Planning for the E-commerce application

Introduction:

The following is the documentation for the preparation and planning of an e-Commerce platform for vendors and buyers. A two-sided marketplace platform where vendors can sell products and buyers can purchase them. The platform will be built using Django's built-in user model.

Users in the system:

1. **Vendors**
   Store:
   - They can create store
   - View their stores
   - Edit their stores
   - remove their store

   Products:
   - They can add products in their stores
   - They can view products
   - They can edit products
   - They can remove products

2. **Buyers**
   Products:
   - They can view products from different stores
   - They can add them to their carts
   - They can checkout the products

   Reviews:
   - They should be able to leave reviews

3. **Shared abilities**
   - User authentication (login/logout)
   - Password reset functionality
   - Email notifications
   - Session management

User stories

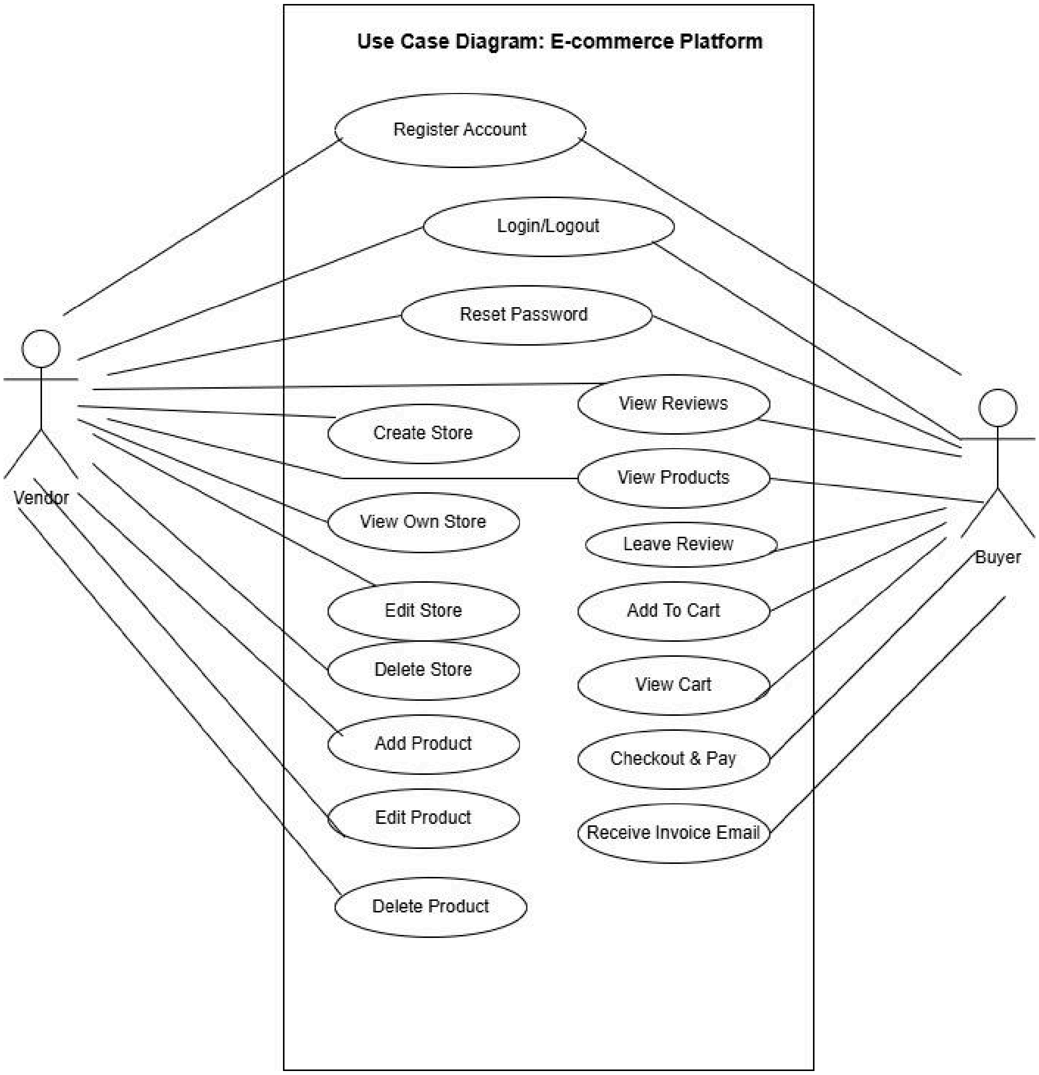
As a Vendor

1. I want to create a store so that I can organise my products.
2. I want to view my inventory so that I can see what I’m selling.
3. I want to delete my store when it is no longer available.
4. I want to add products so that they are available to be bought.
5. I want to edit product details so that I can change the quantity and price.
6. I want to delete products so that I can remove them when they are no longer available.

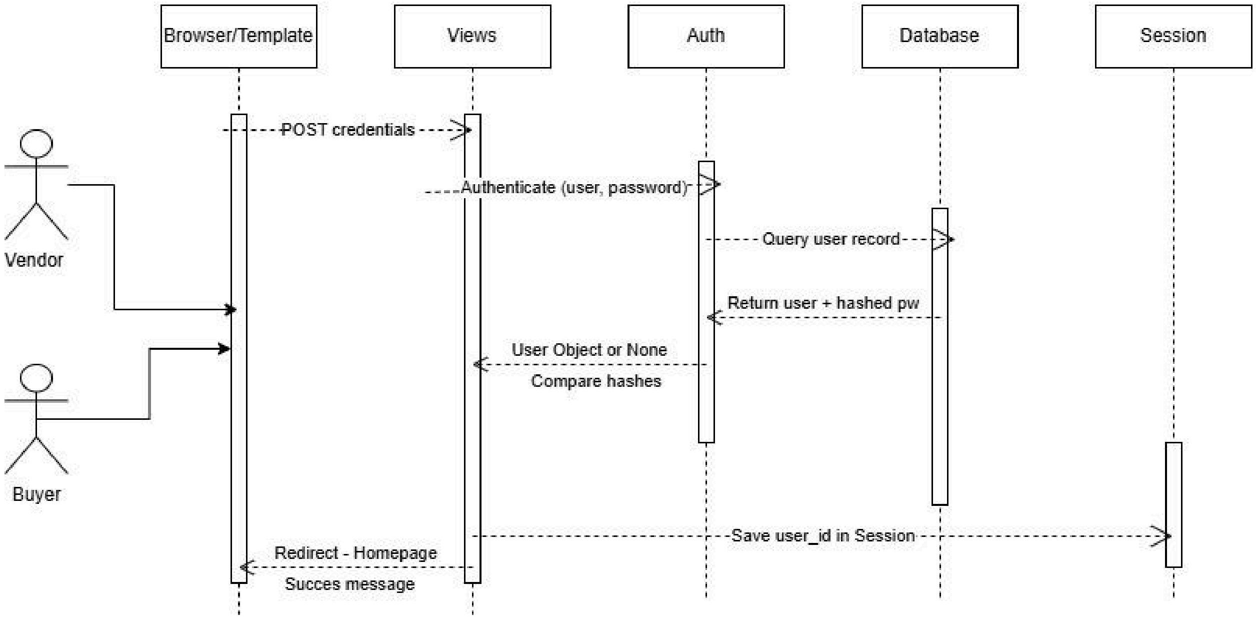
As a Buyer

1. I want to view products so that I can browse what’s available
2. I want to add products to cart so I can checkout later
3. I want to check out so I can purchase items and receive an invoice.
4. I want to leave reviews so that I can share my thoughts and experience with the products
5. I want to reset my password so that I can get access if I forget it.

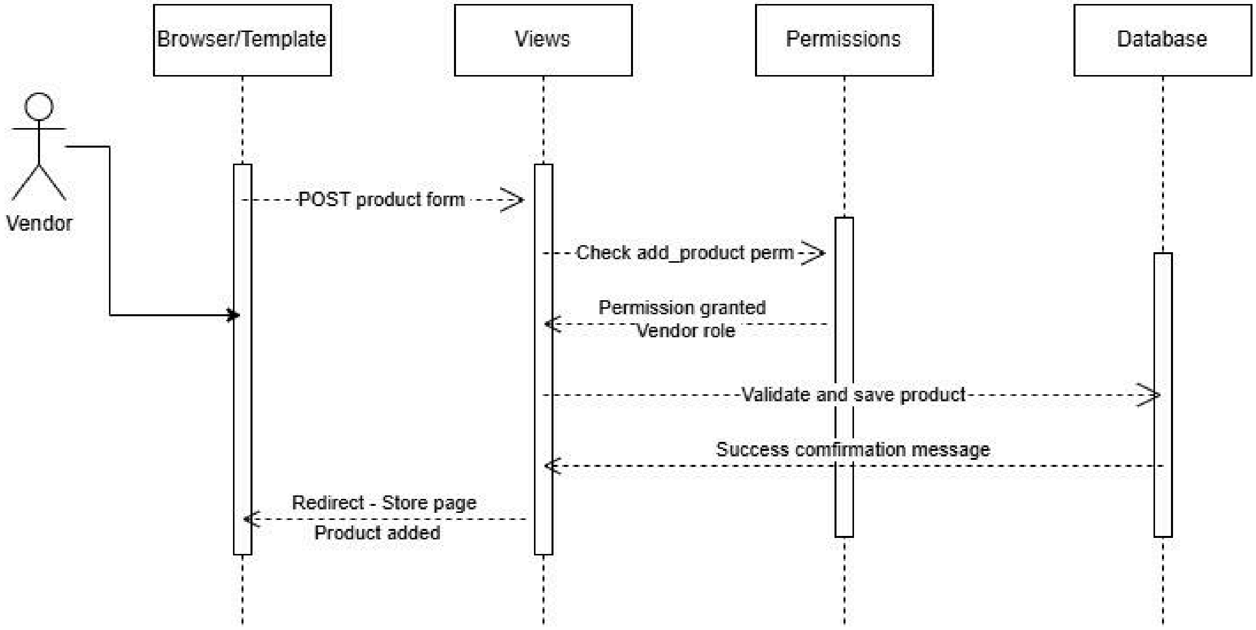
Use Case, Sequence and Class Diagrams



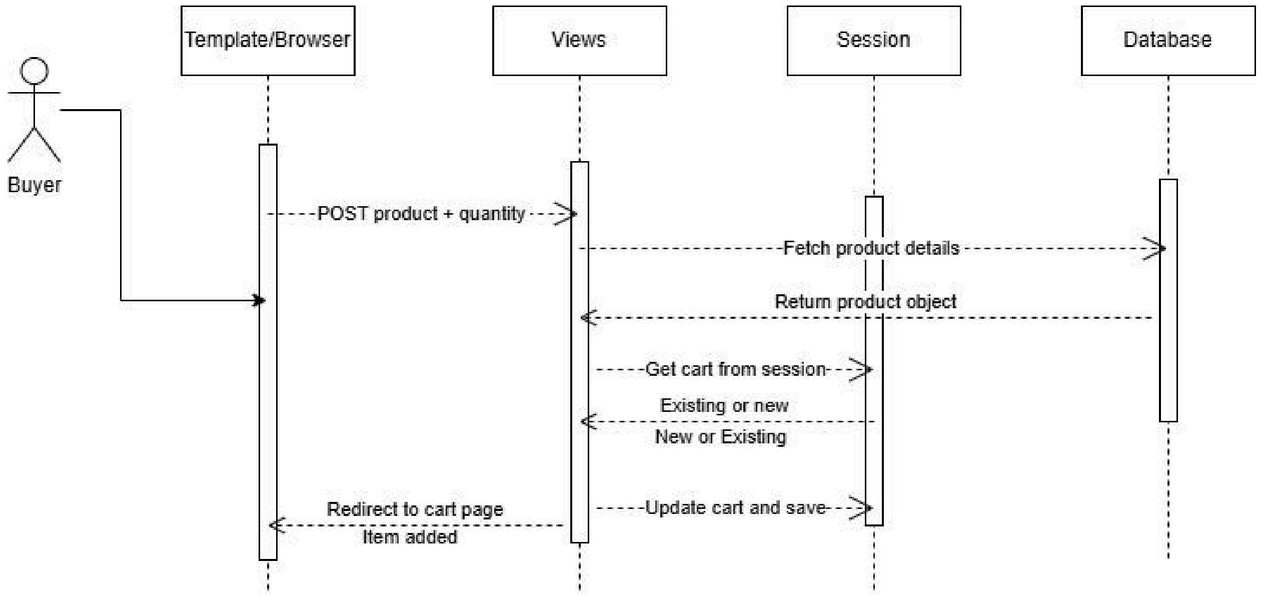
Sequence Diagram: User Login



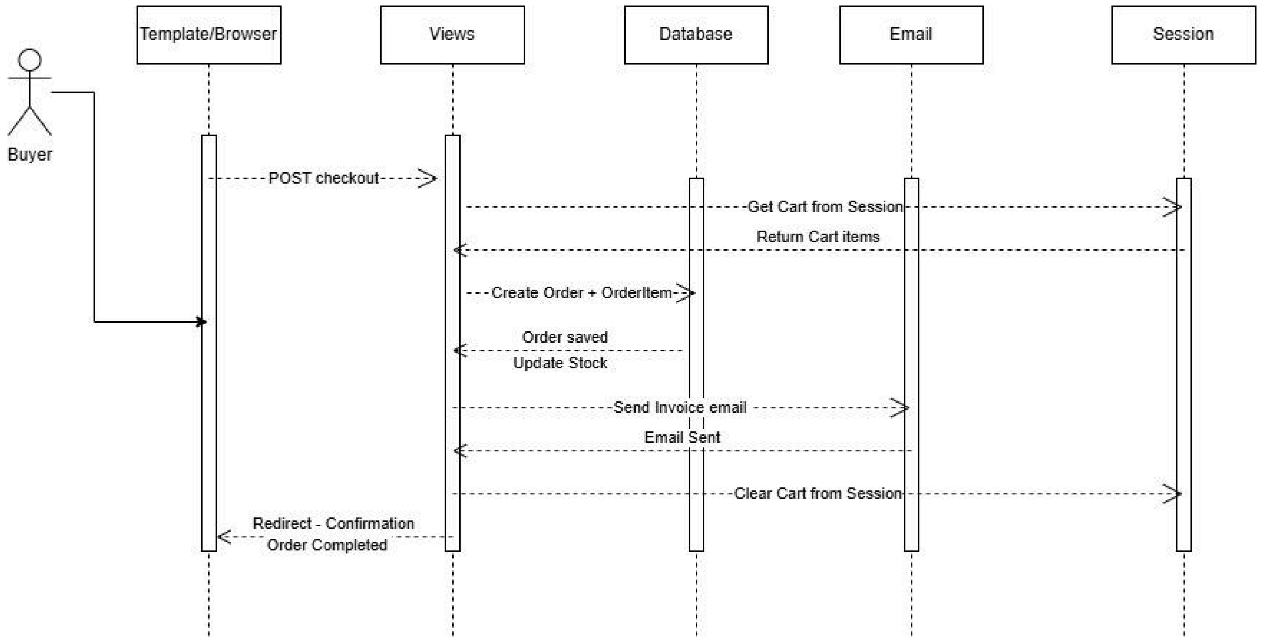
Sequence Diagram: Vendor Adds Product



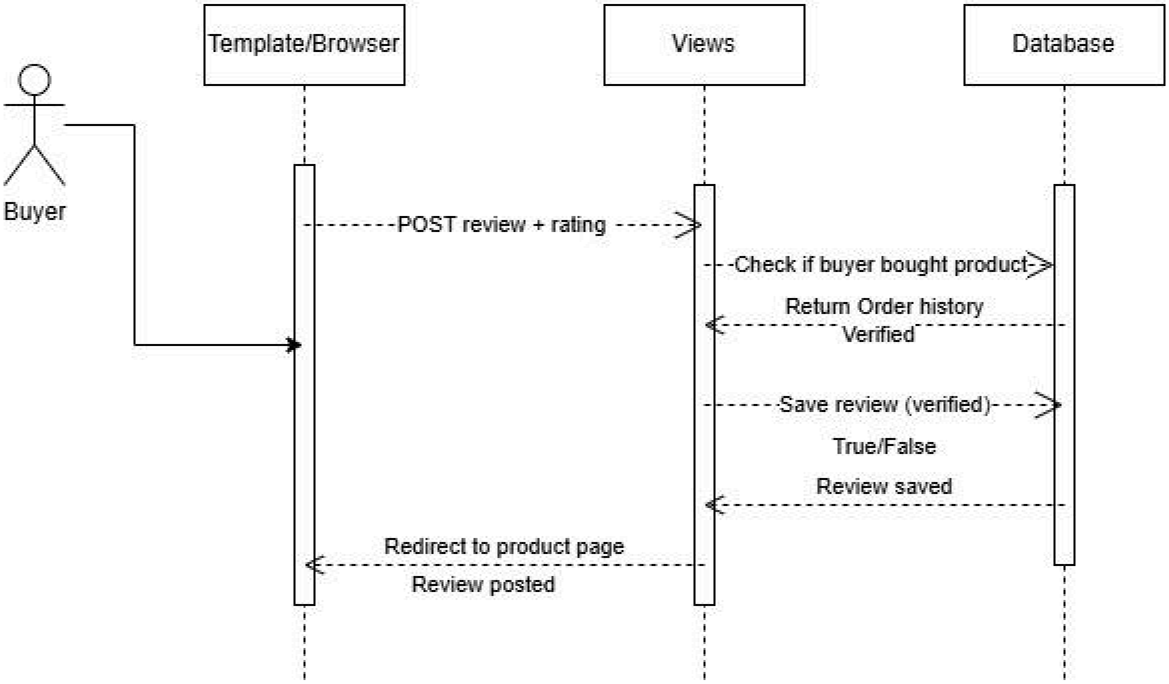
Sequence Diagram: Buyer Adds To Cart



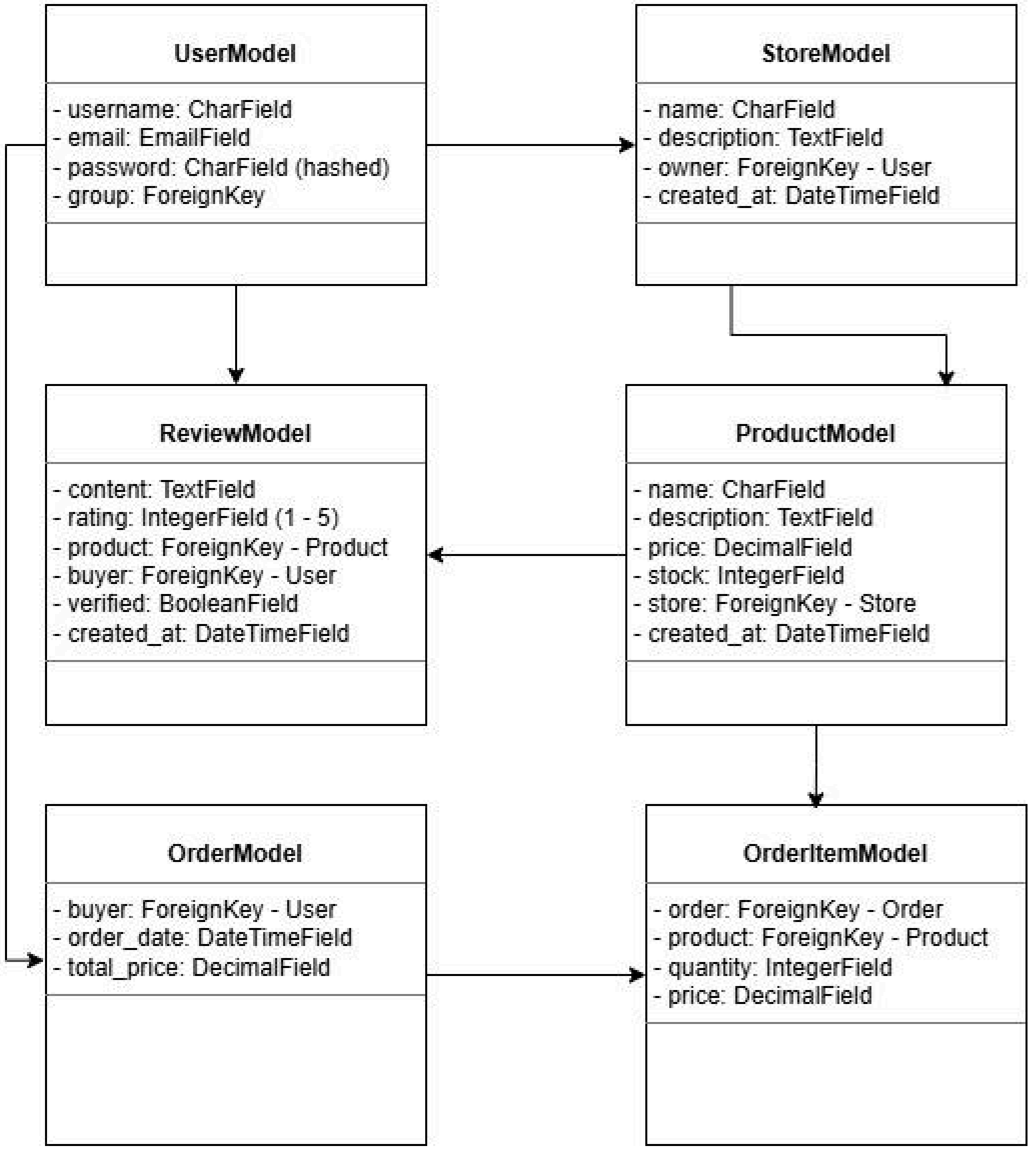
Sequence Diagram: Buyer Checkout



Sequence Diagram: Buyer Leaves Review



Model Diagram: E-Commerce Platform


UserModel
- username: CharField
- email: EmailField
- password: CharField (hashed)
- group: ForeignKey
StoreModel
- name: CharField
- description: TextField
- owner: ForeignKey - User
- created_at: DateTimeField
ReviewModel
- content: TextField
- rating: IntegerField (1 - 5)
- product: ForeignKey - Product
- buyer: ForeignKey - User
- verified: BooleanField
- created_at: DateTimeField
ProductModel
- name: CharField
- description: TextField
- price: DecimalField
- stock: IntegerField
- store: ForeignKey - Store
- created_at: DateTimeField
OrderModel
- buyer: ForeignKey - User
- order_date: DateTimeField
- total_price: DecimalField
OrderItemModel
- order: ForeignKey - Order
- product: ForeignKey - Product
- quantity: IntegerField
- price: DecimalField

Relationships

- One User can own many Stores (One-to-Many)
- One Store can have many Products (One-to-Many)
- One User can place many Orders (One-to-Many)
- One Order can have many OrderItems (One-to-Many)
- One Product can have many Reviews (One-to-Many)
- One Product can appear in many OrderItems (One-to-Many)

CRUD Matrix:

Model	Create	Read	Update	Delete
Store (Vendor)	X	X	X	X
Product (Vendor)	X	X	X	X
Product (Buyer)		X		
Review (Buyer)	X	X	X	X
Order (Buyer)	X	X	X	X
Order Item (Buyer)	X	X	x	x

Routes Table:

Page Name	**URL Pattern**	**Who can access it ?**
Homepage	/ Welcome page, links to register/login	Everyone
Register	/register/ Registration form	Everyone
Login	/login/ Login form	Everyone
Logout	/logout/ Logs user out, redirects to homepage	Everyone
Password Reset Request	/password-reset/ Form to enter email	Everyone

Password Reset Confirm	/password-reset/<token>/ Form to enter new password	Everyone
View All Products	/products/ List of ALL products from all stores	Everyone
View Single Product Detail	/products/<int:pk>/ Detailed page for ONE product	Everyone
View Product Reviews	/products/<int:pk>/reviews/ All reviews for a specific product	Everyone

Page Name	**URL Pattern**	**Who can access it ?**
Vendor Dashboard	/vendor/dashboard/ Overview of vendor's stores and products	Vendor
Create Store	/vendor/stores/create/ Form to create a new store	Vendor
View My Stores (List)	/vendor/stores/ All stores owned by this vendor	Vendor
View Single Store Detail	/vendor/stores/<int:pk>/ Details of ONE specific store	Vendor
Delete Store	/vendor/stores/<int:pk>/delete/ Deletes specific store	Vendor

Add Product (to a specific store)	/vendor/stores/<int:store_pk>/products/add/ Form to add product	Vendor
View My Products (List)	/vendor/products/ All products from all YOUR stores	Vendor
View Single Product	/vendor/products/<int:pk>/ Details of ONE product you own	Vendor
Edit Product	/vendor/products/<int:pk>/edit/ Form to edit a specific product	Vendor
Delete Product	/vendor/products/<int:pk>/delete/ Deletes specific product	Vendor

Page Name	**URL Pattern**	**Who can access it?**
Buyer Dashboard	/buyer/dashboard/ Overview of orders, cart summary	Buyer
View Cart	/buyer/cart/ All items currently in cart	Buyer
Add to Cart	/buyer/cart/add/<int:product_pk>/ Adds product to cart, redirects back	Buyer
Remove from Cart	/buyer/cart/remove/<int:product_pk>/ Removes product from cart	Buyer
Update Cart Quantity	/buyer/cart/update/<int:product_pk>/	Buyer

	Changes quantity of item in cart	
Checkout	/buyer/checkout/ Checkout page with order summary	Buyer
Order Confirmation	/buyer/orders/<int:order_pk>/confirmation/ Success message after checkout	Buyer
View Order History	/buyer/orders/ All past orders	Buyer
View Single Order Detail	/buyer/orders/<int:pk>/ Details of ONE specific order	Buyer
Add Review (for a product)	/products/<int:product_pk>/reviews/add/ Form to write review	Buyer
Edit Review	/reviews/<int:pk>/edit/ Form to edit your review	Buyer
Delete Review	/reviews/<int:pk>/delete/ Deletes your review	Buyer

Pseudocode for Register view

```
FUNCTION register_user(request):
    IF request method is POST:
        GET username, email, password, account_type from form

        VALIDATE:
            IF username is empty OR email is empty OR password is empty:
                SHOW error “All fields required”
                RETURN to register form

            IF username already exists:
                SHOW error “Username already taken”
                RETURN to register form

            IF email already exists:
                SHOW error “Email already registered”
                RETURN to register form

            IF password is too short (less than 8 characters):
                SHOW error “Password too short”
                RETURN to register form

        CREATE user with username, email, password

        IF account_type is “vendor”:
            ADD user to Vendors group
        ELSE IF account_type is “buyer”:
            ADD user to Buyers group

        LOG IN the user automatically

        REDIRECT to homepage with success message

    ELSE:
        SHOW empty registration form
```

Pseudocode for login view

```
FUNCTION login_user(request):
    IF request method is POST:
        GET username and password from form

        IF username is empty OR password is empty:
            SHOW error “All fields required”
            RETURN to login form

        AUTHENTICATE user with username and password

        IF authentication successful:
        LOG IN the user
        CREATE session

        IF user is in the Vendors group:
            REDIRECT to vendor dashboard
        ELSE IF user is in Buyers group:
            REDIRECT to buyer dashboard
        ELSE:
            REDIRECT to homepage

    ELSE:
        SHOW error “Invalid credentials”
        RETURN to login form
ELSE:
    SHOW empty login form
```

Pseudocode for Logout View

```
FUNCTION logout_user(request):
    IF user is logged in:
        LOG OUT the user
        CLEAR session data
        SHOW message “Logged out successfully”

    REDIRECT to homepage
```

Pseudocode for Create Store:

```
FUNCTION vendor_store_create(request):
    CHECK if user is logged in
    CHECK if user is a Vendor (has permission)

    IF request method is POST:
        GET store name and description from form

        VALIDATE:
            IF name is empty OR description is empty:
                SHOW error "All fields required"
                RETURN to form

            IF store name already exists for this vendor:
                SHOW error "You already have a store with this name" RETURN
                to form

            CREATE new Store object:
            - name = form name
            - description = form description
            - owner = current user
            - created_at = auto-filled

            SAVE store to database
            SHOW success message REDIRECT to store detail page

    ELSE:
        SHOW empty create store form
```