Django Models Explanation

1 Introduction

This document provides a detailed explanation of the Django models used in the project. Each model is described with its purpose, fields, relationships, and key methods. The models are part of an e-commerce-like application with user management, product catalog, cart, wishlist, order processing, and feedback functionalities.

2 CustomUser Model

The `CustomUser` model extends Django's `AbstractUser` to provide a customized user model with additional fields and relationships.

- **Purpose**: Represents a user in the system with extended attributes like email, phone number, and profile photo.
- **Fields**:
  - `email`: An email field that must be unique.
  - `phone_number`: A character field for storing phone numbers (up to 15 characters), optional.
  - `photo`: An image field for user profile photos, uploaded to `user_photos/`, optional.
  - Inherits fields from `AbstractUser` (e.g., `username`, `first_name`, `last_name`, `password`).
- **Relationships**:
  - `groups`: Many-to-many relationship with Django's `auth.Group`, allowing users to belong to multiple groups.
  - `user_permissions`: Many-to-many relationship with Django's `auth.Permission`, for specific user permissions.
- **Meta Options**:
  - `verbose_name`: Set to "Custom User" for singular display.
  - `verbose_name_plural`: Set to "Custom Users" for plural display.

3 Vendor Model

The `Vendor` model represents a vendor profile linked to a `CustomUser`, used for managing vendor-specific information.

- **Purpose**: Stores vendor details, such as company name and GST number, and tracks approval status.
- **Fields**:
  - `user`: One-to-one relationship with `CustomUser`, deleted if the user is deleted.
  - `company_name`: A character field for the vendor's company name (up to 100 characters).
  - `gst_number`: A unique character field for the GST number (15 characters).
  - `is_approved`: A boolean field indicating whether the vendor is approved (default: `False`).
  - `applied_at`: A datetime field recording when the vendor applied, automatically set on creation.
- **Methods**:
  - `clean()`: Validates the GST number using a regular expression to ensure it is 15 alphanumeric characters.
  - `__str__()`: Returns a string representation of the vendor, including username, company name, and approval status.

4 ProductCategory Model

The `ProductCategory` model defines categories for organizing products.

- **Purpose**: Groups products into categories (e.g., Electronics, Clothing).
- **Fields**:
  - `name`: A character field for the category name (up to 100 characters).
  - `image`: An image field for category images, uploaded to `category_images/`.
- **Methods**:
  - `__str__()`: Returns the category name as a string.

5 ProductType Model

The `ProductType` model represents specific types within a product category.

- **Purpose**: Organizes products into subtypes under a category (e.g., Smartphones under Electronics).
- **Fields**:

  - `category`: A foreign key to `ProductCategory`, deleted if the category is deleted.
  - `name`: A character field for the type name (up to 100 characters).
  - `image`: An image field for type images, uploaded to `type_images/`.
- **Methods**:
  - `__str__()`: Returns a string combining the category and type names.

6 Product Model

The `Product` model represents individual products available for sale.

- **Purpose**: Stores product details, including pricing, stock, and media, linked to a vendor and category/type.
- **Fields**:
  - `type`: A foreign key to `ProductType`, optional, deleted if the type is deleted.
  - `category`: A foreign key to `ProductCategory`, optional, deleted if the category is deleted.
  - `vendor`: A foreign key to `Vendor`, deleted if the vendor is deleted.
  - `name`: A character field for the product name (up to 100 characters).
  - `basic_description`: An HTML field for a short product description (using TinyMCE).
  - `description`: An HTML field for a detailed product description (using TinyMCE).
  - `actual_price`: A float field for the product's original price.
  - `discounted_price`: A float field for the discounted price.
  - `buying_price`: A float field for the vendor's buying price (default: 0.0).
  - `image`: An image field for the product image, uploaded to `product_images/`.
  - `created_at`: A datetime field recording creation time, automatically set.
  - `stock_count`: A positive integer field for the available stock (default: 0).
- **Methods**:
  - `__str__()`: Returns the product name as a string.
  - `stock_status()`: Returns a string indicating stock availability ("Product not available", "Only a few left", or empty string).
  - `profit()`: Calculates total profit as `(actual_price * stock_count) - (buying_price * stock_count)`.

7 ProductMedia Model

The `ProductMedia` model stores additional media files (images or videos) for products.

- **Purpose**: Allows multiple media files to be associated with a product.
- **Fields**:
  - `product`: A foreign key to `Product`, deleted if the product is deleted.
  - `file`: A file field for media, uploaded to `product_media/`.
- **Methods**:
  - `is_image()`: Checks if the file is an image (based on extensions like .png, .jpg).
  - `is_video()`: Checks if the file is a video (based on extensions like .mp4, .mov).
  - `__str__()`: Returns a string indicating the media type and product name.

8 Cart Model

The `Cart` model represents items in a user's shopping cart.

- **Purpose**: Tracks products and quantities a user intends to purchase.
- **Fields**:
  - `user`: A foreign key to `CustomUser`, deleted if the user is deleted.
  - `product`: A foreign key to `Product`, deleted if the product is deleted.
  - `quantity`: A positive integer field for the number of items (default: 1).
- **Methods**:
  - `total_price()`: Calculates the total price as `quantity * product.discounted_price`.

9 Wishlist Model

The `Wishlist` model stores products a user has marked as favorites.

- **Purpose**: Allows users to save products for future reference.
- **Fields**:
  - `user`: A foreign key to `CustomUser`, deleted if the user is deleted.
  - `product`: A foreign key to `Product`, deleted if the product is deleted.
- **Meta Options**:
  - `unique_together`: Ensures a user cannot add the same product to their wishlist multiple times.

- **Methods**:
  - `__str__()`: Returns a string with the username and product name.

10 Address Model

The `Address` model stores user shipping addresses.

- **Purpose**: Saves address details for order delivery.
- **Fields**:
  - `user`: A foreign key to `CustomUser`, deleted if the user is deleted.
  - `name`: A character field for the recipient's name (up to 100 characters).
  - `locality`: A character field for the locality (up to 200 characters).
  - `city`: A character field for the city (up to 100 characters).
  - `state`: A character field for the state (up to 100 characters).
  - `zipcode`: A character field for the zip code (up to 10 characters).
  - `mobile`: A character field for the mobile number (up to 15 characters).
- **Methods**:
  - `__str__()`: Returns a string with the name, locality, and city.

11 Order Model

The `Order` model represents a user's purchase order.

- **Purpose**: Tracks orders, including product, address, status, and payment method.
- **Fields**:
  - `user`: A foreign key to `CustomUser`, deleted if the user is deleted.
  - `product`: A foreign key to `Product`, deleted if the product is deleted.
  - `address`: A foreign key to `Address`, deleted if the address is deleted.
  - `order_date`: A datetime field recording when the order was placed, automatically set.
  - `status`: A character field with choices (e.g., Ordered, Shipped, Delivered), default: "Ordered".
  - `payment_method`: A character field with choices ("COD", "Online Payment"), default: "COD".
  - `quantity`: A positive integer field for the number of items (default: 1).
  - `purchase_price`: A float field for the price at the time of purchase, optional.
- **Methods**:

  - `__str__()`: Returns a string with the order ID and username.

12 Feedback Model

The `Feedback` model stores user reviews for products.

- **Purpose**: Allows users to rate and comment on products.
- **Fields**:
  - `user`: A foreign key to `CustomUser`, deleted if the user is deleted.
  - `product`: A foreign key to `Product`, deleted if the product is deleted.
  - `rating`: An integer field for ratings (1 to 5).
  - `comment`: A text field for the review comment.
  - `created_at`: A datetime field recording when the feedback was created, automatically set.
- **Methods**:
  - `__str__()`: Returns a string with the username, product name, and rating.

13 OrderCancel Model

The `OrderCancel` model tracks order cancellations.

- **Purpose**: Records details of canceled orders, including the reason.
- **Fields**:
  - `order`: A one-to-one field to `Order`, deleted if the order is deleted.
  - `user`: A foreign key to `CustomUser`, deleted if the user is deleted.
  - `reason`: A text field for the cancellation reason.
  - `canceled_at`: A datetime field recording when the cancellation occurred, automatically set.
- **Methods**:
  - `__str__()`: Returns a string with the order ID and username.

14 Conclusion

These models collectively form the backbone of an e-commerce platform, handling user management, vendor profiles, product organization, shopping cart, wishlist, order processing, feedback, and cancellations. Each model is designed with appropriate fields, relationships, and methods to ensure functionality and data integrity.