FULL STACK DEVELOPMENT

1. Full Stack Development refers to the practice of developing both the front-end (client- side) and back-end (server-side) portions of web applications.
2. A full stack developer is proficient in working with both the front-end and back-end technologies, allowing them to build complete web applications independently or as part of a team.

Technologies used in full stack development:

Front-End Technologies:

- HTML (Hypertext Markup Language): Used for structuring web pages.
- CSS (Cascading Style Sheets): Used for styling the appearance of web pages.
- JavaScript: A programming language used for adding interactivity and dynamic behavior to web pages.
- Front-end frameworks/libraries such as React.js, AngularJS, or Vue.js: These provide tools and utilities for building user interfaces and managing application state.

Back-End Technologies:

- Server-side languages like JavaScript (Node.js), Python (Django, Flask), Ruby (Ruby on Rails), Java (Spring Boot), or PHP (Laravel), C# (.Net Framework), java (Servlets)
- Databases such as MySQL, PostgreSQL, MongoDB, or Firebase for storing and managing data.
- Web servers like Apache or Nginx or IIS or Tomcat or Caddy (with built in support for https) for handling HTTP requests.

Development Tools and Environment:

- Version control systems like Git for managing code changes.
- Integrated Development Environments (IDEs) such as Visual Studio Code, Sublime Text, or Atom.
- Command-line tools for tasks like package management (npm for Node.js, pip for Python, nugget for .Net), running servers, and deployment.

Important STACKS

- MEAN Stack
- MERN Stack
- Django Stack
- LAMP
- WAMP

Why Django?

Rapid Development: - DRY (Don't repeat Yourself)

- Saves Time and Money
- Rounded Solution
- Great Exposure
- Complete Ownership
- Greater opportunity with more learning
- Security (Prevention of threats such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and clickjacking by providing {% csrf_token %} Django tag to be included inside the form.

This token generates a hidden input field containing a unique CSRF token. This token is then validated by Django when the form is submitted, ensuring that the request originated from the same site and protecting against CSRF attacks.

- Batteries Included Philosophy (Model Forms)
- Built in database (SQLite DB)

What is Django

- Django is a free and open source web application framework which offers fast and effective dynamic website development.
- It is written using python.
- It follows MVT (model, view and template architecture)

Framework

A framework is a pre-built collection of libraries, modules, and tools that provides a structured approach to developing software applications.

- Django is a web development framework.
- AngularJS is a Web Frontend development framework
- React is a Web frontend development library.

Django Evolution

1. Write a Web application from scratch.
2. Write another Web application from scratch.
3. Realize the application from step 1 shares much in common with the application from step2.
4. Refactor the code so that application 1 shares code with application 2.
5. Repeat steps 2–4 several times.
6. Realize you've invented a framework

Django MVT

The MVT is a software design pattern which includes three important components Model, View and Template.

- The Model helps to handle database. It is a data access layer which handles the data.
- The Template is a presentation layer which handles User Interface part completely.
- The View is used to execute the business logic and interact with a model to carry data and renders a template.

Characteristics of Django

- Loosely Coupled − Django helps you to make each element of its stack independent of the others.
- Less code - Ensures effective development
- Not repeated- Everything should be developed in precisely one place instead of repeating it again
- Fast development- Django's offers fast and reliable application development.
- Consistent design - Django maintains a clean design and makes it easy to follow the best web development practices.

Python Virtual Environment

- A Python Virtual Environment is an isolated space where you can work on your Python projects, separately from your system-installed Python.
- You can set up your own libraries and dependencies without affecting the system Python.
- There are no limits to the number of virtual environments

- It allows you to have multiple Python environments with different versions of Python and different sets of installed packages on the same system.
- It is generally good to have one new virtual environment for every Python-based project you work on
- You can change the system python version, django version and other dependencies without affecting the project python version, django versions and dependencies

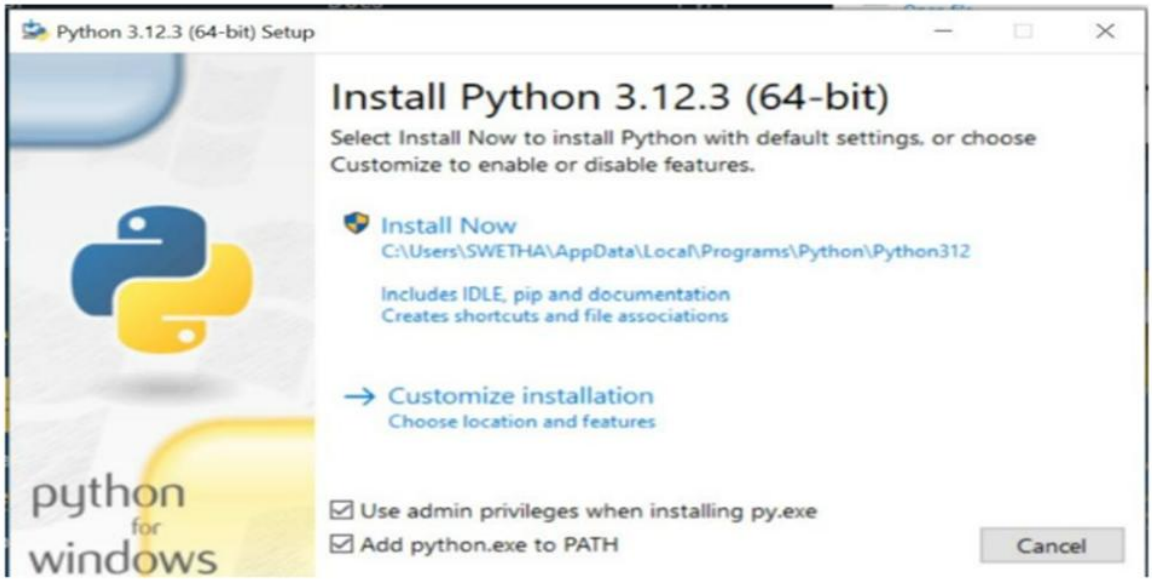
Installation of Python and Visual Studio code editors can be demonstrated.

- Python download Link:

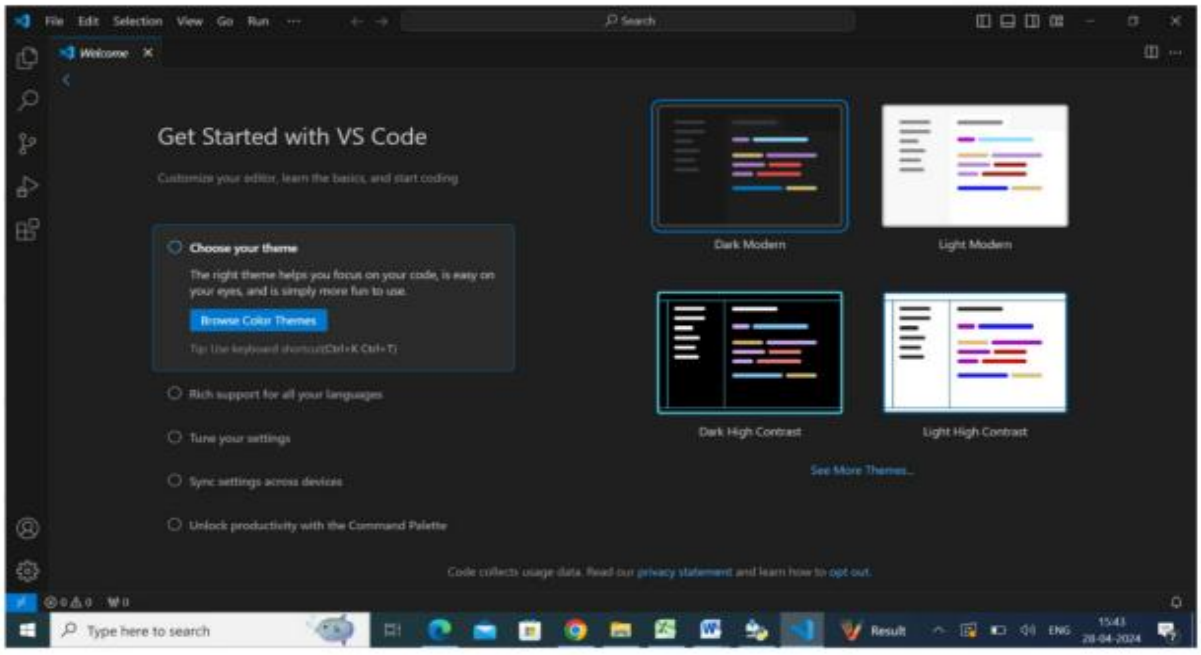
https://www.python.org/downloads/

- Visual Studio Code download and installation link:

https://code.visualstudio.com/




Setup - Microsoft Visual Studio Code (User)
Completing the Visual Studio Code Setup Wizard
Setup has finished installing Visual Studio Code on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch Visual Studio Code
Finish


Get Started with VS Code
Customize your editor, learn the basics, and start coding
Choose your theme
The right theme helps you focus on your code, is easy on your eyes, and is simply more fun to use.
Browse Color Themes
Rich support for all your languages
Tune your settings
Sync settings across devices
Unlock productivity with the Command Palette
Dark Modern
Light Modern
Dark High Contrast
Light High Contrast
See More Themes...
Code collects usage data. Read our privacy statement and learn how to opt out.
Type here to search

Creating system root folder, project root project folder, creating virtual env inside project root folder.

Create a system root folder with the name MIT_ISE in the file system (inside any preferred drive).

- Create a project root folder inside MIT_ISE with the name "hello_world" (assuming we are doing simple hello world program)
- Open cmd inside "hello_world"
- Create virtual env inside "hello_world" with the name "hello_world_venv"

python –m venv <name_of_virtual_env>

python –m venv hello_world_venv

Open project root folder in VS code

- Run "code ." in the cmd prompt to launch the project folder "hello_world" in VS code.

Or

- Launch VS code from the task bar, goto file menu navigate and open "hello_world"

Command palette (select your venv as python interpreter for your project root folder)

- In VS Code, open the Command Palette (View > Command Palette or (Ctrl+Shift+P)). Then select the Python: Select Interpreter command
- The command presents a list of available interpreters that VS Code can locate automatically. From the list, select the virtual environment in your project folder that starts with ./env or .\env:
- Select the virtual environment that is created "hello_world_venv". Look for recommended
- Open VS code terminal and install django framework.

pip install Django

- Check whether django installation is correct

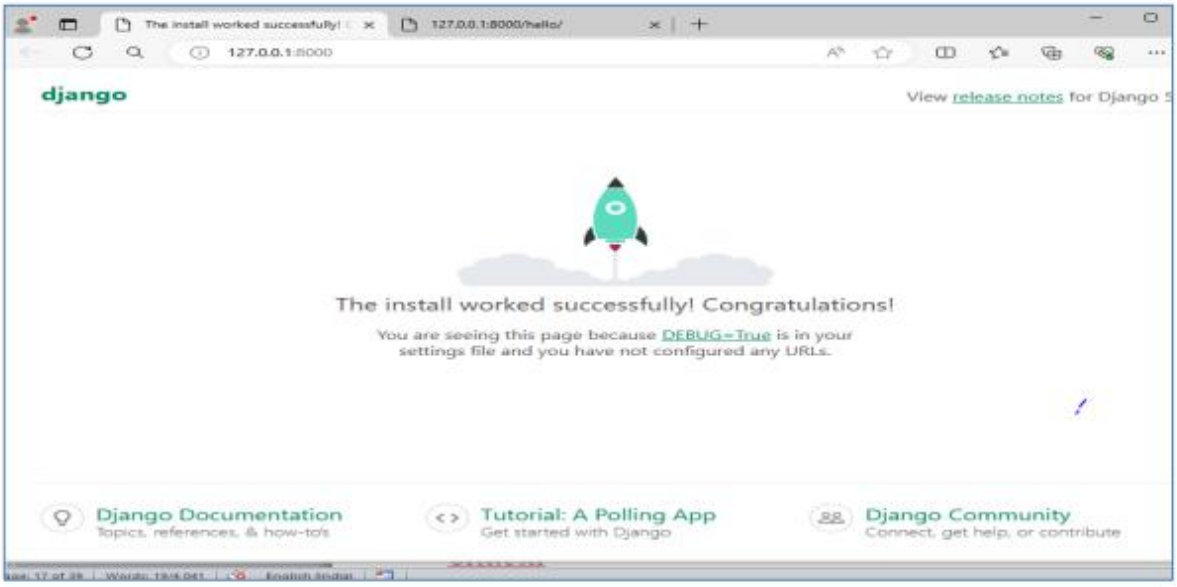
python manage.py runserver

- Create the django project with the name "hello_world_proj" inside the project root folder "hello_world"

django-admin startproject proj .

- Create the django app with the name "hello_world_app"

python manage.py startapp hello_world_app



Project Overview

The objective of this project is to develop a straightforward yet effective e-commerce website dedicated to the purchase of cosmetic products. Leveraging the robust capabilities of the Django framework, this website aims to deliver a seamless and intuitive platform for users to browse through a variety of cosmetic products, make selections, and manage their purchases with ease.

This project focuses on creating a user-centric experience where simplicity and functionality meet elegance. By implementing a clean and responsive design, the website will ensure that users have a smooth and engaging experience, regardless of the device they use—whether it be a desktop computer, tablet, or smartphone.

Key Goals:

1. **User-Friendly Interface:** The website will feature an easy-to-navigate interface, allowing users to effortlessly find and explore cosmetic products. The intuitive design will guide users through their shopping journey, from product discovery to purchase.

2. **Comprehensive Product Catalog:**Users will have access to a wide range of cosmetic products, each presented with detailed information including product name, brand, price, and high-quality images. This comprehensive catalog will help users make informed purchasing decisions.

3. Seamless Shopping Experience: The website will facilitate a seamless shopping experience by providing features such as a shopping cart and wishlist. Users can easily add products to their cart or wishlist from the product detail pages, manage their selections, and proceed to checkout with minimal effort.

4. Responsive Design: A key emphasis of the project is on creating a responsive design that adapts to various screen sizes and devices. This ensures that users enjoy a consistent and optimized browsing experience whether they are using a computer, tablet, or mobile phone.

5. **Efficient Product Management:** An admin panel will be included to allow administrators to efficiently manage the product catalog. This includes adding new products, updating existing ones, and removing outdated items. The admin panel will be designed to be user-friendly, making it easy for administrators to keep the product catalog up-to-date.

6. **Scalability and Performance:** The project will be designed with scalability in mind, ensuring that the website can handle increasing numbers of users and products without compromising on performance. This involves careful planning of the backend architecture and optimization of the database and server configurations.

7. Security and Privacy: Ensuring the security and privacy of user data is paramount. The website will implement robust security measures to protect user information, including secure user authentication, encrypted data transmission, and adherence to best practices in data handling.

By achieving these goals, the project aims to create a reliable and enjoyable online shopping destination for cosmetics, catering to the needs of both consumers and administrators. The combination of Django's powerful framework and modern web development techniques will result in a highly functional, scalable, and secure e-commerce platform.

Features:

1. Cosmetic Catalog:

- Display a List of Cosmetics:
  - This feature allows users to browse through a comprehensive list of available cosmetics. Each product listing will include essential details such as the product name, brand, price, and cover image.
- Product Detail Page:
  - On selecting a product, users are redirected to a detailed page providing in-depth information about the cosmetic item, including a high-resolution image, full description, and price. Users can also add the product to their cart or wishlist from this page.

2. Shopping Cart:

- Add to Cart:
  - Users can conveniently add products to their shopping cart directly from the product detail page. This functionality ensures that users can compile their desired items before proceeding to checkout.
- View Cart:
  - The shopping cart page displays all the items added by the user, along with their details and options to modify quantities. This page also includes a button to simulate the checkout process.

3. Wishlist:

- Add to Wishlist:
  - Similar to the shopping cart, users can add products to their wishlist from the product detail page. This feature allows users to save products for future consideration.
- View Wishlist:
  - The wishlist page lists all the items saved by the user, providing an easy way to view and manage potential future purchases.

4. Admin Panel:

- Manage Products
  - The admin panel provides a robust interface for managing the product catalog. Admin users can add new cosmetics, update existing product details, and remove products from the

catalog as needed. This feature leverages Django's built-in admin interface, customized for enhanced usability.

5. User Experience:

- Responsive Design:
  - The website is designed to be fully responsive, ensuring optimal user experience across a wide range of devices, from desktops to smartphones.

Technologies Used

- **Backend**: Django, Django REST Framework
- **Frontend**: HTML, CSS, JavaScript, Bootstrap
- **Database**: PostgreSQL

Step-by-Step Implementation

1. **Setting Up the Django Project:**
   - Install Django and create a new project:
     - Begin by installing Django and setting up a new project environment. This includes configuring essential settings and setting up the database connection.
   - Configure settings, including the database setup:
     - Modify the settings.py file to include necessary configurations such as database setup, installed apps, and middleware.

2. **Creating the Cosmetics Catalog App:**
   - Define Models for Product, Cart, and Wishlist:
     - Create models for Product, Cart, and Wishlist to structure the database. These models will represent the core data entities of the application.
   - Create Views to Display Product Details:
     - Develop views to handle the display logic for product listings and details. These views will fetch data from the database and render it using templates.
   - Develop Templates for Product Listings and Detail Pages:
     - Create HTML templates for product listings and detail pages. Use Django's template language to dynamically display product information.

3. **Admin Panel Customization**
   - Manage Products and Users:
     - Utilize Django's admin interface to facilitate product and user management. Customize the admin panel to improve usability and streamline the management process.

4. **Deployment:**
   - Prepare the project for deployment:
     - Ensure the project is production-ready by configuring settings for security, performance, and scalability.
   - Choose a hosting platform and set up the environment:
     - Select a hosting platform suitable for Django applications and set up the environment for deployment.
   - Deploy the application and configure the domain:
     - Deploy the application to the chosen platform and configure the domain to make the website accessible to users

Code Overview:

#cos_app(app)

1. admin.py

```
from django.contrib import admin
from .models import Product, Cart, Wishlist

class ProductAdmin(admin.ModelAdmin):
    list_display = ('name', 'price', 'stock', 'image')
    search_fields = ('name',)

admin.site.register(Product)
admin.site.register(Cart)
admin.site.register(Wishlist)
```

2. app.py

```
from django.apps import AppConfig

class CosAppConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'cos_app'
```

3. views.py

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import Product, Cart, Wishlist
from django.contrib.auth.decorators import login_required


def home(request):
    products = Product.objects.all()
    return render(request, 'home.html', {'products': products})


def product_details(request, product_id):
    product = get_object_or_404(Product, id=product_id)
    return render(request, 'product_details.html', {'product': product})


def add_to_cart(request, product_id):
    product = get_object_or_404(Product, id=product_id)
    cart_item, created = Cart.objects.get_or_create(user=request.user, product=product)
    if not created:
        cart_item.quantity += 1
        cart_item.save()
    return redirect('cart')


def add_to_wishlist(request, product_id):
    product = get_object_or_404(Product, id=product_id)
    Wishlist.objects.get_or_create(user=request.user, product=product)
```

```
    return redirect('wishlist')


def cart(request):
    cart_items = Cart.objects.filter(user=request.user)
    return render(request, 'cart.html', {'cart_items': cart_items})


def wishlist(request):
    wishlist_items = Wishlist.objects.filter(user=request.user)
    return render(request, 'wishlist.html', {'wishlist_items': wishlist_items})
```

4. models.py

```
from django.db import models
from django.contrib.auth.models import User
class Product(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    stock = models.IntegerField()
    image = models.ImageField(upload_to='products/')
    def __str__(self):
        return self.name


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 f"{self.user.username}'s cart - {self.product.name}"

class Wishlist(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    def __str__(self):
        return f"{self.user.username}'s wishlist - {self.product.name}"
```

#cos_proj(project)

1. asgi.py

```
import os
from django.core.asgi import get_asgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'cos_proj.settings')
application = get_asgi_application()
```

2. settings.py

```
import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
SECRET_KEY = 'django-insecure-4sj14(vga6+m%0x@eq5pf#ndkarsk1966i___l@%5mi_y%+*r_'
DEBUG = True
```

```
ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'cos_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 = 'cos_proj.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',
            ],
        },
    },
]

WSGI_APPLICATION = 'cos_proj.wsgi.application'

# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

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

# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

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'
USE_I18N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

3. urls.py

```
from django.conf import settings
from django.conf.urls.static import static
from django.urls import path
from django.contrib import admin

from cos_app.views import add_to_cart, add_to_wishlist, cart, home, product_details, wishlist

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', home, name='home'),
    path('product/<int:product_id>/', product_details, name='product_details'),
    path('add-to-cart/<int:product_id>/', add_to_cart, name='add_to_cart'),
    path('add-to-wishlist/<int:product_id>/', add_to_wishlist, name='add_to_wishlist'),
```

```
    path('cart/', cart, name='cart'),
    path('wishlist/', wishlist, name='wishlist'),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

TEMPLATES:

base.html

```
<!DOCTYPE html>
<html>
<head>
    <title>{% block title %}AA shops{% endblock %}</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .carousel-item img {
            height: 300px;
            object-fit: contain;
        }
        .carousel-caption {
            background: rgba(0, 0, 0, 0.5);
            padding: 1rem;
        }
        .btn.btn-primary {
            background-color: aqua;
```

```
            color: black;
            border-color: azure;
        }
        .product-card {
            height: 380px;
            width: 300px;
            margin: auto;
            object-fit: contain;
        }
        .product-card img {
            height: 150px;
            margin-top: 6%;
            object-fit: contain;
        }
        .card-body {
            text-align: center;
            object-fit: contain;
        }
    </style>
</head>
<body>{% load static %}
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="#">AA shops</a>
        <div class="collapse navbar-collapse" id="navbarNav">
```

```
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ml-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'home' %}">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'cart' %}">Cart</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'wishlist' %}">Wishlist</a>
                    </li>
                </ul>
            </div>
        </nav>

    </div>
</nav>
<div class="container mt-4">
    {% block content %}
    {% endblock %}
```

```
    </div>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

home.html

```
{% extends 'base.html' %}
{% block title %}Home - AA shops{% endblock %}
{% block content %}
{% load static %}
    <h1>Welcome to AA shops</h1>
    <div id="productCarousel" class="carousel slide" data-ride="carousel">
        <div class="carousel-inner">
            {% for product in products %}
                <div class="carousel-item{% if forloop.first %} active{% endif %}">
                    <img src="{{ product.image.url }}" alt="{{ product.name }}" class="d-block w-100">
                    <div class="carousel-caption d-none d-md-block">
                        <h5 class="card-title">{{ product.name }}</h5>
                        <!--<p class="card-text">{{ product.description }}</p>-->
                        <a href="{% url 'product_details' product.id %}" class="btn btn-primary">View
Details</a>
                    </div>
                </div>
```

```
            {% endfor %}
        </div>
        <a class="carousel-control-prev" href="#productCarousel" role="button" data-slide="prev">
            <span class="carousel-control-prev-icon" aria-hidden="true"></span>
            <span class="sr-only">Previous</span>
        </a>
        <a class="carousel-control-next" href="#productCarousel" role="button" data-slide="next">
            <span class="carousel-control-next-icon" aria-hidden="true"></span>
            <span class="sr-only">Next</span>
        </a>
    </div>

    <div class="row mt-4">
        {% for product in products %}
            <div class="col-md-4">
                <div class="card product-card" style="height: 400px;">
                    <img src="{{ product.image.url }}" alt="{{ product.name }}">
                    <div class="card-body">
                        <h5 class="card-title">{{ product.name }}</h5>
                    <!--<p class="card-text">{{ product.description }}</p>-->
                        <a href="{% url 'product_details' product.id %}" class="btn btn-primary">View
Details</a>
                    </div>
                </div><br>
            </div>
```

```
        {% endfor %}
    </div>
{% endblock %}.
```

product_details.html

```
{% extends 'base.html' %}
{%block title%}{{ product.name }} - AA shops {%endblock%}
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
{%block content%}{%load static%}
    <div class="container mt-4">
        <div class="row">
            <div class="col-md-6">
                <img src="{{ product.image.url }}" alt="{{ product.name }}" class="img-fluid">
            </div>
            <div class="col-md-6">
                <h1>{{ product.name }}</h1>
                <p>{{ product.description }}</p>
                <h4>${{ product.price }}</h4>
                <p>{{ product.stock }} in stock</p>
                <a href="{% url 'add_to_cart' product.id %}" class="btn btn-primary">Add to Cart</a>
                <a href="{% url 'add_to_wishlist' product.id %}" class="btn btn-secondary">Add to
Wishlist</a>
            </div>
```

```
        </div>
    </div>
    {%endblock%}
```

cart.html

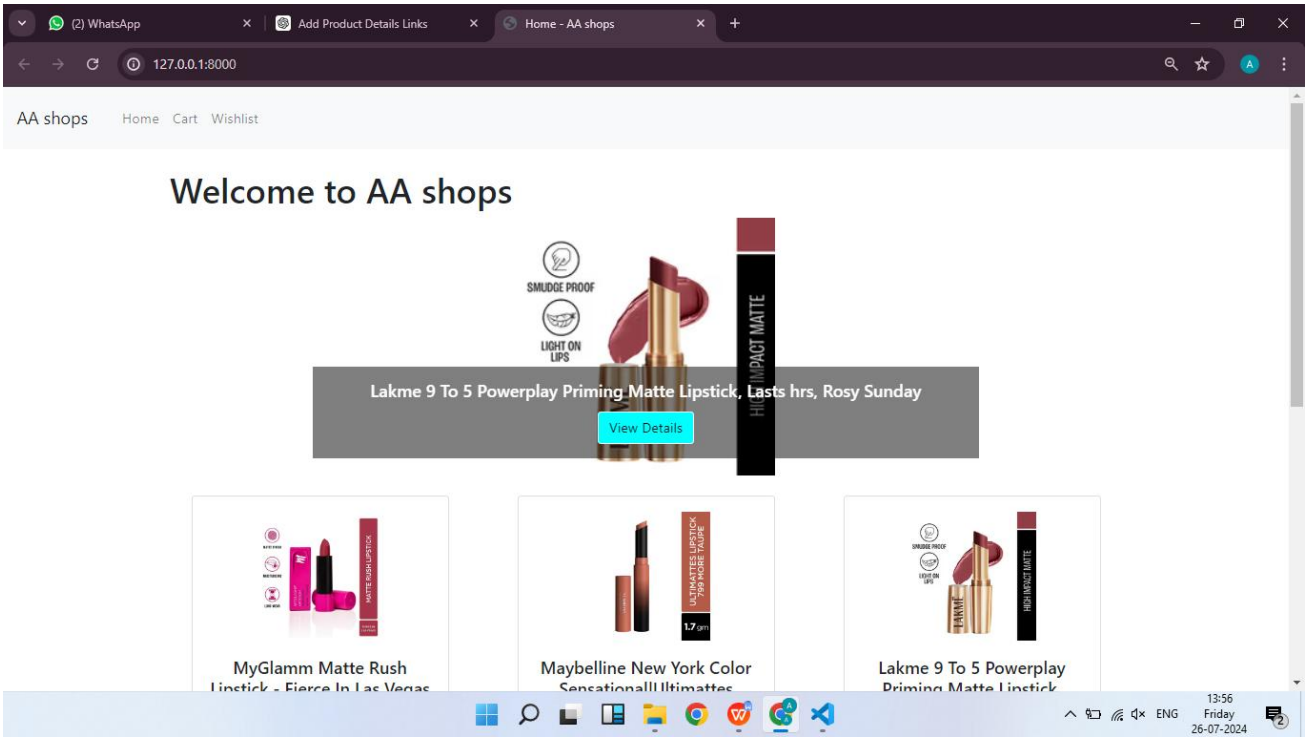
```
{%extends 'base.html' %}
<head>
    {%block title%}Cart - AA shops {%endblock%}
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
    {%block content%}{%load static%}
    <div class="container mt-4">
        <h1>Your Cart</h1>
        <ul class="list-group">
            {% for item in cart_items %}
                <li class="list-group-item">
                    <img src="{{ item.product.image.url }}" alt="{{ item.product.name }}" width="200">
                    {{ item.product.name }} - ${{ item.product.price }} x {{ item.quantity }}
                </li>
            {% endfor %}
        </ul>
        <a href="#" class="btn btn-success mt-3">Proceed to Checkout</a>
    </div>
```

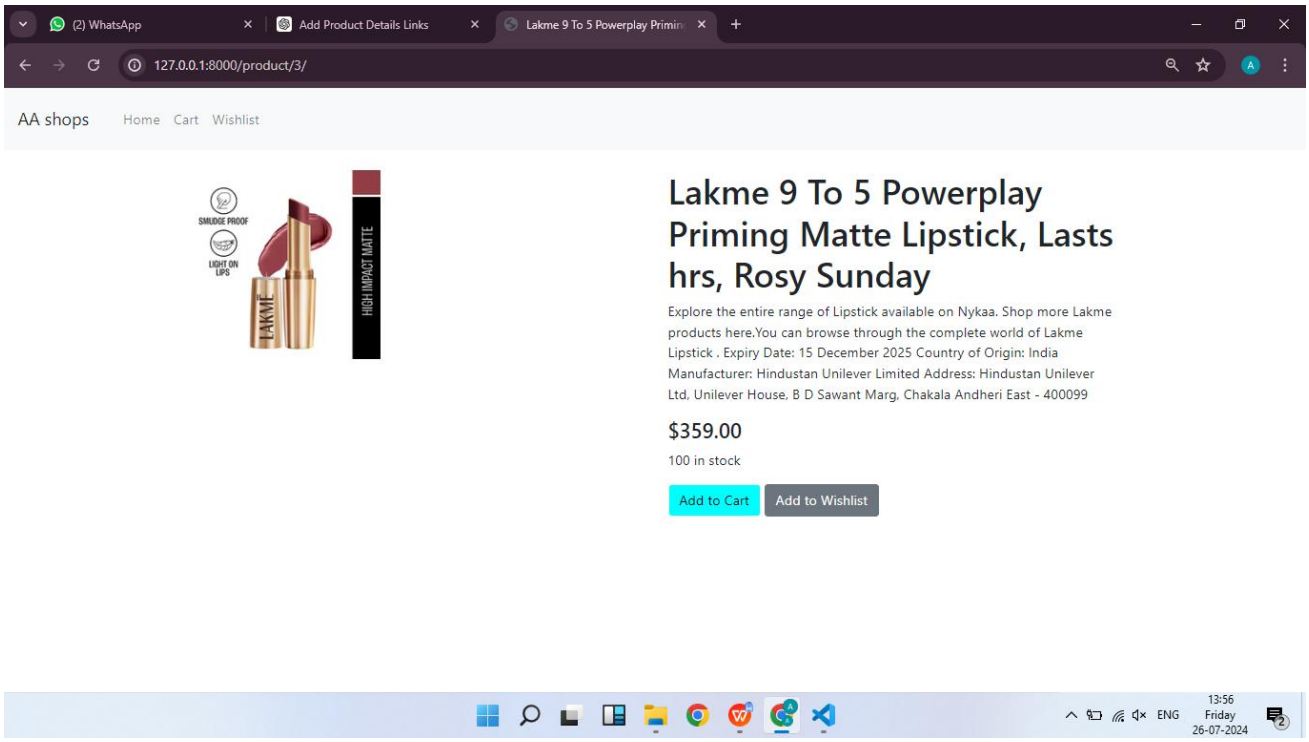
```
    {%endblock%}
```

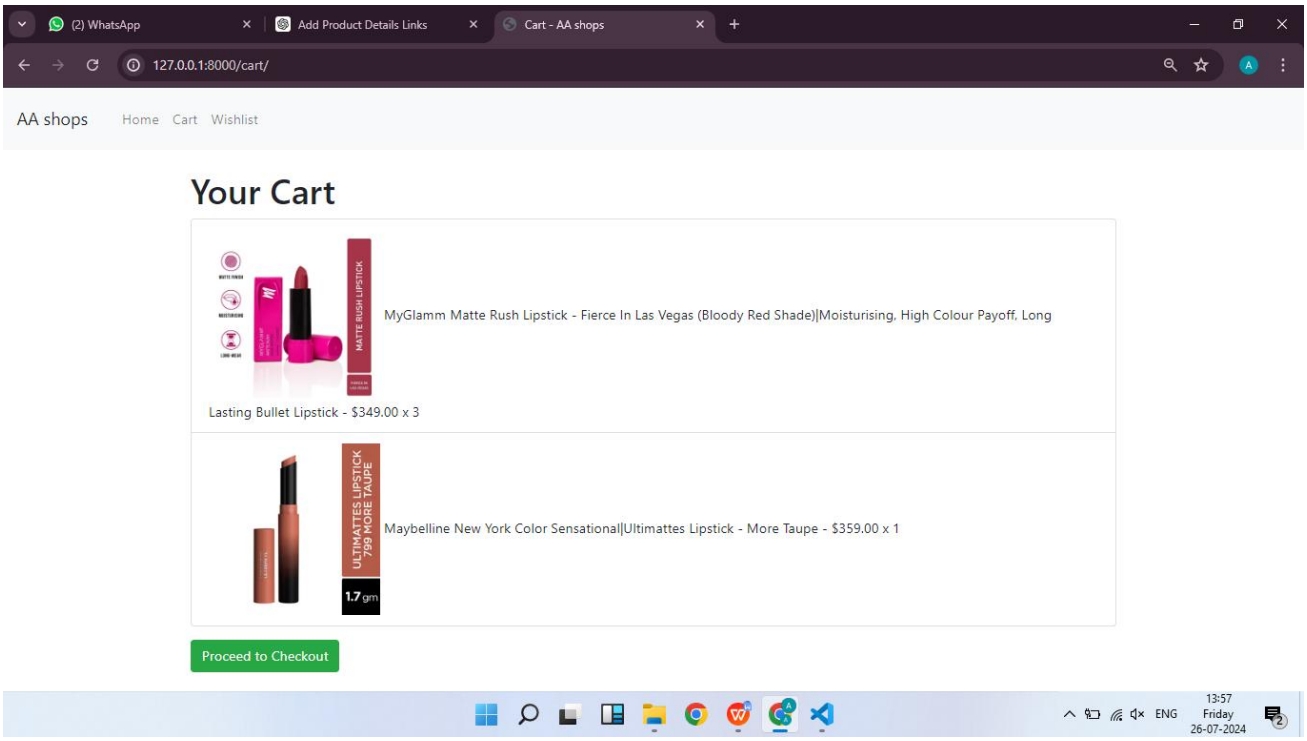
wishlist.html

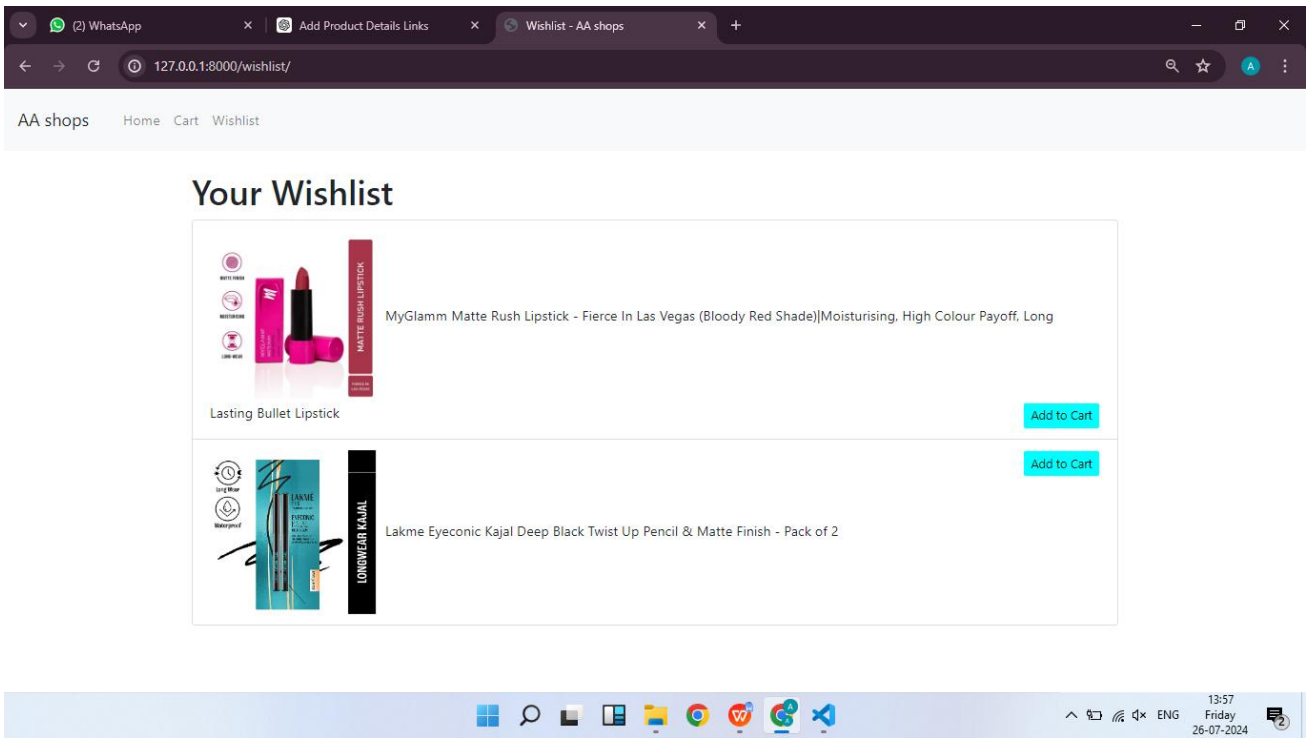
```
{% extends 'base.html' %}
<head>
    {%block title%}Wishlist - AA shops {%endblock%}
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
{%block content%}{% load static %}
    <div class="container mt-4">
        <h1>Your Wishlist</h1>
        <ul class="list-group">
            {% for item in wishlist_items %}
                <li class="list-group-item">
                    <img src="{{ item.product.image.url }}" alt="{{ item.product.name }}" width="200">
                    {{ item.product.name }}
                    <a href="{% url 'add_to_cart' item.product.id %}" class="btn btn-primary btn-sm float-
right">Add to Cart</a>
                </li>
            {% endfor %}
        </ul>
    </div>
    {%endblock%}
```

Snapshots:


(2) WhatsApp
Add Product Details Links
Home - AA shops
127.0.0.1:8000
AA shops
Home Cart Wishlist
Welcome to AA shops
SMUDGE PROOF
LIGHT ON LIPS
HIGH IMPACT MATTE
Lakme 9 To 5 Powerplay Priming Matte Lipstick, Lasts hrs, Rosy Sunday
View Details
MyGlamm Matte Rush
Maybelline New York Color
Lakme 9 To 5 Powerplay
ENG
13:56
Friday
26-07-2024


(2) WhatsApp
Add Product Details Links
Lakme 9 To 5 Powerplay Primin
127.0.0.1:8000/product/3/
AA shops
Home Cart Wishlist
SMUDGE PROOF
LIGHT ON LIPS
LAKME
HIGH IMPACT MATTE
Lakme 9 To 5 Powerplay Priming Matte Lipstick, Lasts hrs, Rosy Sunday
Explore the entire range of Lipstick available on Nykaa. Shop more Lakme products here.You can browse through the complete world of Lakme Lipstick . Expiry Date: 15 December 2025 Country of Origin: India Manufacturer: Hindustan Unilever Limited Address: Hindustan Unilever Ltd, Unilever House, B D Sawant Marg, Chakala Andheri East - 400099
$359.00
100 in stock
Add to Cart
Add to Wishlist
ENG
13:56
Friday
26-07-2024


(2) WhatsApp
Add Product Details Links
Cart - AA shops
127.0.0.1:8000/cart/
AA shops
Home Cart Wishlist
Your Cart
MyGlamm Matte Rush Lipstick - Fierce In Las Vegas (Bloody Red Shade)|Moisturising, High Colour Payoff, Long Lasting Bullet Lipstick - $349.00 x 3
Maybelline New York Color Sensational|Ultimattes Lipstick - More Taupe - $359.00 x 1
Proceed to Checkout
ENG
13:57
Friday
26-07-2024


(2) WhatsApp
Add Product Details Links
Wishlist - AA shops
127.0.0.1:8000/wishlist/
AA shops
Home Cart Wishlist
Your Wishlist
MyGlamm Matte Rush Lipstick - Fierce In Las Vegas (Bloody Red Shade)|Moisturising, High Colour Payoff, Long Lasting Bullet Lipstick
Add to Cart
Add to Cart
Lakme Eyeconic Kajal Deep Black Twist Up Pencil & Matte Finish - Pack of 2
ENG
13:57
Friday
26-07-2024

CONCLUSION:

Building an e-commerce book buying website using Django involves integrating various components such as user authentication, book catalog management, and payment processing. By leveraging Django's robust features and following best practices in web development, you can create a scalable, secure, and user-friendly e-commerce platform. Additionally, focusing on user experience through responsive design, implementing comprehensive search and filter options, and providing personalized recommendations can significantly enhance customer satisfaction and engagement. Ensuring data security, optimizing for performance, and incorporating analytics to track user behavior are also crucial for maintaining a competitive edge and fostering continuous improvement of the platform. Moreover, integrating social media sharing options, offering multiple payment gateways, and providing exceptional customer service can further elevate the user experience, making the platform not only a place to buy books but a trusted and enjoyable shopping destination. This holistic approach, combining technical excellence with a deep understanding of user needs, will ensure the website's success in the competitive e-commerce landscape.