Mini Project: Online Product Catalog System with CRUD Operations and User Authentication

Objective

The goal of this project is to create an **Online Product Catalog System** where users can browse and view detailed information about products. Users can also sign up, log in, and log out of the system. Additionally, users will be able to perform **CRUD (Create, Read, Update, Delete)** operations on product information. This project will help students understand how to create models, implement relationships between tables, and integrate user authentication and basic CRUD functionality in Django.

Entities and Relationships

In this system, we will work with two main entities: **Product** and **Category**. We will also integrate user authentication for managing access to the system.

1. Product Entity

The **Product** entity represents the products in the catalog.

Fields in the Product Entity:

- **name** (CharField):
  - **Type**: CharField
  - **Description**: This field will store the name of the product. It is used to uniquely identify the product.
  - **Argument**: max_length=255 — Defines the maximum length of the product name.
  - **Example**: "Laptop", "Smartphone".
- **description** (TextField):
  - **Type**: TextField
  - **Description**: This field stores a detailed description of the product. It can hold larger text content, such as features and specifications.
  - **Argument**: No max_length needed.
  - **Example**: "This laptop is equipped with a 16GB RAM, 512GB SSD, and a 15.6-inch display."
- **price** (DecimalField):
  - **Type**: DecimalField
  - **Description**: The price of the product will be stored here. It is essential to store the price as a decimal value to represent monetary amounts accurately.
  - **Arguments**: max_digits=10, decimal_places=2 — max_digits ensures the total number of digits, and decimal_places allows for two decimal places (cents).
  - **Example**: 499.99 (representing $499.99).

- **stock** (IntegerField):
  - **Type**: IntegerField
  - **Description**: This field stores the number of units available for purchase.
  - **Argument**: default=0 — Optionally, set a default stock value to 0 to indicate no units are available initially.
  - **Example**: 100 (indicating 100 units available).
- **category** (ForeignKey):
  - **Type**: ForeignKey
  - **Description**: This field links a product to a specific category.
  - **Arguments**: on_delete=models.CASCADE — Ensures that if a category is deleted, all related products will also be deleted automatically.
  - **Relationship**: Many-to-One with **Category** (One category can have multiple products).
  - **Example**: "Computers", "Mobile Phones".
- **image** (ImageField):
  - **Type**: ImageField
  - **Description**: This field stores an image of the product.
  - **Arguments**: upload_to='product_images/' — Specifies the folder where uploaded product images will be stored.
  - **Example**: Image file like "laptop.jpg".

2. Category Entity

The **Category** entity groups products into categories.

Fields in the Category Entity:

- **name** (CharField):
  - **Type**: CharField
  - **Description**: This field stores the name of the category. Categories help organize products into logical groups.
  - **Argument**: max_length=100 — Short text for category names.
  - **Example**: "Computers", "Mobile Phones".
- **description** (TextField):
  - **Type**: TextField
  - **Description**: This field provides a description of what products belong to this category.

- **Argument**: No max_length required.
- **Example**: "This category includes laptops, desktops, and accessories."

User Authentication (User Entity)

Django's built-in User model will be used for user authentication (sign up, login, logout).

Fields in the User Entity:

- **username** (CharField):
  - **Type**: CharField
  - **Description**: This field stores the username of the user, which will be used for login.
  - **Argument**: max_length=150 — Maximum length for the username.
  - **Example**: "john_doe".
- **password** (CharField):
  - **Type**: CharField
  - **Description**: Stores the hashed password of the user.
  - **Argument**: max_length=128 — Typically used for password storage in Django.
- **email** (EmailField):
  - **Type**: EmailField
  - **Description**: Stores the user's email address, used for notifications and recovery.
  - **Argument**: max_length=254 — Standard length for email addresses.

CRUD Features

This project will involve **Create, Read, Update, Delete (CRUD)** functionality for products and user authentication.

For the Product Entity:

1. **Create**:
   - Users can add new products to the catalog by filling in the product details such as name, description, price, stock, category, and image.
   - **URL**: /products/create/
   - **View Function**: create_product()
   - **Template**: create_product.html
2. **Read**:

   - All users can browse and view the product catalog, and see the product details such as name, description, price, and image.
   - **URL**: /products/ (for listing products) or /products/<product_id>/ (for detailed view)
   - **View Function**: product_list(), product_detail()
   - **Template**: product_list.html, product_detail.html

3. **Update**:
   - Users can edit the details of products they have created (only allowed if authenticated as the product creator).
   - **URL**: /products/update/<product_id>/
   - **View Function**: update_product()
   - **Template**: update_product.html

4. **Delete**:
   - Users can delete products they have created (only allowed if authenticated as the product creator).
   - **URL**: /products/delete/<product_id>/
   - **View Function**: delete_product()
   - **Template**: delete_product.html

For User Authentication:

1. **Sign Up (Create)**:
   - New users can sign up by providing a username, email, and password.
   - **URL**: /accounts/signup/
   - **View Function**: signup()
   - **Template**: signup.html

2. **Log In (Read)**:
   - Registered users can log in with their username and password.
   - **URL**: /accounts/login/
   - **View Function**: login_view()
   - **Template**: login.html

3. **Log Out (Delete)**:
   - Users can log out of the system to end their session.
   - **URL**: /accounts/logout/
   - **View Function**: logout_view()

  - **Template**: N/A (Handled by Django's built-in logout)

4. **Password Reset (Update)**:
    - Users should be able to reset their password if they forget it.
    - **URL**: /accounts/password_reset/
    - **View Function**: password_reset_view()
    - **Template**: password_reset.html

Field Descriptions and Suggestions

Product Fields:

- **name** (CharField):
    - Used to store product names (e.g., "Laptop", "Smartphone").
    - **Tip**: Ensure that product names are unique and descriptive for better user experience.
- **description** (TextField):
    - Store detailed descriptions of the product's features, specifications, and usage.
    - **Tip**: Focus on essential features and benefits to attract potential buyers.
- **price** (DecimalField):
    - Store the product price in a decimal format to handle values like $499.99.
    - **Tip**: Ensure that the price is displayed clearly on the product page.
- **stock** (IntegerField):
    - Used to track the number of units available for purchase.
    - **Tip**: Regularly update stock levels to reflect real-time availability.
- **category** (ForeignKey):
    - Links products to categories, helping organize them logically.
    - **Tip**: Make categories specific enough to avoid having too many products in one category.
- **image** (ImageField):
    - Stores an image for the product.
    - **Tip**: Use high-quality images that accurately represent the product.

Category Fields:

- **name** (CharField):
    - Represents the category name (e.g., "Electronics", "Furniture").

    - **Tip**: Categories should be simple and broad enough to contain a variety of products.

- **description** (TextField):
    - Provides more context about what the category entails.
    - **Tip**: Keep the description concise but informative for users.