# C:\Users\Surecode\Documents\GitHub\django\coreEat\auths\models.py

from django.db import models

from django.contrib.auth.models import AbstractUser

# ✅ Category Model

class Category(models.Model):

name = models.CharField(max\_length=100, unique=True)

description = models.TextField(blank=True, null=True)

class Meta:

ordering = ['name']

verbose\_name\_plural = 'Categories'

def \_\_str\_\_(self):

return self.name

# ✅ Abstract Product Model

class Product(models.Model):

product\_id = models.CharField(max\_length=100, unique=True, primary\_key=True)

name = models.CharField(max\_length=200)

price = models.DecimalField(max\_digits=10, decimal\_places=2)

description = models.TextField(blank=True, null=True)

image\_url = models.URLField(max\_length=300, blank=True, null=True)

category = models.ForeignKey(Category, on\_delete=models.CASCADE, related\_name='%(class)s\_products') # Dynamic related\_name

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

class Meta:

abstract = True

ordering = ['-created\_at']

def \_\_str\_\_(self):

return f"{self.name} (${self.price})"

# ✅ FastFood Model (inherits from Product)

class FastFood(Product):

class Meta:

verbose\_name = "Fast Food"

verbose\_name\_plural = "Fast Foods"

# ✅ Food Model (inherits from Product)

class Food(Product):

class Meta:

verbose\_name = "Food"

verbose\_name\_plural = "Foods"

# ✅ Drink Model (inherits from Product)

class Drink(Product):

class Meta:

verbose\_name = "Drink"

verbose\_name\_plural = "Drinks"

# ✅ Custom User Model

class User(AbstractUser):

USER\_TYPE\_CHOICES = [

('admin', 'Admin'),

('customer', 'Customer'),

('staff', 'Staff'),

]

GENDER\_TYPE\_CHOICES = [

('mr', 'Mr.'),

('mrs', 'Mrs.'),

('miss', 'Miss'),

('ms', 'Ms.'),

]

def user\_directory\_path(instance, filename):

# File will be uploaded to MEDIA\_ROOT/profile\_images/<username>/<filename>

return f'profile\_images/{instance.username}/{filename}'

first\_name = models.CharField(max\_length=100)

last\_name = models.CharField(max\_length=100)

user\_type = models.CharField(max\_length=10, choices=USER\_TYPE\_CHOICES, default='customer')

gender = models.CharField(max\_length=10, choices=GENDER\_TYPE\_CHOICES, blank=True, null=True)

phone\_number = models.CharField(max\_length=20, blank=True, null=True)

email = models.EmailField(max\_length=100, blank=False, null=True)

image = models.ImageField(

upload\_to=user\_directory\_path,

default='auths/images/empty.png',

blank=True,

null=True,

verbose\_name="Profile Image"

)

def \_\_str\_\_(self):

gender\_prefix = dict(self.GENDER\_TYPE\_CHOICES).get(self.gender, "")

return f"{gender\_prefix} {self.first\_name} {self.last\_name} ({self.get\_user\_type\_display()})"

from django.db import models

from auths.models import User, FastFood, Food, Drink

class Cart(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

def total(self):

return sum(item.subtotal() for item in self.cartitem\_set.all())

def \_\_str\_\_(self):

return f"Cart for {self.user.username}"

class CartItem(models.Model):

cart = models.ForeignKey(Cart, on\_delete=models.CASCADE)

fast\_food = models.ForeignKey(FastFood, null=True, blank=True, on\_delete=models.CASCADE)

food = models.ForeignKey(Food, null=True, blank=True, on\_delete=models.CASCADE)

drink = models.ForeignKey(Drink, null=True, blank=True, on\_delete=models.CASCADE)

quantity = models.PositiveIntegerField(default=1)

quality = models.CharField(max\_length=50)

def get\_product(self):

return self.fast\_food or self.food or self.drink

def subtotal(self):

product = self.get\_product()

if not product:

return 0

return product.price \* self.quantity

def \_\_str\_\_(self):

product = self.get\_product()

return f"{product.name} x{self.quantity}"

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'sim',

'auths',

'cart',

'payments.apps.PaymentsConfig',

'paypal.standard.ipn',

'pwa',

'staffs.apps.StaffsConfig',

]

# payments/models.py

from django.db import models

from auths.models import User

from cart.models import Cart

import logging

logger = logging.getLogger(\_\_name\_\_)

class DeliveryInfo(models.Model):

id = models.BigAutoField(primary\_key=True)

DELIVERY\_POINTS = [

('evelyhone', 'Evelyn Hone College'),

('zambia\_police', 'Zambia Police Headquarters'),

('zambia\_accountancy', 'Zambia Centre for Accountancy'),

('mukuba\_house', 'Mukuba Pension House'),

('bus\_terminus', 'Lusaka Intercity Bus Terminus'),

('national\_museum', 'Lusaka National Museum'),

]

PAYMENT\_METHODS = [

('cash', 'Cash'),

('mobile\_money', 'Mobile Money'),

('card', 'Card'),

]

MOBILE\_MONEY\_PROVIDERS = [

('airtel', 'Airtel'),

('mtn', 'MTN'),

('zamtel', 'Zamtel'),

]

CARD\_PROVIDERS = [

('paypal', 'Paypal'),

('pesapal', 'Pesapal'),

('stripe', 'Stripe'),

]

DELIVERY\_STATUS\_CHOICES = [

('pending', 'Pending'),

('in\_progress', 'In Progress'),

('completed', 'Completed'),

('cancelled', 'Cancelled'),

]

user = models.ForeignKey(User, on\_delete=models.CASCADE)

cart = models.ForeignKey(Cart, on\_delete=models.CASCADE)

address = models.CharField(max\_length=100, blank=True, null=True)

predefined\_address = models.CharField(

max\_length=30,

choices=DELIVERY\_POINTS,

blank=True,

null=True,

help\_text="Predefined delivery point"

)

delivery\_status = models.CharField(

max\_length=20,

choices=DELIVERY\_STATUS\_CHOICES,

default='pending',

help\_text="Current delivery status"

)

payment\_method = models.CharField(

max\_length=20,

choices=PAYMENT\_METHODS,

default='cash',

help\_text="Payment method used"

)

payment\_provider = models.CharField(

max\_length=20,

blank=True,

null=True,

help\_text="Specific provider for Mobile Money or Card (e.g., Airtel, Stripe)"

)

phone\_number = models.CharField(max\_length=20)

secondary\_phone\_number = models.CharField(

max\_length=20,

blank=True,

null=True,

help\_text="Optional secondary contact number"

)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

def save(self, \*args, \*\*kwargs):

if self.pk:

try:

old\_instance = DeliveryInfo.objects.get(pk=self.pk)

if old\_instance.delivery\_status != self.delivery\_status:

logger.info(

f"DeliveryInfo {self.id} status changed from "

f"{old\_instance.delivery\_status} to {self.delivery\_status}"

)

except DeliveryInfo.DoesNotExist:

logger.warning(f"DeliveryInfo {self.id} not found during save")

else:

logger.info(f"New DeliveryInfo created with status {self.delivery\_status}")

if self.predefined\_address and not self.address:

self.address = self.get\_predefined\_address\_display()

super().save(\*args, \*\*kwargs)

def \_\_str\_\_(self):

address\_display = self.address or self.get\_predefined\_address\_display() or "N/A"

return f"Delivery for {self.user.username} at {address\_display}"

class Meta:

ordering = ['-created\_at']

verbose\_name = "Delivery Info"

verbose\_name\_plural = "Delivery Info"

class PaymentHistory(models.Model):

id = models.BigAutoField(primary\_key=True)

transaction\_id = models.CharField(max\_length=100, blank=True, null=True) # paypal transaction

user = models.ForeignKey(User, on\_delete=models.CASCADE)

cart = models.ForeignKey(Cart, on\_delete=models.SET\_NULL, null=True)

delivery\_info = models.ForeignKey(

DeliveryInfo,

on\_delete=models.SET\_NULL,

null=True,

related\_name='payment\_histories'

)

total = models.DecimalField(max\_digits=10, decimal\_places=2)

created\_at = models.DateTimeField(auto\_now\_add=True)

items = models.JSONField(

help\_text="List of items in the payment (e.g., [{'name': 'Pizza', 'quantity': 2, 'subtotal': 15.00}])"

)

def \_\_str\_\_(self):

return f"Payment by {self.user.username} on {self.created\_at}"

class Meta:

ordering = ['-created\_at']

verbose\_name = "Payment History"

verbose\_name\_plural = "Payment Histories"

from django.db import models

from auths.models import User

from payments.models import DeliveryInfo

class StaffServiceArea(models.Model):

"""

Maps staff members to the predefined delivery points they can serve.

"""

staff = models.ForeignKey(

User,

on\_delete=models.CASCADE,

limit\_choices\_to={'user\_type': 'staff'},

related\_name='service\_areas'

)

point = models.CharField(

max\_length=30,

choices=DeliveryInfo.DELIVERY\_POINTS,

help\_text="Which delivery point this staff covers"

)

class Meta:

unique\_together = ('staff', 'point')

verbose\_name = "Staff Service Area"

verbose\_name\_plural = "Staff Service Areas"

def \_\_str\_\_(self):

return f"{self.staff.username} covers {self.get\_point\_display()}"

class StaffAssignment(models.Model):

staff = models.ForeignKey(

User,

on\_delete=models.CASCADE,

limit\_choices\_to={'user\_type': 'staff'},

related\_name='delivery\_assignments'

)

delivery = models.ForeignKey(

DeliveryInfo,

on\_delete=models.CASCADE,

related\_name='staff\_assignments'

)

assigned\_at = models.DateTimeField(auto\_now\_add=True)

class Meta:

unique\_together = ['staff', 'delivery']

verbose\_name = "Staff Assignment"

verbose\_name\_plural = "Staff Assignments"

def \_\_str\_\_(self):

return f"{self.staff.username} assigned to Delivery {self.delivery.id}"

class Notification(models.Model):

NOTIFICATION\_TYPES = [

('new\_order', 'New Order'),

('delivery\_almost\_complete', 'Delivery Almost Complete'),

('delivery\_completed', 'Delivery Completed'),

('alert', 'Alert'),

('delivery\_declined', 'Delivery Declined'),

]

recipient = models.ForeignKey(

User,

on\_delete=models.CASCADE,

limit\_choices\_to={'user\_type': 'staff'},

related\_name='notifications'

)

message = models.TextField()

notification\_type = models.CharField(

max\_length=30,

choices=NOTIFICATION\_TYPES,

default='new\_order'

)

is\_read = models.BooleanField(default=False)

created\_at = models.DateTimeField(auto\_now\_add=True)

related\_delivery = models.ForeignKey(

DeliveryInfo,

on\_delete=models.SET\_NULL,

null=True,

blank=True,

related\_name='notifications'

)

def \_\_str\_\_(self):

return f"Notification for {self.recipient.username}: {self.message}"

class Meta:

ordering = ['-created\_at']

verbose\_name = "Notification"

verbose\_name\_plural = "Notifications"