**Phase 1 - Admin Dashboard**

**1. SOURCE CODE:**

**1.1. views.py**

from .forms import ConnectionForm, CustomNotificationForm, PlanForm

from .models import Plan, Subscription, CustomNotification, Connection

from user.models import User

from django.http.response import JsonResponse

from django.shortcuts import redirect, render

from django.views import View

from django.contrib.auth.mixins import LoginRequiredMixin

from django.db.models import F, Count

from channels.layers import get\_channel\_layer

from asgiref.sync import async\_to\_sync

import json

**# home page view**

class HomeView(LoginRequiredMixin, View):

template\_name = "main/home.html"

def get\_plan\_list(self):

return Plan.objects.all()

def get\_active\_user(self, active):

return User.objects.filter(is\_active=active).count()

def get\_subscription(self,active):

return Subscription.objects.filter(is\_active=active).count()

def get\_plans(self):

return Plan.objects.filter().count()

def get(self, request, \*args, \*\*kwargs):

if request.user.is\_superuser:

return render(

request,

self.template\_name,

{

"active\_user": self.get\_active\_user(True),

"non\_active\_user": self.get\_active\_user(False),

'subscription\_active': self.get\_subscription(True),

'subscription\_not\_active' : self.get\_subscription(True),

'number\_of\_plans':self.get\_plans()

},

)

return render(request, self.template\_name, {"plans": self.get\_plan\_list()})

**# designer view**

class DesignerView(LoginRequiredMixin, View):

template\_name = "main/designer.html"

def get\_list(self):

return User.objects.filter(user\_type="DS").exclude(is\_superuser=True)

def get(self, request, \*args, \*\*kwargs):

return render(request, self.template\_name, {"designers": self.get\_list()})

**# user view**

class UserView(LoginRequiredMixin, View):

template\_name = "main/user.html"

def get\_list(self):

return User.objects.filter(user\_type="NU").exclude(is\_superuser=True)

def get(self, request, \*args, \*\*kwargs):

return render(request, self.template\_name, {"users": self.get\_list()})

**# plan view**

class PlanView(LoginRequiredMixin, View):

template\_name = "main/plan.html"

def get\_list(self):

return Plan.objects.all()

def get(self, request, \*args, \*\*kwargs):

return render(request, self.template\_name, {"plans": self.get\_list()})

class PlanAddView(LoginRequiredMixin, View):

template\_name = "main/add\_plan.html"

def get(self, request, \*args, \*\*kwargs):

form = PlanForm()

return render(request, self.template\_name, {"form": form})

def post(self, request, \*args, \*\*kwargs):

try:

form = PlanForm(request.POST)

if form.is\_valid():

form.save()

return redirect("plan")

return render(request, self.template\_name, {"form": form})

except Exception as e:

return render(request, self.template\_name, {"form": form})

**#Subscription View**

class SubscriptionView(LoginRequiredMixin, View):

template\_name = "main/subscription.html"

def get\_list(self):

return Subscription.objects.all()

def get(self, request, \*args, \*\*kwargs):

return render(request, self.template\_name, {"subscriptions": self.get\_list()})

Description: Views are Python functions or classes that receive a web request and deliver a web response in the Django framework, as shown in the following source code from Django views.py files. The possible responses can be a plain HTTP response, an HTML template response, or an HTTP redirect response that sends the user to another page.

**1.2. Models.py**

from enum import unique

from django.db.models import BooleanField, DateTimeField, ImageField, TextField

from django.db.models.fields import TextField

from user.models import User

from django.db import models

from django.db.models import CASCADE

from common.models import BasicBaseModel

from django.db.models import CharField, IntegerField

**# importing DateTime module**

from datetime import datetime, timedelta

request\_status\_choices = [('Pending',"PD"),("Accepted","AD")]

class Plan(BasicBaseModel):

name = CharField(max\_length=30)

number\_of\_days = IntegerField(null=False, blank=False)

price = IntegerField(null=False, blank=False)

class Subscription(BasicBaseModel):

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

plan = models.ForeignKey(Plan, on\_delete=CASCADE)

start\_timestamp = models.DateTimeField(auto\_now\_add=True)

end\_timestamp = DateTimeField(blank=False, null=True)

payment\_method = CharField(max\_length=30)

is\_active = BooleanField(default=False, null=False, blank=False)

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

self.end\_timestamp = datetime.now() + timedelta(days=self.plan.number\_of\_days)

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

**Description:** Models are Python objects that Django web applications use to access and manage data. Models describe the structure of stored data, including field types and, in some cases, their maximum size, default values, selection list options, documentation help text, and form label text. Models.py has classes named plan and subscription in the following code, where the plan keeps information such as plan name, number of days, and price for each plan. In the case of the subscription class, it holds information such as the user, the type of plan selected, as well as the start and end dates of the subscription.

**2. CODE SNIPPETS of Core Functionalities:**

**2.1. Home page view**

class HomeView(LoginRequiredMixin, View):

template\_name = "main/home.html"

def get\_plan\_list(self):

return Plan.objects.all()

def get\_active\_user(self, active):

return User.objects.filter(is\_active=active).count()

def get\_subscription(self,active):

return Subscription.objects.filter(is\_active=active).count()

def get\_plans(self):

return Plan.objects.filter().count()

def get(self, request, \*args, \*\*kwargs):

if request.user.is\_superuser:

return render(

Request,

self.template\_name,

{

"active\_user": self.get\_active\_user(True),

"non\_active\_user": self.get\_active\_user(False),

'subscription\_active': self.get\_subscription(True),

'subscription\_not\_active' : self.get\_subscription(True),

'number\_of\_plans':self.get\_plans()

},

)

return render(request, self.template\_name, {"plans": self.get\_plan\_list()})

**Description:** We have a class named HomeView in the preceding code that renders the application's homepage. The functions in this class provide the activities that an admin can perform while visiting the home page, such as viewing plans, checking active users (which reveals if a user is active or not), and seeing subscriptions. View the list of the plans in the same way that the number of plans is displayed.

**2.2. Designer view**

class DesignerView(LoginRequiredMixin, View):

template\_name = "main/designer.html"

def get\_list(self):

return User.objects.filter(user\_type="DS").exclude(is\_superuser=True)

def get(self, request, \*args, \*\*kwargs):

return render(request, self.template\_name, {"designers": self.get\_list()})

Description:

In the previous code, we have a class named DesignerView that renders Designer profiles that have been registered with our application. The functions in this class provide the tasks that an admin can perform while accessing this feature, such as browsing a list of registered designers, checking their details such as id, whether the designer is active or not, and the plan they have selected.

**2.3. User view**

class UserView(LoginRequiredMixin, View):

template\_name = "main/user.html"

def get\_list(self):

return User.objects.filter(user\_type="NU").exclude(is\_superuser=True)

def get(self, request, \*args, \*\*kwargs):

return render(request, self.template\_name, {"users": self.get\_list()})

**Description:**

We have a class named UserView in the following code sample that renders user-profiles registered with our application. The functions in this class give actions that an admin can accomplish while using this feature, such as reading a list of registered users, checking their details like id, whether the designer is active or not, and the plan they have chosen.

**2.4. Plan view**

class PlanView(LoginRequiredMixin, View):

template\_name = "main/plan.html"

def get\_list(self):

return Plan.objects.all()

def get(self, request, \*args, \*\*kwargs):

return render(request, self.template\_name, {"plans": self.get\_list()})

#plan add view

class PlanAddView(LoginRequiredMixin, View):

template\_name = "main/add\_plan.html"

def get(self, request, \*args, \*\*kwargs):

form = PlanForm()

return render(request, self.template\_name, {"form": form})

def post(self, request, \*args, \*\*kwargs):

try:

form = PlanForm(request.POST)

if form.is\_valid():

form.save()

return redirect("plan")

return render(request, self.template\_name, {"form": form})

except Exception as e:

return render(request, self.template\_name, {"form": form})

**Description:** In the following code sample, the classes named Planview and PlanAddView renders plans registered by users while registering with our application. The functions in this class describe actions that an admin can take while using this feature, such as reading a list of available plans for users, such as silver, gold, and diamond plans, each with a different validity period, such as 3, 6, or 12 months, and then viewing or adding any of these plans to the users who have registered with them.

**2.5. Subscription View**

class SubscriptionView(LoginRequiredMixin, View):

template\_name = "main/subscription.html"

def get\_list(self):

return Subscription.objects.all()

def get(self, request, \*args, \*\*kwargs):

return render(request, self.template\_name, {"subscriptions": self.get\_list()})

Description: The class SubscriptionView in the preceding code sample enables the administrator to view subscriptions in the applications. When an administrator requests to view a subscription, the programme accepts the request and searches the database for the subscription, which is then displayed on the dashboard.

**2.6. Plan model**

class Plan(BasicBaseModel):

name = CharField(max\_length=30)

number\_of\_days = IntegerField(null=False, blank=False)

price = IntegerField(null=False, blank=False)

**Description:** The plan class in the above code stores information about each plan, such as its name, number of days, and price. Users can choose from three different plans: silver, which has a three-month validity, gold, which has a six-month validity, and diamond, which has a 12-month validity.

**2.7. Subscription model**

class Subscription(BasicBaseModel):

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

plan = models.ForeignKey(Plan, on\_delete=CASCADE)

start\_timestamp = models.DateTimeField(auto\_now\_add=True)

end\_timestamp = DateTimeField(blank=False, null=True)

payment\_method = CharField(max\_length=30)

is\_active = BooleanField(default=False, null=False, blank=False)

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

self.end\_timestamp = datetime.now() + timedelta(days=self.plan.number\_of\_days)

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

**Description:** The subscription class in the following code contains information such as the userid, the plan type selected, and the subscription's start and end dates.