Ecommerce -shofy

Session: 2021 - 2025

Submitted by:

Naseeb Amjad

2021-CS-165

Supervised by:

Muhammad Laeeq Uz Zaman Khan Niazi

Department of Computer Science

University of Engineering and Technology Lahore

Pakistan

Contents

1 Introduction 3
2 Objectives 3
3 Project Features 3
 3.1 User Authentication 3
 3.2 Product browsing 3
 3.3 sorting 3
 3.4 Add to cart 3
 3.5 Wishlist management 4
 3.6 Cart management 4
 3.7 Order placement 4
4 Project Diagrams 4
5 Best Practices 9
6 Project Management Tool..10
7 Project Management...10
8 Code Snippets...10
9 Screens Layout along with navigation flow..14
10 Target Audience...15

1 Introduction

Welcome to ShopEase, your go-to e-commerce shopping app! With Shofy, enjoy effortless browsing through a variety of top-brand products. Our user-friendly interface ensures a smooth experience, while personalized recommendations cater to your unique preferences. Feel secure with protected transactions and track your orders in real-time. Benefit from exclusive deals, create wishlists, and access efficient customer support—all in one app. Shop with ease, style, and convenience.

2 Objectives

The primary objectives of Shofy include enhancing user engagement through personalized features, acquiring and retaining a loyal user base with effective marketing strategies, ensuring a seamless and secure shopping experience, and optimizing conversion rates through streamlined processes. The app aims to build trust through robust security measures while continuously seeking user feedback for ongoing improvements.

3 Project Features

3.1 User Authentication (Login/Signup):

Users start their journey by authenticating through a secure login or signup process, ensuring a personalized and secure experience.

3.2 Product Browsing and Viewing:

Explore a vast array of products with an intuitive product viewer, offering detailed information, images, and specifications for informed decision-making..

3.3 Sorting Options (High to Low):

Enhance the shopping experience by allowing users to sort products based on price, enabling them to prioritize items based on their budget and preferences..

3.4 Add to Cart:

Seamlessly add desired products to the shopping cart, providing a centralized location for users to review and manage their selected items before proceeding to checkout.

3.5 Wishlist Management:

Enable users to create and manage wishlists, allowing them to save and track favorite products for future consideration or sharing with others.

3.6 Cart Management:

Effortlessly manage items in the cart, giving users the flexibility to review, modify, or remove products before finalizing their purchase.

3.7 Order Placement:

Facilitate a secure and straightforward order placement process, guiding users through the necessary steps to complete their transaction.

4 Project Diagram

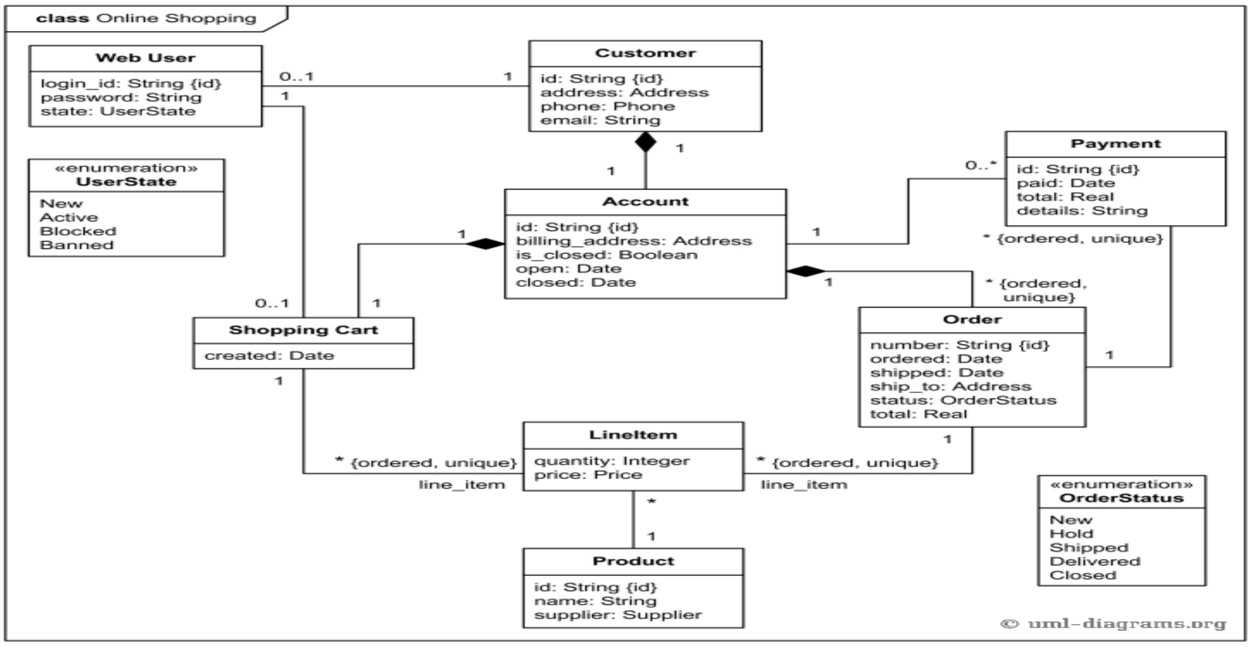


FIGURE 1: Class Diagram of the Project

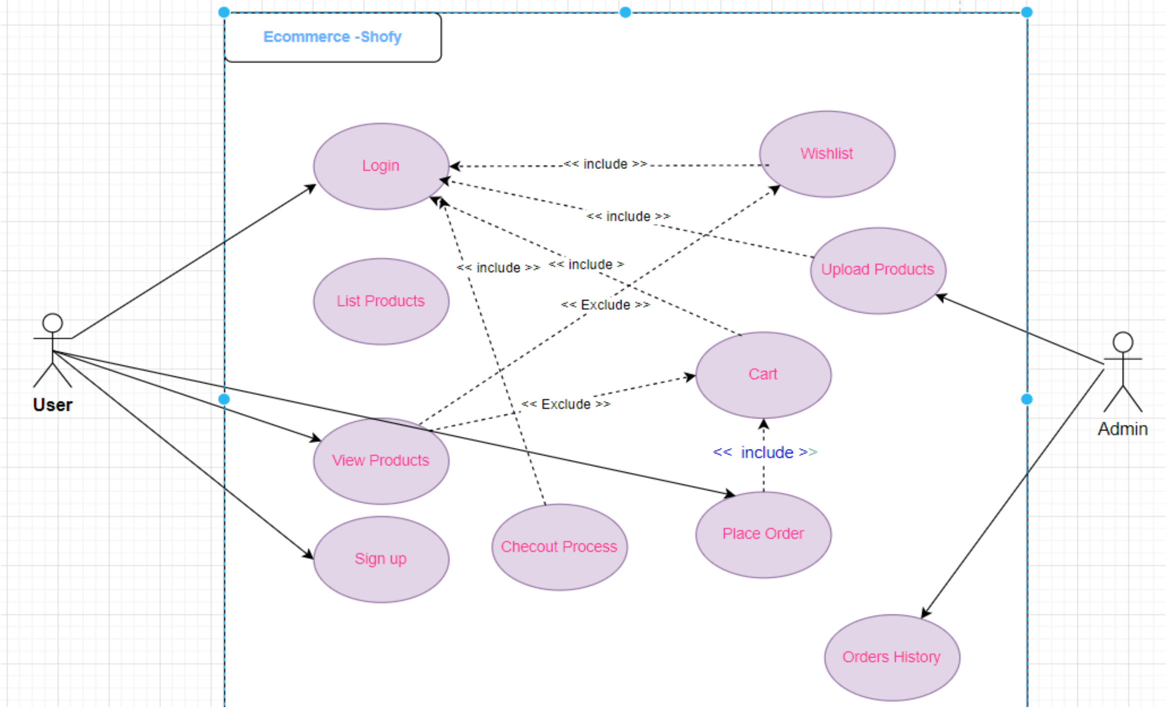


FIGURE 2: Use Case Diagram of the Project

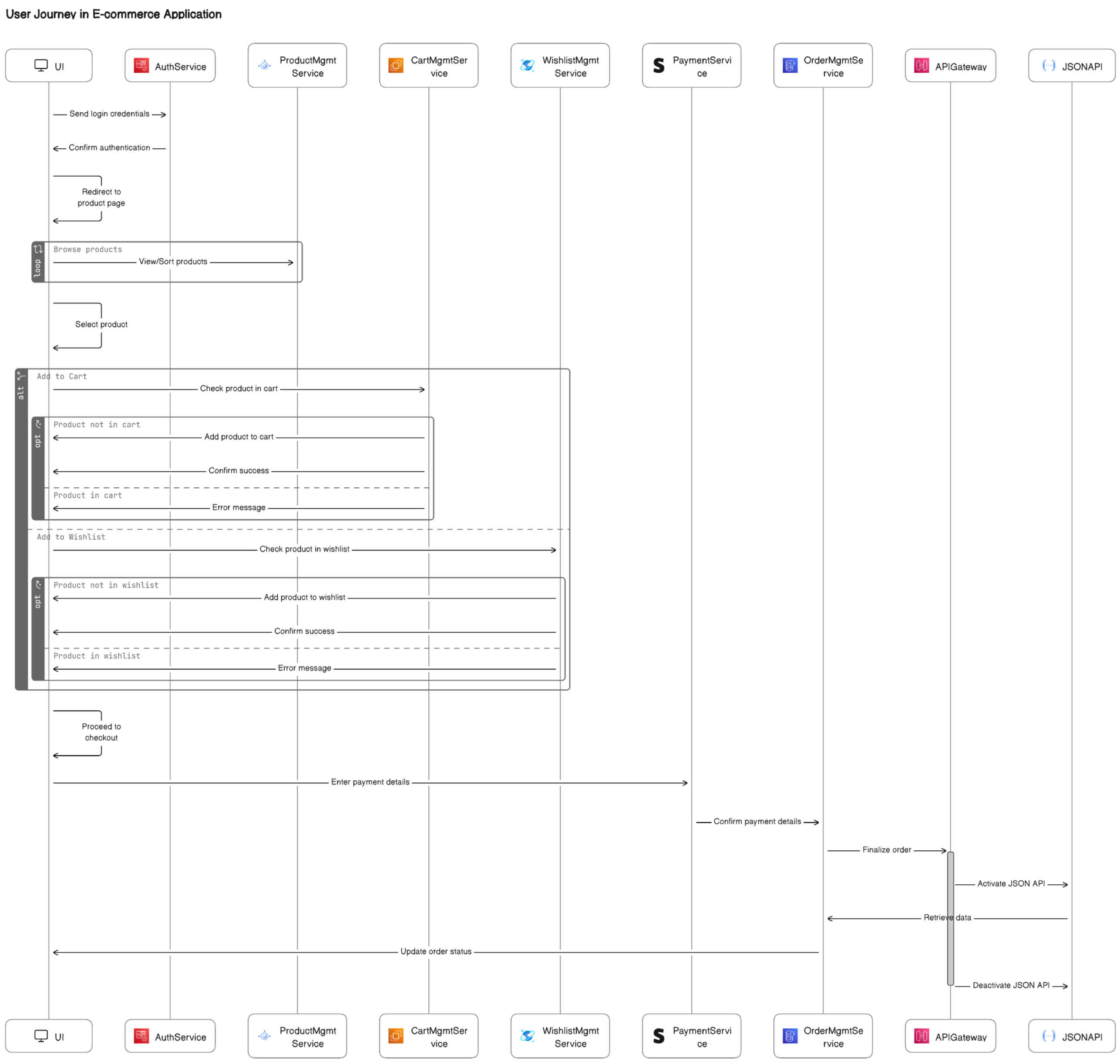


FIGURE 3: Sequence Diagram of the Project

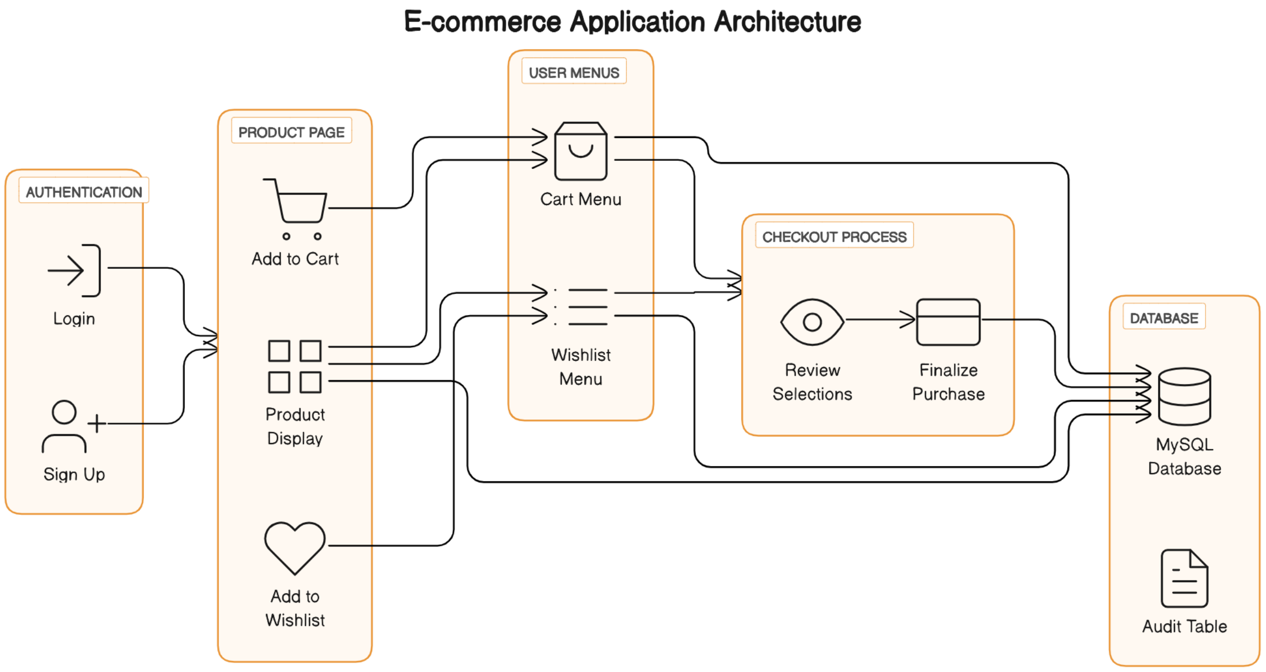


FIGURE 4: Architecture Diagram of the Project

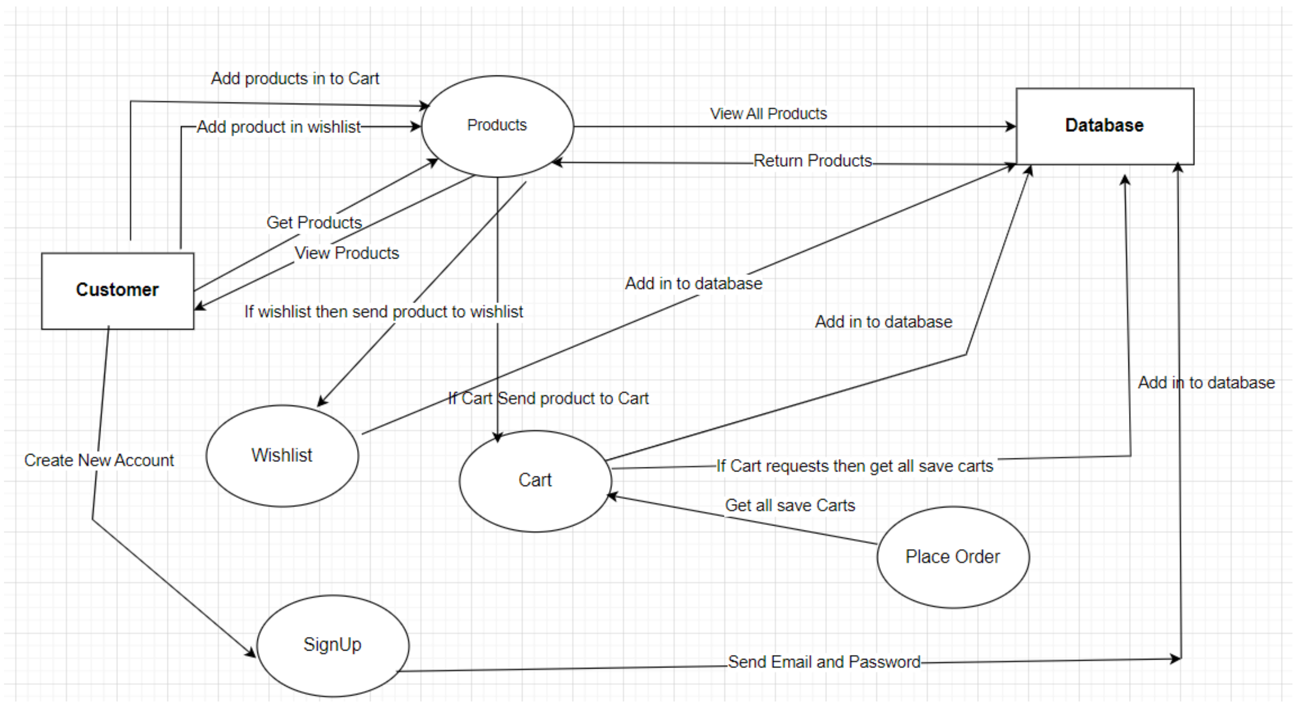


Figure 5 : Data flow diagram level 0

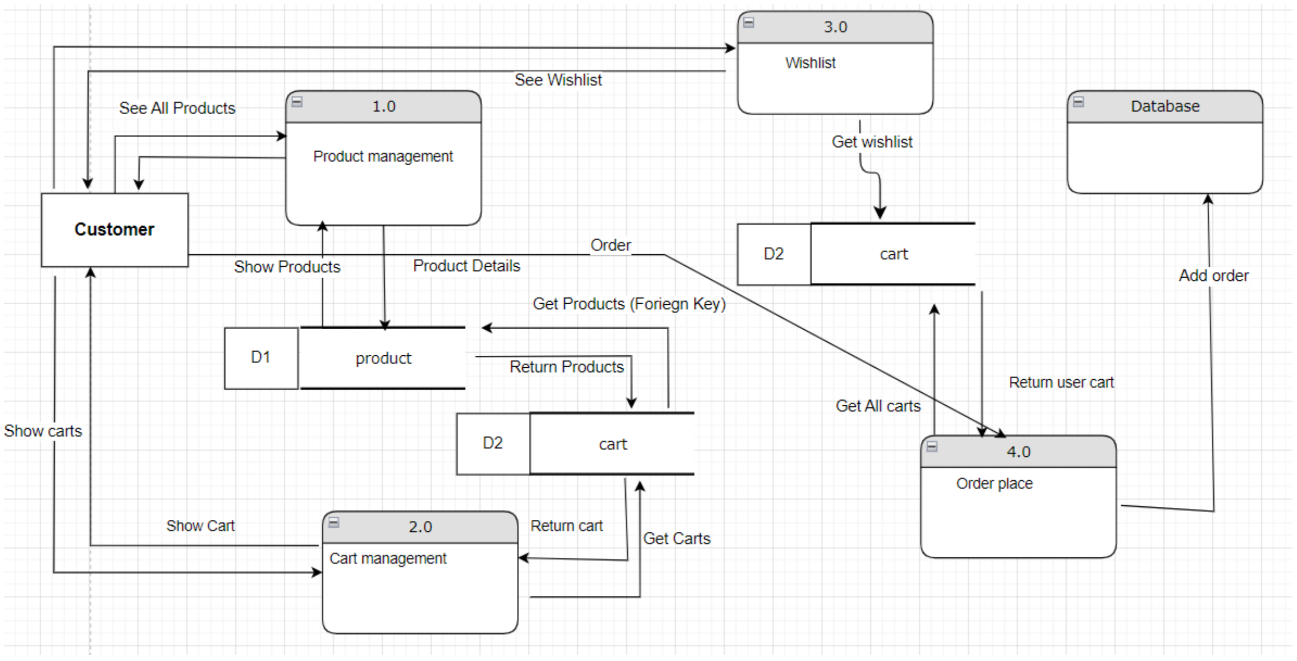
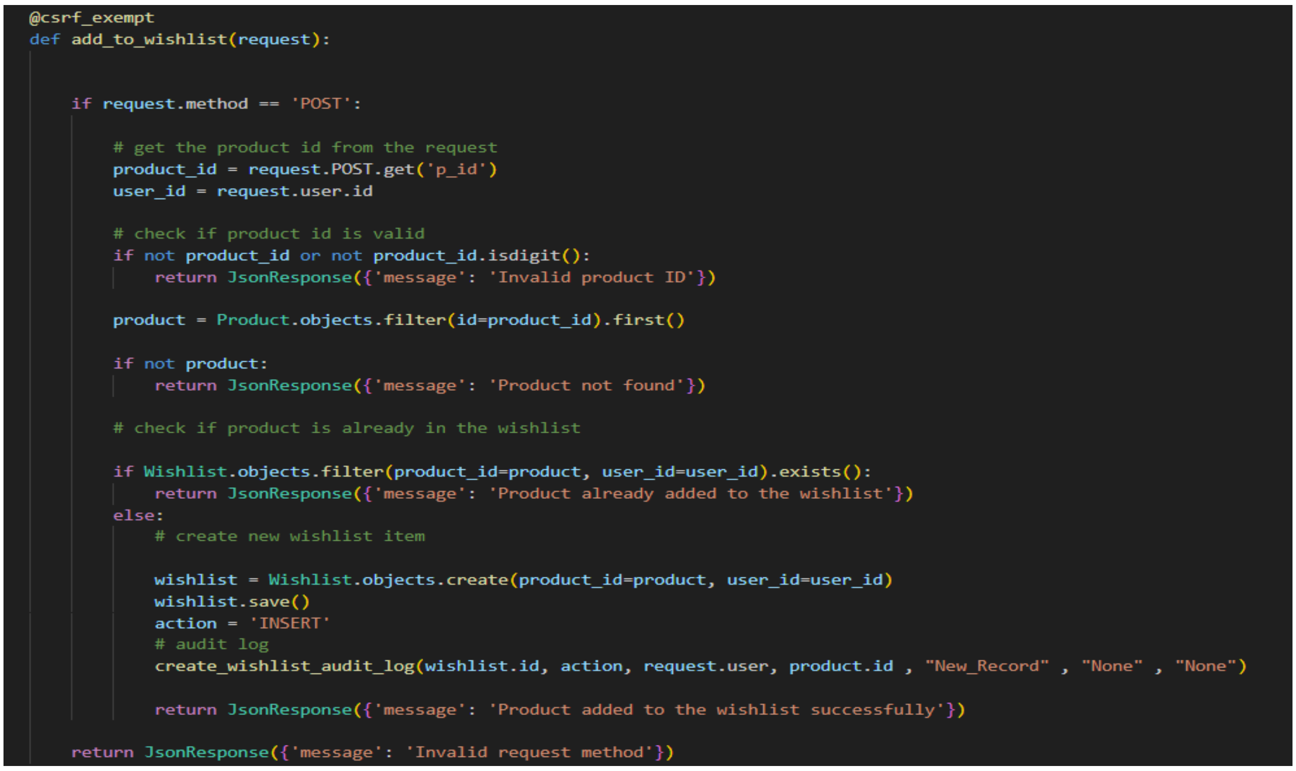


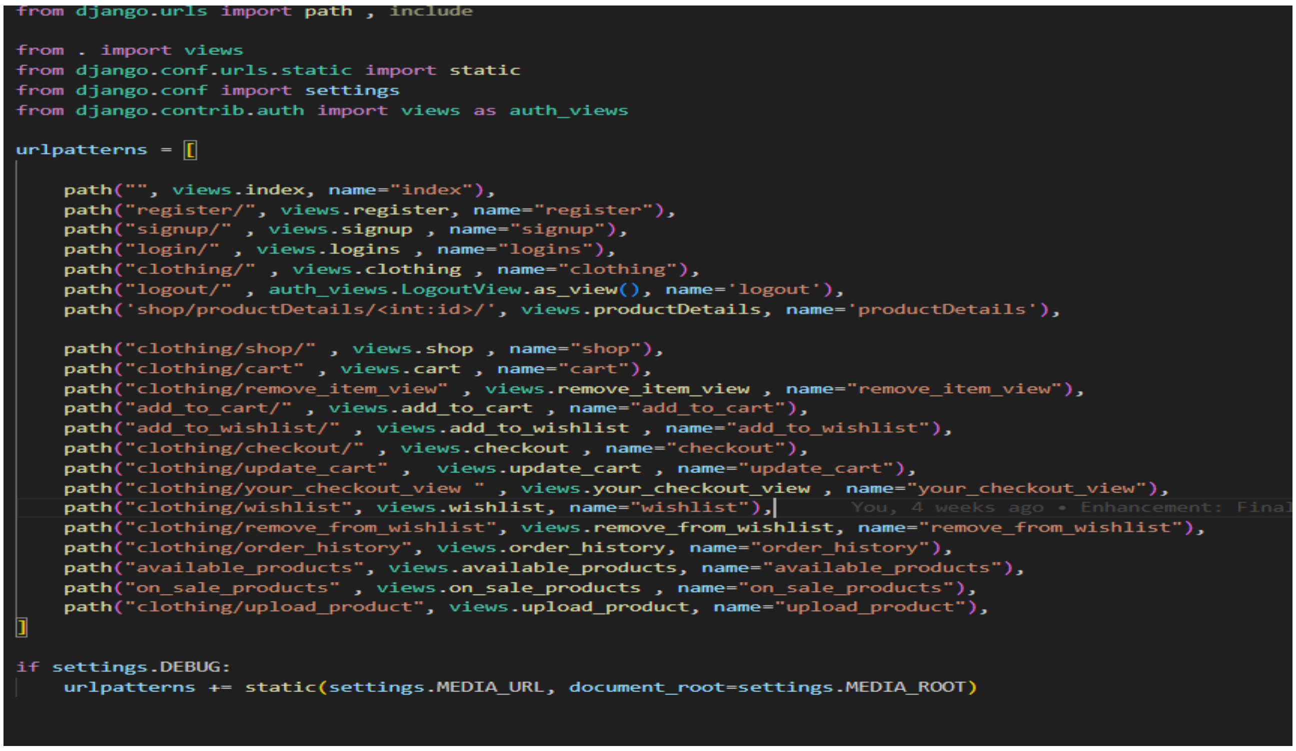
FIGURE 6 : Data flow diagram level 1

5 Best Practices



```python
@csrf_exempt
def add_to_wishlist(request):


    if request.method == 'POST':

        # get the product id from the request
        product_id = request.POST.get('p_id')
        user_id = request.user.id

        # check if product id is valid
        if not product_id or not product_id.isdigit():
            return JsonResponse({'message': 'Invalid product ID'})

        product = Product.objects.filter(id=product_id).first()

        if not product:
            return JsonResponse({'message': 'Product not found'})

        # check if product is already in the wishlist

        if Wishlist.objects.filter(product_id=product, user_id=user_id).exists():
            return JsonResponse({'message': 'Product already added to the wishlist'})
        else:
            # create new wishlist item

            wishlist = Wishlist.objects.create(product_id=product, user_id=user_id)
            wishlist.save()
            action = 'INSERT'
            # audit log
            create_wishlist_audit_log(wishlist.id, action, request.user, product.id , "New_Record" , "None" , "None")

            return JsonResponse({'message': 'Product added to the wishlist successfully'})

    return JsonResponse({'message': 'Invalid request method'})
```

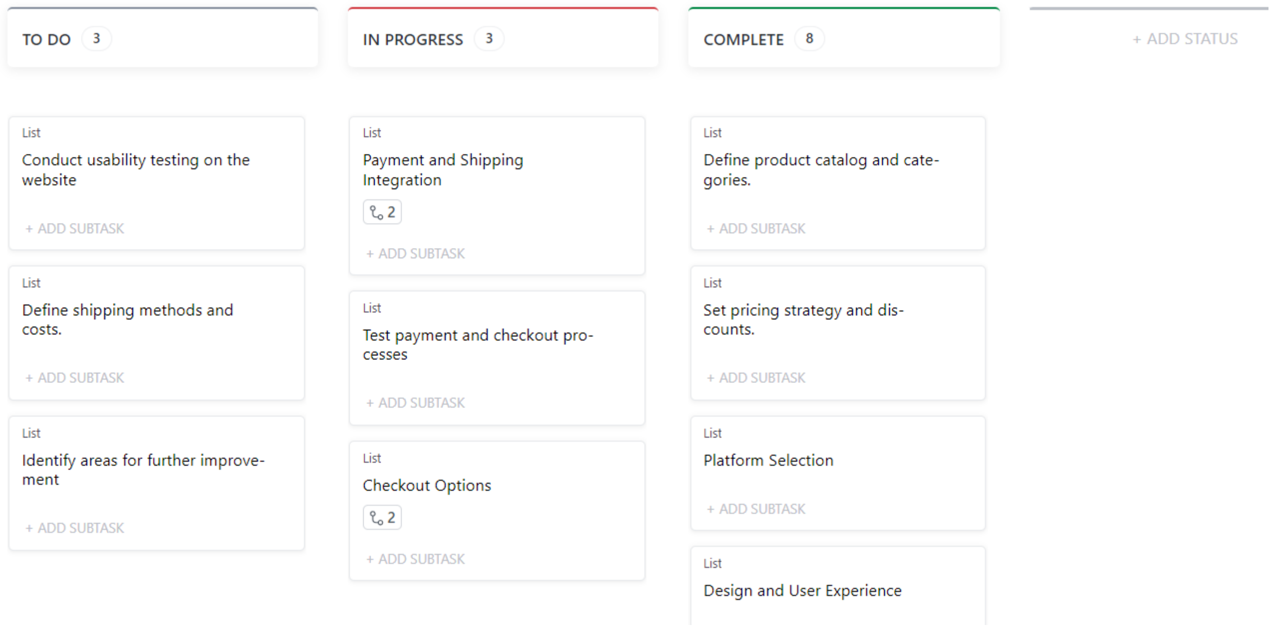
FIGURE 6: Comments as Best Practices



```python
from django.urls import path , include

from . import views
from django.conf.urls.static import static
from django.conf import settings
from django.contrib.auth import views as auth_views

urlpatterns = [

    path("", views.index, name="index"),
    path("register/", views.register, name="register"),
    path("signup/" , views.signup , name="signup"),
    path("login/" , views.logins , name="logins"),
    path("clothing/" , views.clothing , name="clothing"),
    path("logout/" , auth_views.LogoutView.as_view(), name='logout'),
    path('shop/productDetails/<int:id>/', views.productDetails, name='productDetails'),

    path("clothing/shop/" , views.shop , name="shop"),
    path("clothing/cart" , views.cart , name="cart"),
    path("clothing/remove_item_view" , views.remove_item_view , name="remove_item_view"),
    path("add_to_cart/" , views.add_to_cart , name="add_to_cart"),
    path("add_to_wishlist/" , views.add_to_wishlist , name="add_to_wishlist"),
    path("clothing/checkout/" , views.checkout , name="checkout"),
    path("clothing/update_cart" ,  views.update_cart , name="update_cart"),
    path("clothing/your_checkout_view " , views.your_checkout_view , name="your_checkout_view"),
    path("clothing/wishlist", views.wishlist, name="wishlist"),      You, 4 weeks ago • Enhancement: Final
    path("clothing/remove_from_wishlist", views.remove_from_wishlist, name="remove_from_wishlist"),
    path("clothing/order_history", views.order_history, name="order_history"),
    path("available_products", views.available_products, name="available_products"),
    path("on_sale_products" , views.on_sale_products , name="on_sale_products"),
    path("clothing/upload_product", views.upload_product, name="upload_product"),
]

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

FIGURE 7: API Naming Best Practices

6 Project Management Tool



TO DO 3

List
Conduct usability testing on the website
+ ADD SUBTASK

List
Define shipping methods and costs.
+ ADD SUBTASK

List
Identify areas for further improvement
+ ADD SUBTASK

IN PROGRESS 3

List
Payment and Shipping Integration
2
+ ADD SUBTASK

List
Test payment and checkout processes
+ ADD SUBTASK

List
Checkout Options
2
+ ADD SUBTASK

COMPLETE 8

List
Define product catalog and categories.
+ ADD SUBTASK

List
Set pricing strategy and discounts.
+ ADD SUBTASK

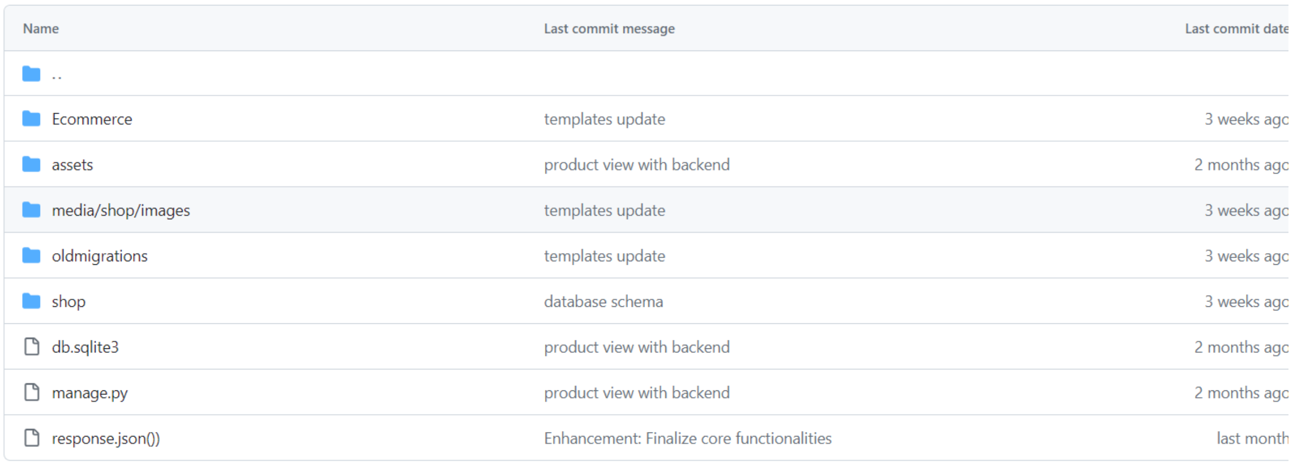
List
Platform Selection
+ ADD SUBTASK

List
Design and User Experience

+ ADD STATUS

FIGURE 8: Utilizing ClickUp as a Project Management Tool

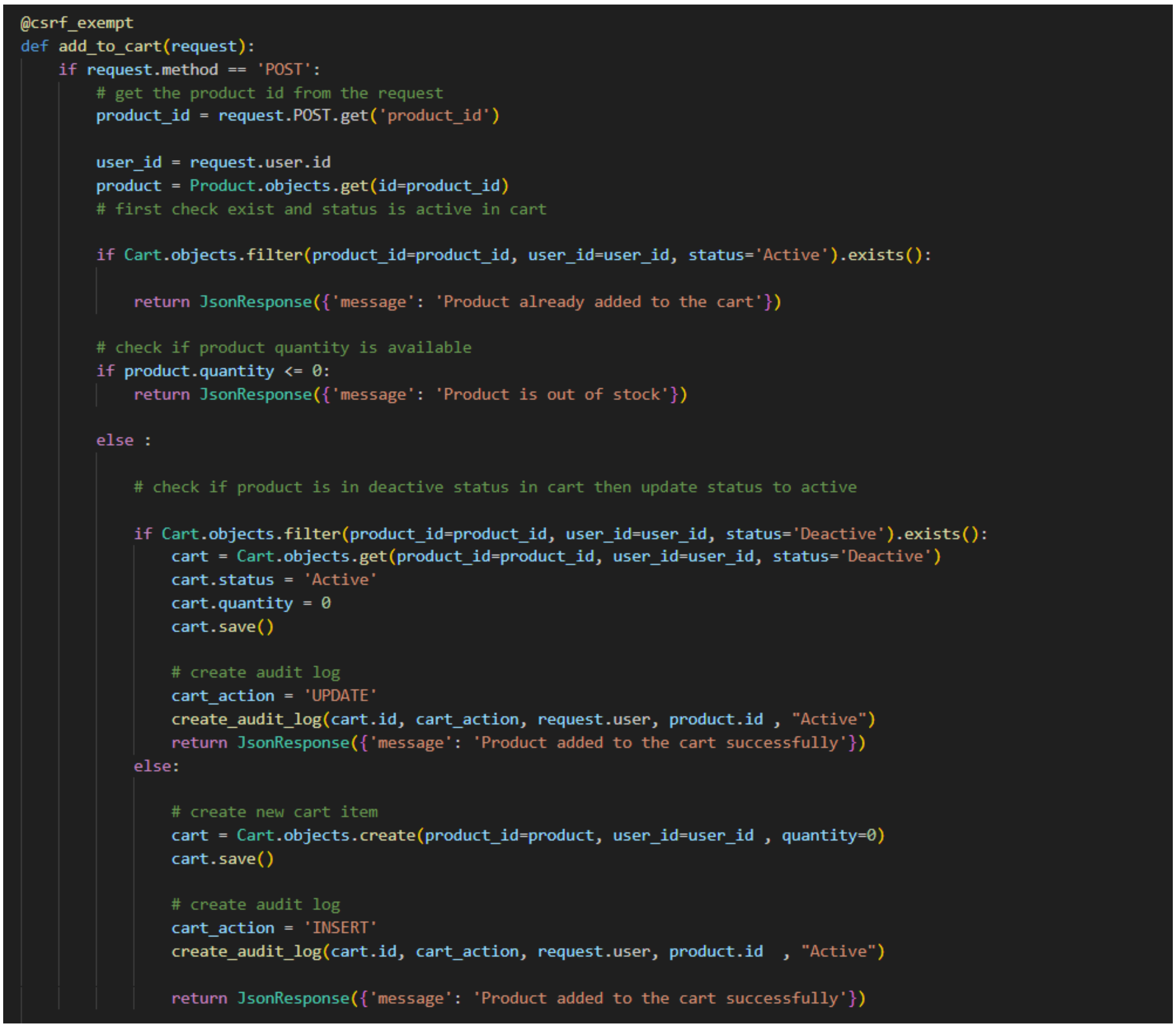
7 Project Management



Name	Last commit message	Last commit date
..		
Ecommerce	templates update	3 weeks ago
assets	product view with backend	2 months ago
media/shop/images	templates update	3 weeks ago
oldmigrations	templates update	3 weeks ago
shop	database schema	3 weeks ago
db.sqlite3	product view with backend	2 months ago
manage.py	product view with backend	2 months ago
response.json()	Enhancement: Finalize core functionalities	last month

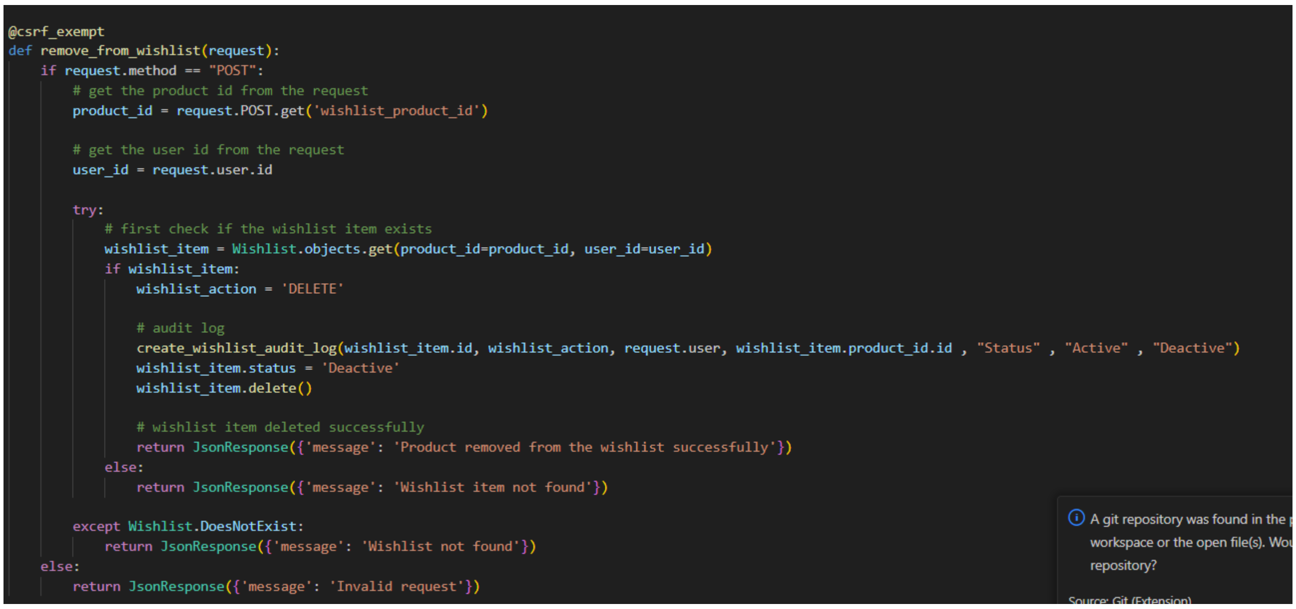
FIGURE 9: Leveraging GitHub as an Integrated Project Management

8 Code Snippets



```
@csrf_exempt
def add_to_cart(request):
    if request.method == 'POST':
        # get the product id from the request
        product_id = request.POST.get('product_id')

        user_id = request.user.id
        product = Product.objects.get(id=product_id)
        # first check exist and status is active in cart

        if Cart.objects.filter(product_id=product_id, user_id=user_id, status='Active').exists():

            return JsonResponse({'message': 'Product already added to the cart'})

        # check if product quantity is available
        if product.quantity <= 0:
            return JsonResponse({'message': 'Product is out of stock'})

        else :

            # check if product is in deactive status in cart then update status to active

            if Cart.objects.filter(product_id=product_id, user_id=user_id, status='Deactive').exists():
                cart = Cart.objects.get(product_id=product_id, user_id=user_id, status='Deactive')
                cart.status = 'Active'
                cart.quantity = 0
                cart.save()

                # create audit log
                cart_action = 'UPDATE'
                create_audit_log(cart.id, cart_action, request.user, product.id , "Active")
                return JsonResponse({'message': 'Product added to the cart successfully'})
            else:

                # create new cart item
                cart = Cart.objects.create(product_id=product, user_id=user_id , quantity=0)
                cart.save()

                # create audit log
                cart_action = 'INSERT'
                create_audit_log(cart.id, cart_action, request.user, product.id  , "Active")

                return JsonResponse({'message': 'Product added to the cart successfully'})
```

FIGURE 11: Exemplary Code Snippet

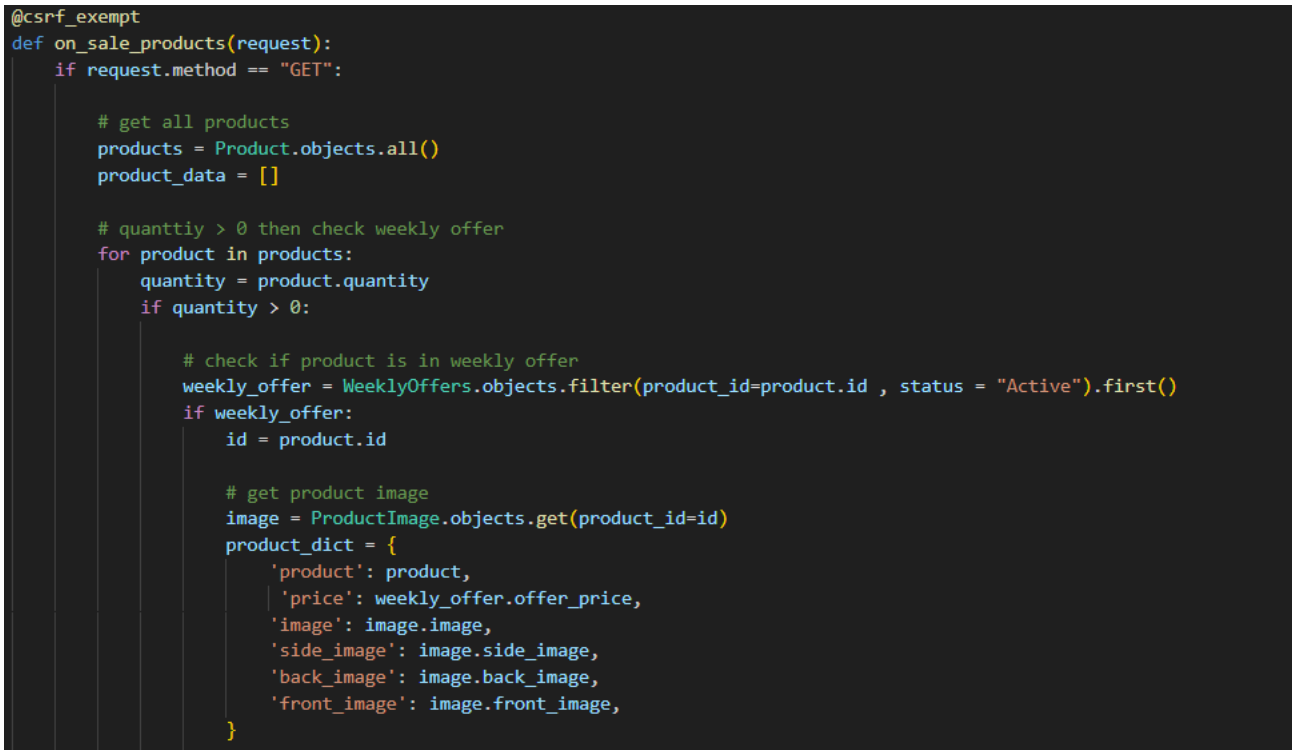


```
@csrf_exempt
def remove_from_wishlist(request):
    if request.method == "POST":
        # get the product id from the request
        product_id = request.POST.get('wishlist_product_id')

        # get the user id from the request
        user_id = request.user.id

        try:
            # first check if the wishlist item exists
            wishlist_item = Wishlist.objects.get(product_id=product_id, user_id=user_id)
            if wishlist_item:
                wishlist_action = 'DELETE'

                # audit log
                create_wishlist_audit_log(wishlist_item.id, wishlist_action, request.user, wishlist_item.product_id.id , "Status" , "Active" , "Deactive")
                wishlist_item.status = 'Deactive'
                wishlist_item.delete()

                # wishlist item deleted successfully
                return JsonResponse({'message': 'Product removed from the wishlist successfully'})
            else:
                return JsonResponse({'message': 'Wishlist item not found'})

        except Wishlist.DoesNotExist:
            return JsonResponse({'message': 'Wishlist not found'})
    else:
        return JsonResponse({'message': 'Invalid request'})
```

A git repository was found in the p
workspace or the open file(s). Wou
repository?

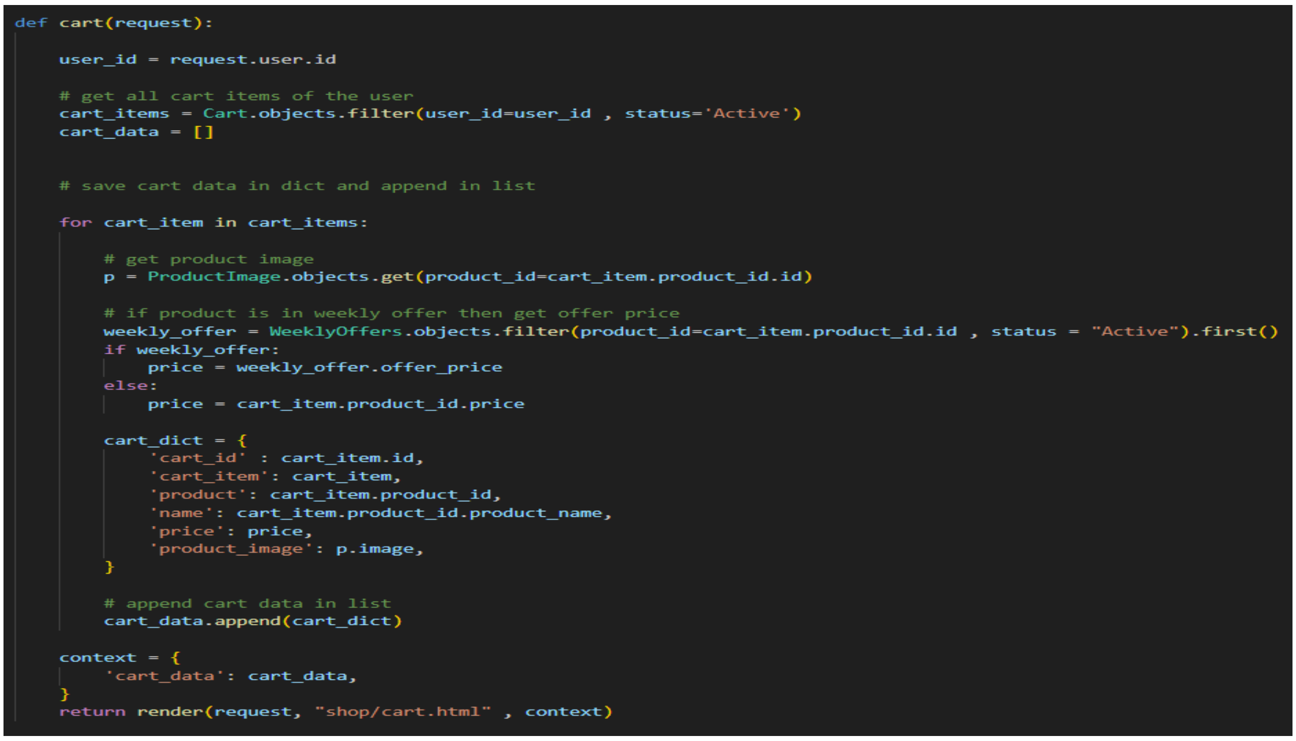
Source: Git (Extension)

FIGURE 12: Exemplary Code Snippet



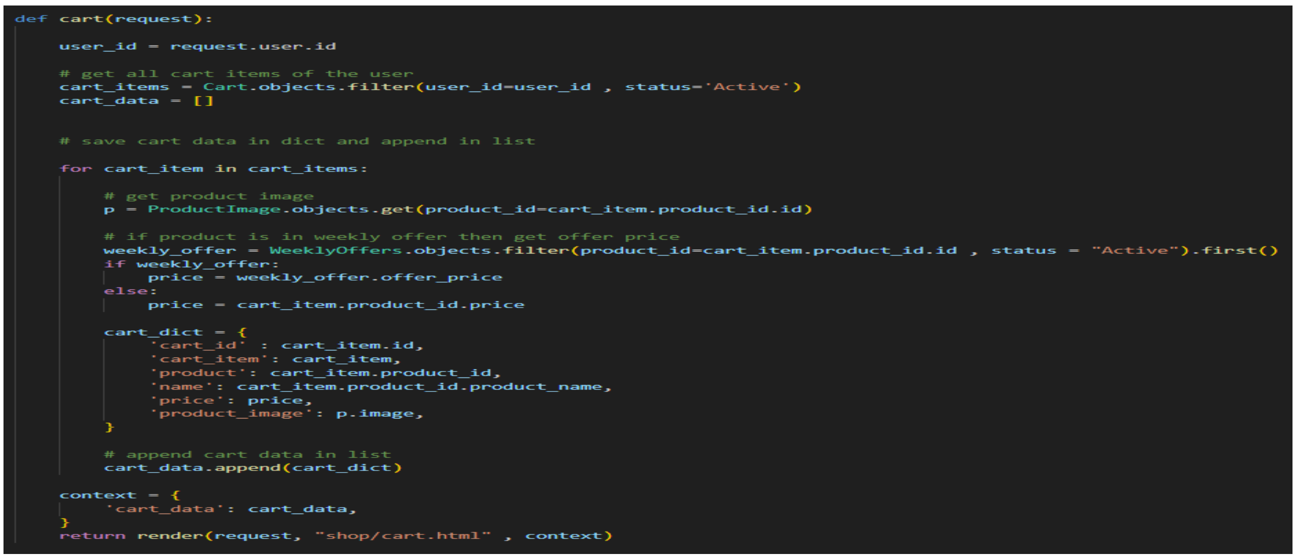
```
@csrf_exempt
def on_sale_products(request):
    if request.method == "GET":

        # get all products
        products = Product.objects.all()
        product_data = []

        # quanttiy > 0 then check weekly offer
        for product in products:
            quantity = product.quantity
            if quantity > 0:

                # check if product is in weekly offer
                weekly_offer = WeeklyOffers.objects.filter(product_id=product.id , status = "Active").first()
                if weekly_offer:
                    id = product.id

                    # get product image
                    image = ProductImage.objects.get(product_id=id)
                    product_dict = {
                        'product': product,
                         'price': weekly_offer.offer_price,
                        'image': image.image,
                        'side_image': image.side_image,
                        'back_image': image.back_image,
                        'front_image': image.front_image,
                    }
```

FIGURE 13: Exemplary Code Snippet



```python
def cart(request):

    user_id = request.user.id

    # get all cart items of the user
    cart_items = Cart.objects.filter(user_id=user_id , status='Active')
    cart_data = []


    # save cart data in dict and append in list

    for cart_item in cart_items:

        # get product image
        p = ProductImage.objects.get(product_id=cart_item.product_id.id)

        # if product is in weekly offer then get offer price
        weekly_offer = WeeklyOffers.objects.filter(product_id=cart_item.product_id.id , status = "Active").first()
        if weekly_offer:
            price = weekly_offer.offer_price
        else:
            price = cart_item.product_id.price

        cart_dict = {
            'cart_id' : cart_item.id,
            'cart_item': cart_item,
            'product': cart_item.product_id,
            'name': cart_item.product_id.product_name,
            'price': price,
            'product_image': p.image,
        }

        # append cart data in list
        cart_data.append(cart_dict)

    context = {
        'cart_data': cart_data,
    }
    return render(request, "shop/cart.html" , context)
```

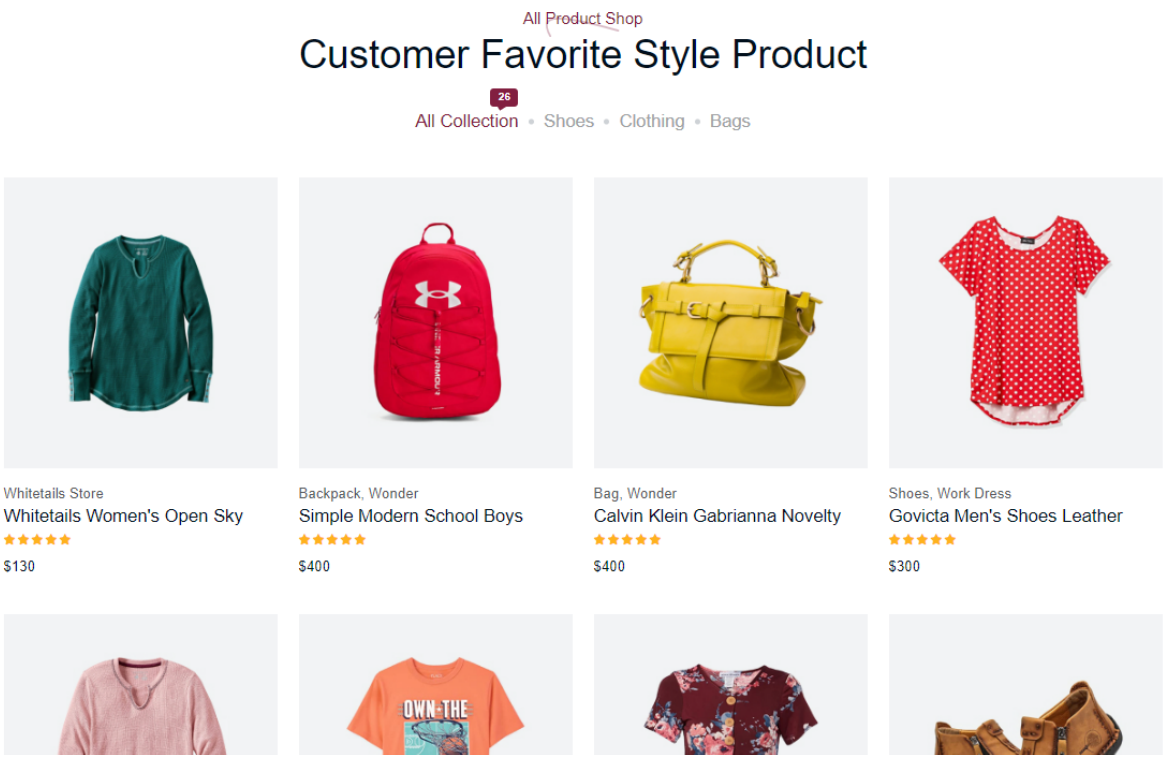
FIGURE 14: Exemplary Code Snippet



```python
def cart(request):

    user_id = request.user.id

    # get all cart items of the user
    cart_items = Cart.objects.filter(user_id=user_id , status='Active')
    cart_data = []


    # save cart data in dict and append in list

    for cart_item in cart_items:

        # get product image
        p = ProductImage.objects.get(product_id=cart_item.product_id.id)

        # if product is in weekly offer then get offer price
        weekly_offer = WeeklyOffers.objects.filter(product_id=cart_item.product_id.id , status = "Active").first()
        if weekly_offer:
            price = weekly_offer.offer_price
        else:
            price = cart_item.product_id.price

        cart_dict = {
            'cart_id' : cart_item.id,
            'cart_item': cart_item,
            'product': cart_item.product_id,
            'name': cart_item.product_id.product_name,
            'price': price,
            'product_image': p.image,
        }

        # append cart data in list
        cart_data.append(cart_dict)

    context = {
        'cart_data': cart_data,
    }
    return render(request, "shop/cart.html" , context)
```

FIGURE 15: Exemplary Code Snippet

9 Screens Layout along with navigation flow



All Product Shop

Customer Favorite Style Product

26

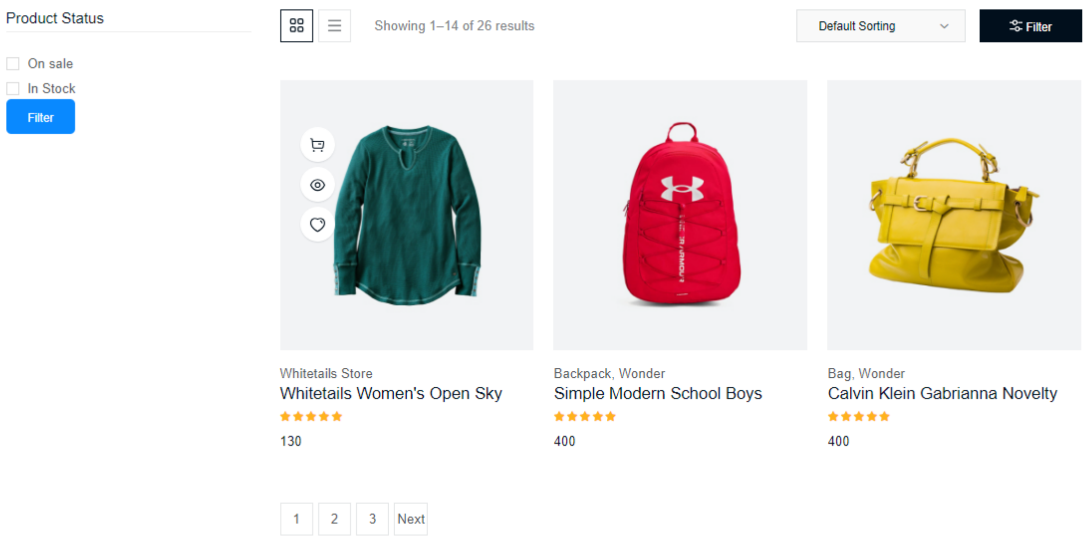
All Collection • Shoes • Clothing • Bags

Whitetails Store
Whitetails Women's Open Sky
★★★★★
$130

Backpack, Wonder
Simple Modern School Boys
★★★★★
$400

Bag, Wonder
Calvin Klein Gabrianna Novelty
★★★★★
$400

Shoes, Work Dress
Govicta Men's Shoes Leather
★★★★★
$300



Product Status

On sale

In Stock

Filter

Showing 1–14 of 26 results

Default Sorting

Filter

Whitetails Store
Whitetails Women's Open Sky
★★★★★
130

Backpack, Wonder
Simple Modern School Boys
★★★★★
400

Bag, Wonder
Calvin Klein Gabrianna Novelty
★★★★★
400

1 2 3 Next

Wishlist

Home • Wishlist

Product		Price	Action	
	Govicta Men's Shoes Leather	$300	Add to Cart	× Remove
	Brown Gown for Women	$350	Add to Cart	× Remove
	Simple Modern School Boys	$400	Add to Cart	× Remove

Go To Cart

Shopping Cart

Home • Shopping Cart

Product		Price	Quantity	
>	Tommy Hilfiger Women's Jaden	5099	- 0 +	× Remove
>	Calvin Klein Gabrianna Novelty	400	- 0 +	× Remove
>	Brown Gown for Women	350	- 0 +	× Remove

Update Cart

Subtotal 0

Shipping

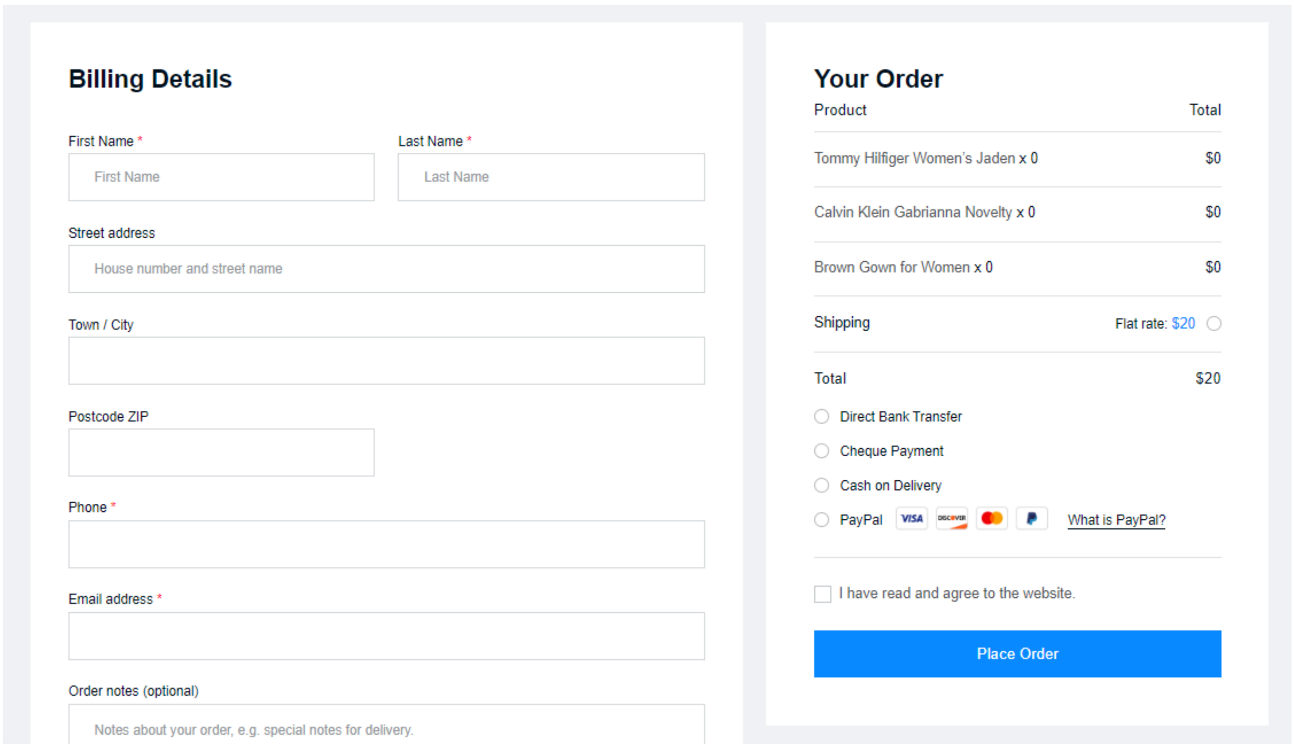
○ Flat rate: $20.00

Total $20

Proceed to Checkout

Shofy

December 2023



Billing Details

First Name *
First Name

Last Name *
Last Name

Street address
House number and street name

Town / City

Postcode ZIP

Phone *

Email address *

Order notes (optional)
Notes about your order, e.g. special notes for delivery.

Your Order

Product	Total
Tommy Hilfiger Women's Jaden x 0	$0
Calvin Klein Gabrianna Novelty x 0	$0
Brown Gown for Women x 0	$0
Shipping	Flat rate: $20
Total	$20

Direct Bank Transfer

Cheque Payment

Cash on Delivery

PayPal — What is PayPal?

I have read and agree to the website.

Place Order

10 Target Audience

Online Shoppers:

Individuals who prefer the convenience of online shopping for a wide range of products.

Tech-Savvy Consumers:

Users comfortable with digital platforms and technology, seeking a seamless and user-friendly online shopping experience.

Fashion Enthusiasts:

Those with an interest in fashion and trends, looking for a platform offering a diverse selection of clothing, accessories, and related products.

Busy Professionals:

Individuals with hectic schedules who value the efficiency of online shopping to meet their needs without spending excess time in physical stores.

Budget-Conscious Consumers:

Shoppers who appreciate the ability to sort products based on price, allowing them to make informed decisions according to their budget.

Gift Shoppers:

Individuals seeking a platform to explore and purchase gifts for various occasions, utilizing wishlist features for thoughtful and planned gifting.

Security-Conscious Users:

Consumers who prioritize secure transactions and personalized experiences, appreciating a robust login and signup process for account management.

E-commerce Enthusiasts:

Individuals who actively engage in online shopping and seek platforms that provide a comprehensive and enjoyable shopping experience