Subscription Payment & Invoice System — Documentation + Code

Overview

This document provides a complete Django implementation blueprint for a subscription & add-on management system, including payments, invoices, admin views, and templates. The code uses Django's built-in User model (no custom user). Save the code snippets into your Django app (replace 'your_app' with your app name) and follow instructions.

Models (models.py)

```
from django.db import models
from django.conf import settings
from django.utils import timezone

class SubscriptionPlan2(models.Model):
    PLAN_TYPE_CHOICES = [
        ('subscription', 'Subscription'),
        ('addon', 'Add-On Service'),
    ]
    name = models.CharField(max_length=200)
    plantype = models.CharField(max_length=20, choices=PLAN_TYPE_CHOICES)
    description = models.TextField(blank=True, null=True)
    durationdays = models.IntegerField(default=30)
    baseprice = models.DecimalField(max_digits=10, decimal_places=2)
    discountpercent = models.DecimalField(max_digits=5, decimal_places=2, default=0)
    offerprice = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    offerstartdate = models.DateField(null=True, blank=True)
    offerenddate = models.DateField(null=True, blank=True)
    maxlistings = models.PositiveIntegerField(default=1)
    isactive = models.BooleanField(default=True)
    createdat = models.DateTimeField(auto_now_add=True)
    updatedat = models.DateTimeField(auto_now=True)

    def calculate_offer_price(self):
        if self.discountpercent > 0:
            return round(self.baseprice - (self.baseprice * self.discountpercent / 100), 2)
        return self.baseprice

    def is_offer_active(self):
        today = timezone.now().date()
        return (self.offerstartdate and self.offerenddate
                and self.offerstartdate <= today <= self.offerenddate)

    def __str__(self):
        return f"{self.name} [{self.plantype}]"

class UserSubscription2(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    plan = models.ForeignKey(SubscriptionPlan2, on_delete=models.CASCADE)
    activated_on = models.DateField(auto_now_add=True)
    expires_on = models.DateField()
    active = models.BooleanField(default=True)
    listings_used = models.PositiveIntegerField(default=0)

    def is_active(self):
        return self.active and self.expires_on >= timezone.now().date()

    def can_post_listing(self):
        return self.is_active() and self.listings_used < self.plan.maxlistings

class PaymentTransaction(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    plan = models.ForeignKey(SubscriptionPlan2, on_delete=models.CASCADE)
    amount_paid = models.DecimalField(max_digits=10, decimal_places=2)
    payment_method = models.CharField(max_length=50, default="Online")
    transaction_id = models.CharField(max_length=100, unique=True)
    payment_date = models.DateTimeField(default=timezone.now)
    status = models.CharField(max_length=20, default="Success")

    def __str__(self):
        return f"{self.user.username} - {self.plan.name} - {self.amount_paid}"

class Invoice(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    payment = models.OneToOneField(PaymentTransaction, on_delete=models.CASCADE)
    invoice_no = models.CharField(max_length=30, unique=True)
    generated_date = models.DateTimeField(default=timezone.now)
    total_amount = models.DecimalField(max_digits=10, decimal_places=2)

    def __str__(self):
        return f"Invoice {self.invoice_no} - {self.user.username}"
```

Views (views.py) - key functions

```
from django.shortcuts import render, redirect, get_object_or_404
from django.utils import timezone
from datetime import timedelta
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from .models import SubscriptionPlan2, UserSubscription2, PaymentTransaction, Invoice
import uuid

def plan_list1(request):
    today = timezone.now().date()
    plans = SubscriptionPlan2.objects.filter(isactive=True)
    for plan in plans:
        plan.current_price = plan.offerprice if plan.is_offer_active() else plan.baseprice
    return render(request, "module/plan_list1.html", {"plans": plans})

@login_required
def subscribe1(request, plan_id):
    plan = get_object_or_404(SubscriptionPlan2, pk=plan_id, isactive=True)
    if request.method == "POST":
        user = request.user
        activated_on = timezone.now().date()
        expires_on = activated_on + timedelta(days=plan.durationdays)

        # Payment simulation
        txn = PaymentTransaction.objects.create(
            user=user,
            plan=plan,
            amount_paid=plan.offerprice if plan.is_offer_active() else plan.baseprice,
            payment_method="Online",
            transaction_id=str(uuid.uuid4())[:12],
            status="Success",
        )

        # Create invoice
        invoice_no = f"INV-{uuid.uuid4().hex[:8].upper()}"
        Invoice.objects.create(user=user, payment=txn, invoice_no=invoice_no, total_amount=txn.amount_paid)

        # Deactivate existing subscriptions and activate new
        UserSubscription2.objects.filter(user=user, active=True).update(active=False)
        UserSubscription2.objects.create(user=user, plan=plan, activated_on=activated_on, expires_on=expires

        return render(request, "module/payment_success.html", {"txn": txn, "invoice_no": invoice_no, "plan":
    return render(request, "module/subscribe1.html", {"plan": plan})

@login_required
def user_dashboard1(request):
    sub = UserSubscription2.objects.filter(user=request.user, active=True).order_by('-activated_on').first()
    today = timezone.now().date()
    soon = today + timedelta(days=5)
    payments = PaymentTransaction.objects.filter(user=request.user).order_by('-payment_date')[:5]
    return render(request, "module/dashboard.html", {"subs": sub, "today": today, "soon": soon, "payments":

@login_required
def admin_payments(request):
    if not request.user.is_superuser:
        return redirect('user_dashboard')
    payments = PaymentTransaction.objects.all().order_by('-payment_date')
    return render(request, "module/admin_payments.html", {"payments": payments})

@login_required
def admin_invoices(request):
    if not request.user.is_superuser:
        return redirect('user_dashboard')
    invoices = Invoice.objects.all().order_by('-generated_date')
    return render(request, "module/admin_invoices.html", {"invoices": invoices})
```

URLs (urls.py)

```
from django.urls import path
from . import views

urlpatterns = [
    path('plans/', views.plan_list1, name='plan_list1'),
    path('subscribe/<int:plan_id>/', views.subscribe1, name='subscribe1'),
    path('dashboard/', views.user_dashboard1, name='dashboard'),
    path('property/post/', views.post_property1, name='post_property1'),
    path('admin/payments/', views.admin_payments, name='admin_payments'),
    path('admin/invoices/', views.admin_invoices, name='admin_invoices'),
]
```

Key Templates (templates/module/*.html)

Below are snippets of templates. Save each as specified (templates/module/.html).

```
<!-- subscribe1.html -->
<form method="post">{% csrf_token %}
  <h3>Subscribe to {{ plan.name }}</h3>
  <p>Price: ■{{ plan.offerprice if plan.is_offer_active else plan.baseprice }}</p>
  <button type="submit">Pay & Subscribe</button>
</form>
```

```
<!-- payment_success.html -->
<h3>Payment Successful</h3>
<p>Transaction ID: {{ txn.transaction_id }}</p>
<p>Invoice No: {{ invoice_no }}</p>
<p>Amount: ■{{ txn.amount_paid }}</p>
<a href="{% url 'dashboard' %}">Go to Dashboard</a>
```

```
<!-- admin_payments.html -->
<table>
  <tr><th>User</th><th>Plan</th><th>Amount</th><th>Txn</th><th>Date</th></tr>
  {% for p in payments %}
  <tr><td>{{ p.user.username }}</td><td>{{ p.plan.name }}</td><td>{{ p.amount_paid }}</td><td>{{ p.transacti
  {% endfor %}
</table>
```

Management Command: send_expiry_notifications

```
# your_app/management/commands/send_expiry_notifications.py
from django.core.management.base import BaseCommand
from django.utils import timezone
from datetime import timedelta
from your_app.models import UserSubscription2
from django.core.mail import send_mail

class Command(BaseCommand):
    help = 'Send expiry reminder emails and deactivate expired subscriptions'

    def handle(self, *args, **options):
        today = timezone.now().date()
        soon = today + timedelta(days=5)
        about_to_expire = UserSubscription2.objects.filter(expires_on__lte=soon, active=True)
        for sub in about_to_expire:
            send_mail(
                "Subscription Expiring Soon",
                f"Dear {sub.user.username}, your subscription ({sub.plan.name}) will expire on {sub.expires_
                "support@example.com",
                [sub.user.email],
                fail_silently=True,
            )
        expired = UserSubscription2.objects.filter(expires_on__lt=today, active=True)
        expired.update(active=False)
```

Notes & Next Steps

- For real payments integrate Razorpay, Stripe or PayPal in subscribe1 and handle webhooks. - To generate PDF invoices use ReportLab or WeasyPrint; create a view that returns a PDF response. - Secure views: ensure @login_required and is_superuser checks for admin pages. - Configure EMAIL_BACKEND in settings.py for production emails.