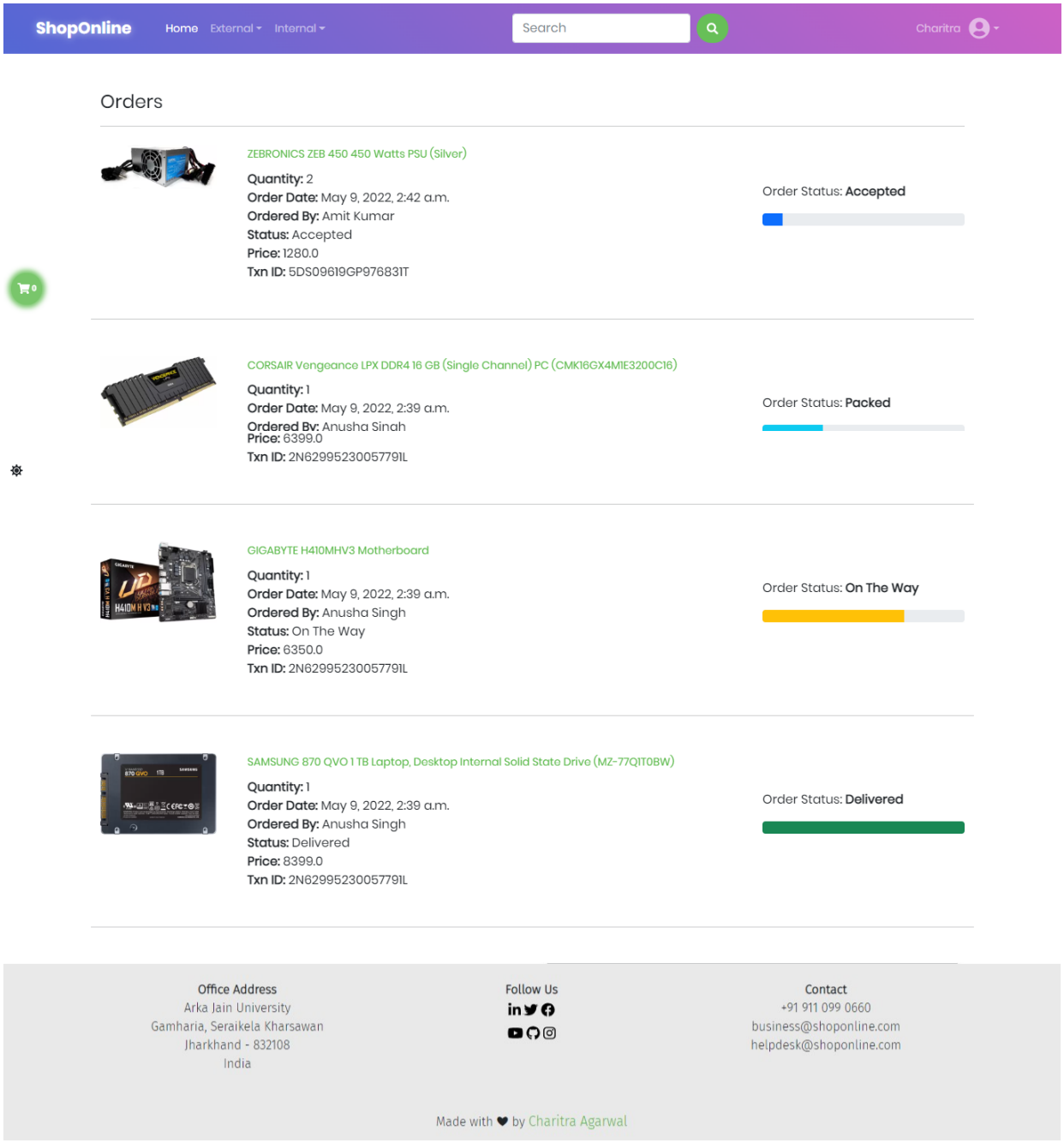
Cart Page



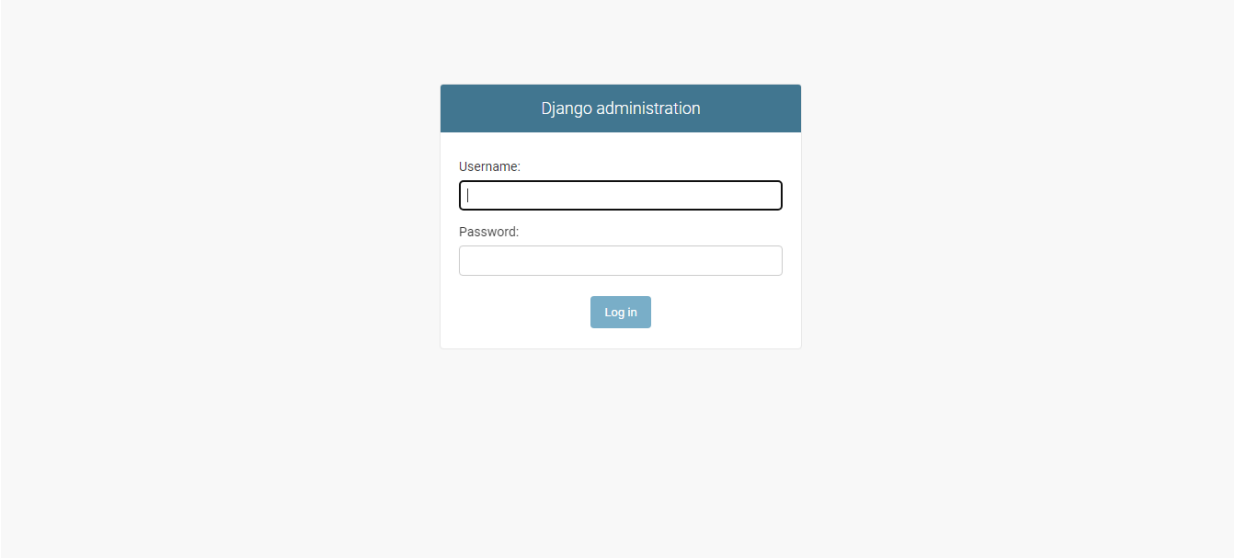
Checkout Page



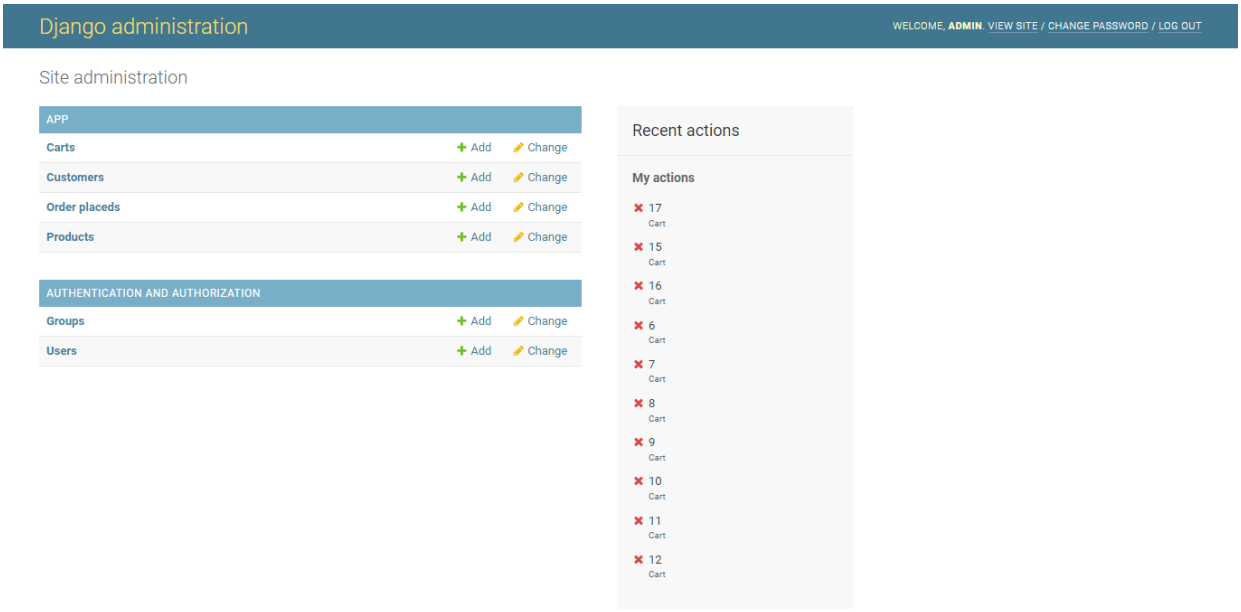
Payment Gateway Section


ShopOnline
Home
External
Internal
Search
Charitra
Orders
ZEBRONICS ZEB 450 450 Watts PSU (Silver)
Quantity: 2
Order Date: May 9, 2022, 2:42 a.m.
Ordered By: Amit Kumar
Status: Accepted
Price: 1280.0
Txn ID: 5DS09619GP97683IT
Order Status: Accepted
CORSAIR Vengeance LPX DDR4 16 GB (Single Channel) PC (CMK16GX4M1E3200C16)
Quantity: 1
Order Date: May 9, 2022, 2:39 a.m.
Ordered By: Anusha Sinah
Price: 6399.0
Txn ID: 2N62995230057791L
Order Status: Packed
GIGABYTE H410MHV3 Motherboard
Quantity: 1
Order Date: May 9, 2022, 2:39 a.m.
Ordered By: Anusha Singh
Status: On The Way
Price: 6350.0
Txn ID: 2N62995230057791L
Order Status: On The Way
SAMSUNG 870 QVO 1 TB Laptop, Desktop Internal Solid State Drive (MZ-77QIT0BW)
Quantity: 1
Order Date: May 9, 2022, 2:39 a.m.
Ordered By: Anusha Singh
Status: Delivered
Price: 8399.0
Txn ID: 2N62995230057791L
Order Status: Delivered
Office Address
Arka Jain University
Gamharia, Seraikela Kharsawan
Jharkhand - 832108
India
Follow Us
Contact
+91 911 099 0660
business@shoponline.com
helpdesk@shoponline.com
Made with ♥ by Charitra Agarwal

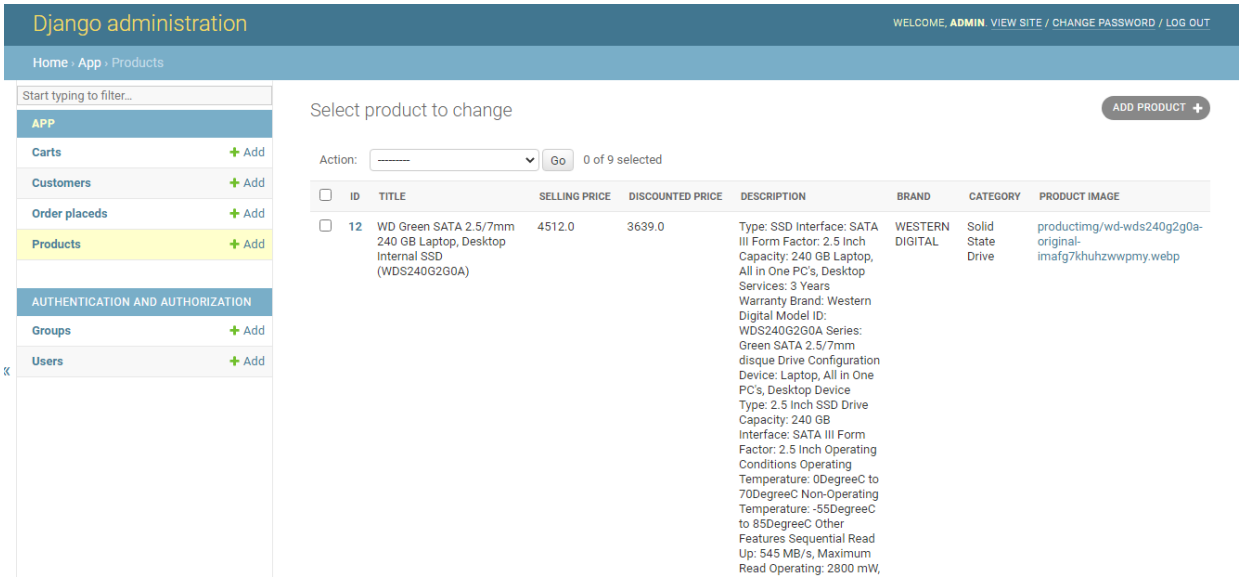
Orders Page



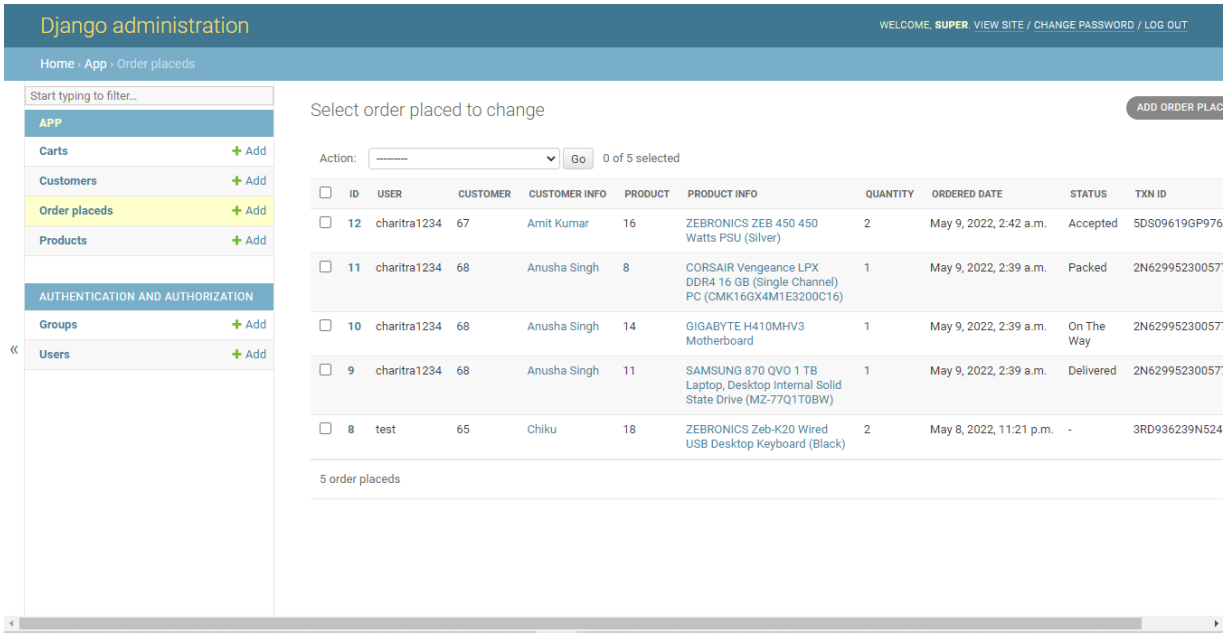
Admin login



Admin Home



Add Product Page



Manage Orders Page

CHAPTER 6 : CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This project is only a humble venture to satisfy the needs in a shop. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the organization. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses. This website provides a computerized version of shop manipulate system which will benefit the users as well as the visitor of the shop. It makes entire process online where users can search product, and buy various product. It also has a facility for common user by login into the system where user can login and can see status of ordered item as well request for items or give some suggestions. It provide the facility of admin's login where admins can add various item, review users activity and also give occasional discount and also add info about different events for the customer.

6.2 FUTURE SCOPE OF THE PROJECT

The project has a very vast scope in future. The project can be implemented on intranet in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of database Space Manager ready and fully functional the client is now able to manage and hence run the entire work in a much better, accurate and error free manner.

The following are the future scope for the project.

- ✓ Can be added inventory management system
- ✓ Can be added multiple branches
- ✓ Can be added multilingual to this site
- ✓ And many features can be added this project to make it more robust.

CHAPTER 7 : REFERENCES

The following reference has been used to develop the project “ONLINE SHOPPING SYSTEM”:

- **https://www.w3schools.com/django/**
- **https://www.tutorialspoint.com/django/**
- **https://www.geeksforgeeks.org/django-tutorial/**
- **https://www.youtube.com/playlist?list=PLbGui_ZYuhigchy8DTw4pX4duTTpvqlh6**
- **https://stackoverflow.com/**
- **https://www.javatpoint.com/django-tutorial**