Digital Local OTT Platform

Sample Code:

Views:

from datetime import timedelta  
  
from django.shortcuts import render, redirect, get\_object\_or\_404  
from django.utils import timezone  
  
from .forms import ContactForm, CustomerForm, NotificationForm, CategoryForm, MovieForm, WebseriesForm, SerialsForm, \  
 EpisodeForm, SerialEpisodeForm, MovieReviewForm, SerialReviewForm, WebseriesReviewForm, SubscriptionForm, \  
 SubscribersForm  
from .models import Customer, AdminLogin, Notification, Category, Movie, Webseries, Serials, Episode, Serials\_Episodes, \  
 Movie\_Review, Webseries\_Review, Serial\_Review, Subscription, Subscribers  
  
  
# Create your views here.  
def index(request):  
 return render(request, "index.html", {})  
  
  
def about(request):  
 profile = Movie.objects.all()  
 return render(request, "about.html", {"profile": profile})  
  
  
def services(request):  
 return render(request, "services.html", {})  
  
  
def apartment(request):  
 return render(request, "apartment.html", {})  
  
  
def blog(request):  
 return render(request, "blog.html", {})  
  
  
def elements(request):  
 return render(request, "elements.html", {})  
  
  
def project(request):  
 return render(request, "project.html", {})  
  
  
def single\_blog(request):  
 return render(request, "single-blog.html", {})  
  
  
def contact(request):  
 if request.method == "POST":  
 form = ContactForm(request.POST)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return render(request, "contact.html", {"msg": "Contact Sent Successfully"})  
 return render(request, "contact.html", {"psg": "Contact Not Sent...Please Try Again"})  
 return render(request, "contact.html", {})  
  
  
# def customer\_login(request):  
# if request.method == 'POST':  
# email = request.POST['email']  
# password = request.POST['password']  
# log = Customer.objects.filter(email=email, password=password)  
# if log.exists():  
# if log[0].status == "Accepted":  
# request.session['email'] = email  
# return render(request, "customer\_home.html", {"msg": "Accepted Successfully"})  
# elif log[0].status == "Rejected":  
# return render(request, "customer\_login.html", {"msg": "Rejected Login"})  
# else:  
# return render(request, "customer\_login.html", {"msg": "Pending"})  
# return render(request, "customer\_login.html", {"msg": "Login Failed"})  
# return render(request, "customer\_login.html", {})  
  
  
def customer\_login(request):  
 if request.method == "POST":  
 email = request.POST["email"]  
 password = request.POST["password"]  
 print(email, "", password)  
 login = Customer.objects.filter(email=email, password=password)  
 if login.exists():  
 print("hi")  
 if login[0].status == "Accepted":  
 request.session['email'] = email  
 client = Customer.objects.get(email=email)  
 return render(request, "customer\_home.html", {"psg": "Login Successfully"})  
 else:  
 return render(request, "customer\_login.html", {"msg": "Your Account Is On Hold !"})  
 else:  
 return render(request, "customer\_login.html", {"msg": "Invalid Data"})  
 return render(request, "customer\_login.html", {})  
  
  
# def customer\_registered(request):  
# if request.method == "POST":  
# form = CustomerForm(request.POST, request.FILES)  
# print(form.errors)  
# if form.is\_valid():  
# form.save()  
# return render(request, "customer\_login.html", {"msg": "Your "})  
# return render(request, "customer\_registered.html", {"msg": "Not Registered"})  
# return render(request, "customer\_registered.html", {})  
  
def customer\_registered(request):  
 if request.method == "POST":  
 form = CustomerForm(request.POST, request.FILES)  
 print(form.errors)  
 email = request.POST["email"]  
 if Customer.objects.filter(email=email).exists():  
 return render(request, "customer\_registered.html", {"psg": "This Email Is Already Exists"})  
 else:  
 if form.is\_valid():  
 form.save()  
 return render(request, "customer\_login.html", {"msg": "Your Details Is Register Successfully"})  
 return render(request, "customer\_registered.html", {})  
  
  
def customer\_home(request):  
 return render(request, "customer\_home.html", {})  
  
  
def customer\_view\_movies(request):  
 email = request.session.get('email') # Get the customer's email from the session  
 movies = Movie.objects.all() # Fetch all movies  
  
 if not email: # Check if the user is logged in or has an email session  
 return redirect('login') # Redirect to login page if not logged in  
  
 # Get customer and their current subscription if available  
 customer = get\_object\_or\_404(Customer, email=email)  
 active\_subscription = Subscribers.objects.filter(  
 customer\_email=email,  
 startdate\_\_lte=timezone.now().date(),  
 enddate\_\_gte=timezone.now().date()  
 ).first()  
  
 # Determine if the user is eligible to watch movies  
 eligible\_to\_watch = active\_subscription is not None  
  
 return render(request, "customer\_view\_movies.html", {  
 "movies": movies,  
 "eligible\_to\_watch": eligible\_to\_watch  
 })  
  
  
def customer\_view\_webseries(request):  
 email = request.session.get('email') # Get the customer's email from the session  
 profile = Webseries.objects.all() # Fetch all web series  
  
 if not email: # Check if the user is logged in or has an email session  
 return redirect('login') # Redirect to login page if not logged in  
  
 # Get customer and their current subscription if available  
 customer = get\_object\_or\_404(Customer, email=email)  
 active\_subscription = Subscribers.objects.filter(  
 customer\_email=email,  
 startdate\_\_lte=timezone.now().date(),  
 enddate\_\_gte=timezone.now().date()  
 ).first()  
  
 # Determine if the user is eligible to view episodes  
 eligible\_to\_watch = active\_subscription is not None  
  
 return render(request, "customer\_view\_webseries.html", {  
 "profile": profile,  
 "eligible\_to\_watch": eligible\_to\_watch  
 })  
  
  
def customer\_view\_serials(request):  
 email = request.session.get('email') # Get the customer's email from the session  
 profile = Serials.objects.all() # Fetch all serials  
  
 if not email: # Check if the user is logged in or has an email session  
 return redirect('login') # Redirect to login page if not logged in  
  
 # Get customer and their current subscription if available  
 customer = get\_object\_or\_404(Customer, email=email)  
 active\_subscription = Subscribers.objects.filter(  
 customer\_email=email,  
 startdate\_\_lte=timezone.now().date(),  
 enddate\_\_gte=timezone.now().date()  
 ).first()  
  
 # Determine if the user is eligible to view serials  
 eligible\_to\_watch = active\_subscription is not None  
  
 return render(request, "customer\_view\_serials.html", {  
 "profile": profile,  
 "eligible\_to\_watch": eligible\_to\_watch  
 })  
  
  
def customer\_profile(request):  
 email = request.session.get('email') # Get customer email from session  
 profile = get\_object\_or\_404(Customer, email=email) # Fetch customer profile  
  
 # Fetch all subscriptions for the customer  
 all\_subscriptions = Subscribers.objects.filter(customer\_email=email)  
  
 # Filter active subscriptions (where end date is in the future or today)  
 active\_subscription = all\_subscriptions.filter(enddate\_\_gte=timezone.now().date()).order\_by('-enddate').first()  
  
 # Filter expired subscriptions (where end date is in the past)  
 expired\_subscriptions = all\_subscriptions.filter(enddate\_\_lt=timezone.now().date()).order\_by('-enddate')  
  
 return render(request, "customer\_profile.html", {  
 "x": profile,  
 "active\_subscription": active\_subscription, # Active subscription details  
 "expired\_subscriptions": expired\_subscriptions, # Expired subscriptions for history  
 })  
  
def customer\_logout(request):  
 request.session.flush()  
 return redirect('/customerlogin')  
  
  
def admin\_login(request):  
 if request.method == "POST":  
 email = request.POST["email"]  
 password = request.POST["password"]  
 log = AdminLogin.objects.filter(email=email, password=password)  
 try:  
 if log.exists():  
 request.session["email"] = email  
 return render(request, "admin\_home.html", {"msg": "Successfully Login"})  
 return render(request, "admin\_login.html", {"msg": "Incorrect Email Or Password"})  
 except Exception as e:  
 print(e)  
 return render(request, "admin\_home.html", {"msg": ""})  
 return render(request, "admin\_login.html", {"msg": ""})  
  
  
def admin\_home(request):  
 return render(request, "admin\_home.html", {})  
  
  
def admin\_view\_contact(request):  
 contact = Customer.objects.all()  
 return render(request, "admin\_view\_contact.html", {"contact": contact})  
  
  
def admin\_view\_customer(request):  
 customer = Customer.objects.all()  
 return render(request, "admin\_view\_customer.html", {"customer": customer})  
  
  
def accept\_customer(request, id):  
 cus = Customer.objects.get(id=id)  
 cus.status = "Accepted"  
 cus.save()  
 return redirect('/admin\_view\_customer')  
  
  
def reject\_customer(request, id):  
 cus = Customer.objects.get(id=id)  
 cus.status = "Rejected"  
 cus.save()  
 return redirect('/admin\_view\_customer')  
  
  
def admin\_logout(request):  
 request.session.flush()  
 return redirect('/admin\_login')  
  
  
def customer\_edit(request):  
 email = request.session['email']  
 customer = Customer.objects.get(email=email)  
 return render(request, 'customer\_edit.html', {'x': customer})  
  
  
def customer\_update(request):  
 if request.method == 'POST':  
 id = request.POST['id']  
 cus = Customer.objects.get(id=id)  
 form = CustomerForm(request.POST, request.FILES, instance=cus)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('/customer\_profile')  
 return render(request, 'customer\_edit.html', {})  
 return render(request, 'customer\_edit.html', {})  
  
  
def is\_login(request):  
 if request.session.\_\_contains\_\_('email'):  
 return True  
 else:  
 return False  
  
  
def customer\_change\_password(request):  
 email = request.session['email']  
 if is\_login(request):  
 if request.method == 'POST':  
  
 password = request.POST['password']  
 new\_password = request.POST['new\_password']  
 try:  
 cust = Customer.objects.get(email=email, password=password)  
 cust.password = new\_password  
 cust.save()  
 return redirect('/customer\_login')  
 except Exception as e:  
 print(e)  
 return render(request, "customer\_change\_password.html", {"msg": "Invalid Credentials", "email": email})  
 return render(request, "customer\_change\_password.html", {"email": email})  
 return render(request, "customer\_change\_password.html", {"email": email})  
  
  
def customer\_deactivate(request, id):  
 cust = Customer.objects.get(id=id)  
 cust.status = 'pending'  
 cust.save()  
 return redirect('/customerlogin')  
  
  
# def admin\_change\_password(request):  
# email = request.session['email']  
# if (request):  
# if request.method == 'POST':  
#  
# password = request.POST['password']  
# new\_password = request.POST['new\_password']  
# try:  
# cust = Customer.objects.get(email=email, password=password)  
# cust.password = new\_password  
# cust.save()  
# return redirect('/admin\_login')  
# except Exception as e:  
# print(e)  
# return render(request, "admin\_change\_password.html",  
# {"msg": "Your Password Not Changed, Please Try Again", "email": email})  
# return render(request, "admin\_change\_password.html", {"email": email})  
# return render(request, "admin\_change\_password.html", {"email": email})  
  
  
def admin\_change\_password(request):  
 email = request.session["email"]  
 if is\_login(request):  
 print("hello")  
 if request.method == 'POST':  
 print("hello1")  
 password = request.POST['password']  
 new\_password = request.POST['new\_password']  
 try:  
 user = AdminLogin.objects.get(email=email, password=password)  
 user.password = new\_password  
 user.save()  
 return redirect('/admin\_login')  
 except Exception as e:  
 print(e)  
 return render(request, "admin\_change\_password.html",  
 {"msg": "Your Password Not Changed, Please Try Again", "email": email})  
 return render(request, "admin\_change\_password.html", {"email": email})  
 return render(request, "admin\_change\_password.html", {"email": email})  
  
  
def add\_notification(request):  
 email = request.session['email']  
 if request.method == 'POST':  
 form = NotificationForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_notification')  
 return render(request, "add\_notification.html", {"msg":"Notification Not Posted, Please Try Again","email":email})  
 return render(request, "add\_notification.html", {"email":email})  
  
  
def view\_notification(request):  
 profile = Notification.objects.all()  
 return render(request, "view\_notification.html", {"profile": profile})  
  
  
def notification\_update(request):  
 if request.method == 'POST':  
 id = request.POST['id']  
 cus = Notification.objects.get(id=id)  
 form = NotificationForm(request.POST, request.FILES, instance=cus)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('/view\_notification')  
 return render(request, "notification\_edit.html", {})  
 return render(request, "notification\_edit.html", {})  
  
  
def notification\_edit(request, id):  
 profile = Notification.objects.get(id=id)  
 return render(request, "notification\_edit.html", {'x': profile})  
  
  
def view\_notification\_delete(request, id):  
 cus = Notification.objects.get(id=id)  
 cus.delete()  
 return redirect('/view\_notification')  
  
  
def admin\_add\_category(request):  
 if request.method == 'POST':  
 form = CategoryForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('admin\_view\_category')  
 return render(request, "admin\_add\_category.html", {})  
 return render(request, "admin\_add\_category.html", {})  
  
  
def admin\_view\_category(request):  
 profile = Category.objects.all()  
 return render(request, "admin\_view\_category.html", {"profile": profile})  
  
  
def category\_update(request):  
 if request.method == 'POST':  
 id = request.POST['id']  
 cus = Category.objects.get(id=id)  
 form = CategoryForm(request.POST, request.FILES, instance=cus)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('/admin\_view\_category')  
 return render(request, "category\_edit.html", {})  
 return render(request, "category\_edit.html", {})  
  
  
def category\_edit(request, id):  
 profile = Category.objects.get(id=id)  
 return render(request, "category\_edit.html", {'x': profile})  
  
  
def view\_category\_delete(request, id):  
 cus = Category.objects.get(id=id)  
 cus.delete()  
 return redirect('/admin\_view\_category')  
  
  
def add\_movie(request):  
 if request.method == 'POST':  
 form = MovieForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_movie')  
 return render(request, "add\_movie.html", {"msg":"Movie Not Added , Please Try Again"})  
 return render(request, "add\_movie.html", {})  
  
  
def view\_movie(request):  
 profile = Movie.objects.all()  
 return render(request, "view\_movie.html", {"profile": profile})  
  
  
def movie\_update(request):  
 if request.method == 'POST':  
 id = request.POST['id']  
 cus = Movie.objects.get(id=id)  
 form = MovieForm(request.POST, request.FILES, instance=cus)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('/view\_movie')  
 return render(request, 'movie\_edit.html', {})  
 return render(request, 'movie\_edit.html', {})  
  
  
def movie\_edit(request, id):  
 profile = Movie.objects.get(id=id)  
 return render(request, 'movie\_edit.html', {'x': profile})  
  
  
def view\_movie\_delete(request, id):  
 cus = Movie.objects.get(id=id)  
 cus.delete()  
 return redirect('/view\_movie')  
  
  
def add\_web\_series(request):  
 if request.method == 'POST':  
 form = WebseriesForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_webseries')  
 return render(request, "add\_webseries.html", {"msg": "Webseries Not Added , Please Try Again"})  
 return render(request, "add\_webseries.html", {})  
  
  
def view\_webseries(request):  
 profile = Webseries.objects.all()  
 return render(request, "view\_webseries.html", {"profile": profile})  
  
  
def webseries\_update(request):  
 if request.method == 'POST':  
 id = request.POST['id']  
 cus = Webseries.objects.get(id=id)  
 form = WebseriesForm(request.POST, request.FILES, instance=cus)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('/view\_webseries')  
 return render(request, "webseries\_edit.html", {})  
 return render(request, "webseries\_edit.html", {})  
  
  
def webseries\_edit(request, id):  
 pro = Webseries.objects.get(id=id)  
 return render(request, "webseries\_edit.html", {'x': pro})  
  
  
def view\_webseries\_delete(request, id):  
 cus = Webseries.objects.get(id=id)  
 cus.delete()  
 return redirect('/view\_webseries')  
  
  
def add\_serials(request):  
 if request.method == 'POST':  
 form = SerialsForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_serials')  
 return render(request, "add\_serials.html", {"psg": "Serial Not Added...Please Try Again"})  
 return render(request, "add\_serials.html", {})  
  
  
def view\_serial(request):  
 profile = Serials.objects.all()  
 return render(request, "view\_serials.html", {"profile": profile})  
  
  
def serial\_update(request):  
 if request.method == 'POST':  
 id = request.POST['id']  
 cus = Serials.objects.get(id=id)  
 form = SerialsForm(request.POST, request.FILES, instance=cus)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('/view\_serials')  
 return render(request, "serial\_edit.html", {})  
 return render(request, "serial\_edit.html", {})  
  
  
def serial\_edit(request, id):  
 profile = Serials.objects.get(id=id)  
 return render(request, "serial\_edit.html", {'x': profile})  
  
  
def view\_serial\_delete(request, id):  
 cus = Serials.objects.get(id=id)  
 cus.delete()  
 return redirect('/view\_serials')  
  
  
def webseries\_add\_episode(request, id):  
 web\_series = Webseries.objects.get(id=id)  
 if request.method == 'POST':  
 form = EpisodeForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return render(request, 'view\_webseries\_episodes.html',  
 {'form': form, 'x': web\_series, "msg": "Episode Added"})  
 else:  
 form = EpisodeForm()  
 return render(request, 'webseries\_add\_episodes.html',  
 {'form': form, 'x': web\_series, "psg": "Episode Not Added"})  
 return render(request, 'webseries\_add\_episodes.html', {'x': web\_series, "msg": ""})  
  
  
def view\_webseries\_episodes(request, id):  
 episode = Episode.objects.filter(webseries\_id=id)  
 return render(request, "view\_webseries\_episodes.html", {"episode": episode})  
  
  
def serial\_add\_episode(request, id):  
 serial = Serials.objects.get(id=id)  
 if request.method == 'POST':  
 form = SerialEpisodeForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return render(request, 'view\_serials\_episodes.html',  
 {'form': form, 'x': serial, "msg": "episode added"})  
 else:  
 form = SerialEpisodeForm()  
 return render(request, 'serial\_add\_episode.html',  
 {'form': form, 'x': serial, "msg": "episode not added"})  
 return render(request, 'serial\_add\_episode.html', {'x': serial, "msg": ""})  
  
  
def view\_serials\_episodes(request, id):  
 episode = Serials\_Episodes.objects.filter(serial\_id=id)  
 return render(request, "view\_serials\_episodes.html", {"episode": episode})  
  
  
def customer\_view\_webseries\_episodes(request, id):  
 episode = Episode.objects.filter(webseries\_id=id)  
 return render(request, "customer\_view\_webseries\_episodes.html", {"episode": episode})  
  
  
def customer\_view\_serials\_episodes(request, id):  
 episode = Serials\_Episodes.objects.filter(serial\_id=id)  
 return render(request, "customer\_view\_serials\_episodes.html", {"episode": episode})  
  
  
def add\_movie\_review(request, id):  
 email = request.session['email']  
 movie = Movie.objects.get(id=id)  
 if request.method == 'POST':  
 form = MovieReviewForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_movie\_review')  
 return render(request, "add\_movie\_review.html", {'x': movie, "email": email,"msg":"Review Not Posted ,Please Try Again"})  
 return render(request, "add\_movie\_review.html", {'x': movie, "email": email})  
  
  
def view\_movie\_review(request):  
 profile = Movie\_Review.objects.all()  
 return render(request, "view\_movie\_review.html", {"profile": profile})  
  
  
def add\_webseries\_review(request, id):  
 email = request.session['email']  
 webseries = Webseries.objects.get(id=id)  
 if request.method == 'POST':  
 form = WebseriesReviewForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_webseries\_review')  
 return render(request, "add\_webseries\_review.html", {'x': webseries, "email": email,"msg":"Review Not Posted ,Please Try Again"})  
 return render(request, "add\_webseries\_review.html", {'x': webseries, "email": email})  
  
  
def view\_webseries\_review(request):  
 profile = Webseries\_Review.objects.all()  
 return render(request, "view\_webseries\_review.html", {"profile": profile})  
  
  
def add\_serial\_review(request, id):  
 email = request.session['email']  
 serial = Serials.objects.get(id=id)  
 if request.method == 'POST':  
 form = SerialReviewForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_serial\_review')  
 return render(request, "add\_serial\_review.html", {'x': serial, "email": email,"msg":"Review Not Posted ,Please Try Again"})  
 return render(request, "add\_serial\_review.html", {'x': serial, "email": email})  
  
  
def view\_serial\_review(request):  
 profile = Serial\_Review.objects.all()  
 return render(request, "view\_serial\_review.html", {"profile": profile})  
  
  
def add\_plan(request):  
 email = request.session['email']  
 if request.method == 'POST':  
 form = SubscriptionForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_plan')  
 return render(request, "add\_plan.html", {"email": email,"msg":"Plan Not Added ,Please Try Again"})  
 return render(request, "add\_plan.html", {"email": email})  
  
  
def view\_plan(request):  
 plan = Subscription.objects.all()  
 return render(request, "view\_plan.html", {"plan": plan})  
  
  
def customer\_view\_plans(request):  
 plans = Subscription.objects.all()  
 return render(request, "customer\_view\_plans.html", {"plans": plans})  
  
  
from datetime import timedelta  
from django.shortcuts import render, get\_object\_or\_404  
from django.utils import timezone  
from .models import Subscribers, Subscription, Customer  
from .forms import SubscribersForm  
  
  
def buy\_plan(request, id):  
 email = request.session.get('email') # Get the customer's email from the session  
 plans = get\_object\_or\_404(Subscription, id=id) # Retrieve the selected subscription plan  
 customer = get\_object\_or\_404(Customer, email=email) # Retrieve the customer object  
  
 # Check if the customer has any active subscriptions that are still valid  
 active\_subscriptions = Subscribers.objects.filter(customer\_email=email, enddate\_\_gte=timezone.now().date())  
  
 # If there is an active subscription, show a message and prevent buying a new plan  
 if active\_subscriptions.exists():  
 return render(request, 'buy\_plan.html', {  
 "message": "Already subscribed to this plan. Enjoy your current subscription!"  
 })  
  
 # Auto-calculate start and end dates for the new subscription  
 start\_date = timezone.now().date() # Set the start date to today  
 end\_date = start\_date + timedelta(days=plans.months \* 30) # Calculate the end date based on the plan duration  
  
 if request.method == "POST":  
 form = SubscribersForm(request.POST)  
 if form.is\_valid():  
 # Save the new subscription  
 new\_subscription = form.save(commit=False)  
 new\_subscription.customer\_email = email  
 new\_subscription.subscription = plans  
 new\_subscription.startdate = start\_date  
 new\_subscription.enddate = end\_date  
 new\_subscription.save()  
  
 # Update the customer's startdate and enddate fields  
 customer.startdate = start\_date  
 customer.enddate = end\_date  
 customer.save()  
  
 return render(request, 'buy\_plan.html', {"message": "Thank you for subscribing! Enjoy your plan."})  
 else:  
 print(form.errors)  
  
 # Pass 'startdate' and 'enddate' in context  
 return render(request, 'buy\_plan.html', {  
 "plans": plans,  
 'email': email,  
 'startdate': start\_date,  
 'enddate': end\_date  
 })  
  
def admin\_view\_subscribers(request):  
 # Fetch all subscribers with their related subscription plan  
 subscribers = Subscribers.objects.select\_related('subscription').all()  
  
 # Get the current date to compare subscription status  
 today = timezone.now().date()  
  
 # Render the 'admin\_view\_subscribers.html' template with the list of subscribers  
 return render(request, "admin\_view\_subscribers.html", {"subscribers": subscribers, "today": today})  
  
from django.shortcuts import render, redirect  
from django.utils import timezone  
from .models import Subscribers, Subscription  
  
def customer\_view\_history(request):  
 # Get the customer's email from the session  
 email = request.session.get('email')  
  
 if not email: # Check if the user is logged in or has an email session  
 return redirect('login') # Redirect to login page if not logged in  
  
 # Fetch the customer's subscription history  
 subscriptions = Subscribers.objects.filter(customer\_email=email).select\_related('subscription')  
  
 # Get the current date to compare subscription status  
 today = timezone.now().date()  
  
 # Render the 'customer\_view\_history.html' template with the list of subscriptions  
 return render(request, "customer\_view\_history.html", {  
 "subscriptions": subscriptions,  
 "today": today  
 })

URLS:

*"""  
URL configuration for digital\_ott project.  
  
The `urlpatterns` list routes URLs to views. For more information please see:  
 https://docs.djangoproject.com/en/5.0/topics/http/urls/  
Examples:  
Function views  
 1. Add an import: from my\_app import views  
 2. Add a URL to urlpatterns: path('', views.home, name='home')  
Class-based views  
 1. Add an import: from other\_app.views import Home  
 2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')  
Including another URLconf  
 1. Import the include() function: from django.urls import include, path  
 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))  
"""*from django.conf import settings  
from django.conf.urls.static import static  
from django.contrib import admin  
from django.urls import path  
from dreamapp import views  
  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('',views.index,name="index"),  
 path('about',views.about,name="about"),  
 path('apartment',views.apartment,name="apartment"),  
 path('blog',views.blog,name="blog"),  
 path('elements',views.elements,name="elements"),  
 path('project',views.project,name="project"),  
 path('services',views.services,name="services"),  
 path('single\_blog',views.single\_blog,name="single\_blog"),  
 path('contact',views.contact,name="contact"),  
 path('customerlogin', views.customer\_login,name="customerlogin"),  
 path('customer\_registered', views.customer\_registered,name="customer\_registered"),  
 path('customer\_home', views.customer\_home, name="customer\_home"),  
 path('customer\_profile', views.customer\_profile, name="customer\_profile"),  
 path('customer\_view\_movies', views.customer\_view\_movies, name="customer\_view\_movies"),  
 path('customer\_view\_webseries', views.customer\_view\_webseries, name="customer\_view\_webseries"),  
 path('customer\_view\_serials', views.customer\_view\_serials, name="customer\_view\_serials"),  
 path('customer\_logout', views.customer\_logout, name="customer\_logout"),  
 path('accept\_customer/<int:id>', views.accept\_customer, name="accept\_customer"),  
 path('reject\_customer/<int:id>', views.reject\_customer, name="reject\_customer"),  
 path('customer\_edit', views.customer\_edit, name="customer\_edit"),  
 path('customer\_update', views.customer\_update, name="customer\_update"),  
 path('customer\_change\_password', views.customer\_change\_password, name="customer\_change\_password"),  
 path('admin\_login', views.admin\_login, name="admin\_login"),  
 path('admin\_home', views.admin\_home, name="admin\_home"),  
 path('admin\_view\_contact', views.admin\_view\_contact, name="admin\_view\_contact"),  
 path('admin\_view\_customer', views.admin\_view\_customer, name="admin\_view\_customer"),  
 path('admin\_logout', views.admin\_logout, name="admin\_logout"),  
 path('admin\_change\_password', views.admin\_change\_password, name="admin\_change\_password"),  
 path('customer\_deactivate/<int:id>',views.customer\_deactivate,name="customer\_deactivate"),  
 path('add\_notification',views.add\_notification,name="add\_notification"),  
 path('view\_notification', views.view\_notification, name="view\_notification"),  
 path('notification\_edit/<int:id>', views.notification\_edit, name="notification\_edit"),  
 path('notification\_update', views.notification\_update, name="notification\_update"),  
 path('view\_notification\_delete/<int:id>', views.view\_notification\_delete, name="view\_notification\_delete"),  
 path('admin\_add\_category', views.admin\_add\_category, name="admin\_add\_category"),  
 path('admin\_view\_category', views.admin\_view\_category, name="admin\_view\_category"),  
 path('category\_edit/<int:id>', views.category\_edit, name="category\_edit"),  
 path('category\_update', views.category\_update, name="category\_update"),  
 path('view\_category\_delete/<int:id>', views.view\_category\_delete, name="view\_category\_delete"),  
 path('add\_movie',views.add\_movie,name="add\_movie"),  
 path('view\_movie', views.view\_movie, name="view\_movie"),  
 path('movie\_edit/<int:id>', views.movie\_edit, name="movie\_edit"),  
 path('movie\_update', views.movie\_update, name="movie\_update"),  
 path('view\_movie\_delete/<int:id>', views.view\_movie\_delete, name="view\_movie\_delete"),  
 path('add\_webseries',views.add\_web\_series,name="add\_webseries"),  
 path('view\_webseries', views.view\_webseries, name="view\_webseries"),  
 path('webseries\_edit/<int:id>', views.webseries\_edit, name="webseries\_edit"),  
 path('webseries\_update', views.webseries\_update, name="webseries\_update"),  
 path('view\_webseries\_delete/<int:id>', views.view\_webseries\_delete, name="view\_webseries\_delete"),  
 path('add\_serials', views.add\_serials, name="add\_serials"),  
 path('view\_serials', views.view\_serial, name="view\_serials"),  
 path('serial\_edit/<int:id>', views.serial\_edit, name="serial\_edit"),  
 path('serial\_update', views.serial\_update, name="serial\_update"),  
 path('view\_serial\_delete/<int:id>', views.view\_serial\_delete, name="view\_serial\_delete"),  
 path('webseries\_add\_episode/<int:id>', views.webseries\_add\_episode, name='webseries\_add\_episode'),  
 path('view\_webseries\_episodes/<int:id>', views.view\_webseries\_episodes, name="view\_webseries\_episodes"),  
 path('serial\_add\_episode/<int:id>', views.serial\_add\_episode, name='serial\_add\_episode'),  
 path('view\_serials\_episodes/<int:id>', views.view\_serials\_episodes, name="view\_serials\_episodes"),  
 path('customer\_view\_webseries\_episodes/<int:id>', views.customer\_view\_webseries\_episodes, name="customer\_view\_webseries\_episodes"),  
 path('customer\_view\_serials\_episodes/<int:id>', views.customer\_view\_serials\_episodes,name="customer\_view\_serials\_episodes"),  
 path('add\_movie\_review/<int:id>',views.add\_movie\_review,name="add\_movie\_review"),  
 path('view\_movie\_review', views.view\_movie\_review, name="view\_movie\_review"),  
 path('add\_webseries\_review/<int:id>',views.add\_webseries\_review,name="add\_webseries\_review"),  
 path('view\_webseries\_review', views.view\_webseries\_review, name="view\_webseries\_review"),  
 path('add\_serial\_review/<int:id>',views.add\_serial\_review,name="add\_serial\_review"),  
 path('view\_serial\_review', views.view\_serial\_review, name="view\_serial\_review"),  
 path('add\_plan', views.add\_plan, name="add\_plan"),  
 path('view\_plan', views.view\_plan, name="view\_plan"),  
 path('customer\_view\_plans', views.customer\_view\_plans, name="customer\_view\_plans"),  
 path('buy\_plan/<int:id>', views.buy\_plan, name="buy\_plan"),  
 path('admin\_view\_subscribers/', views.admin\_view\_subscribers, name='admin\_view\_subscribers'),  
 path('customer\_view\_history/', views.customer\_view\_history, name='customer\_view\_history'),  
  
]  
if settings.DEBUG: # new  
 urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

Models:

from django.db import models  
  
  
# Create your models here.  
class Contact(models.Model):  
 message = models.TextField()  
 name = models.CharField(max\_length=50)  
 email = models.EmailField()  
 subject = models.TextField(max\_length=50)  
  
  
class Customer(models.Model):  
 name = models.CharField(max\_length=50)  
 email = models.EmailField()  
 gender = models.CharField(max\_length=40)  
 age = models.IntegerField()  
 mobile = models.BigIntegerField()  
 address = models.TextField()  
 password = models.CharField(max\_length=70)  
 status = models.CharField(max\_length=40, default='pending')  
 startdate = models.DateField(null=True, blank=True)  
 enddate = models.DateField(null=True, blank=True)  
  
  
class Notification(models.Model):  
 email = models.EmailField()  
 title = models.CharField(max\_length=70)  
 description = models.TextField()  
 date\_time = models.DateTimeField(auto\_now=True)  
  
  
class Category(models.Model):  
 title = models.CharField(max\_length=80)  
  
  
class Movie(models.Model):  
 title = models.CharField(max\_length=60)  
 category = models.TextField()  
 cast = models.CharField(max\_length=40)  
 description = models.TextField()  
 director = models.CharField(max\_length=60)  
 duration = models.DurationField()  
 release\_date = models.DateField()  
 language = models.CharField(max\_length=60)  
 thumbnail = models.ImageField()  
 movie\_video = models.FileField()  
  
  
class Webseries(models.Model):  
 title = models.CharField(max\_length=255)  
 category = models.CharField(max\_length=255)  
 cast = models.CharField(max\_length=255)  
 description = models.TextField()  
 duration = models.DurationField()  
 director = models.CharField(max\_length=255)  
 release\_date = models.DateField()  
 language = models.CharField(max\_length=50)  
 thumbnail = models.ImageField(upload\_to='thumbnails/')  
 webseries\_video = models.FileField()  
  
 def \_\_str\_\_(self):  
 return self.title  
  
  
class Episode(models.Model):  
 webseries = models.ForeignKey(Webseries, on\_delete=models.CASCADE)  
 title = models.CharField(max\_length=255)  
 description = models.TextField()  
 episode\_number = models.PositiveIntegerField()  
 release\_date = models.DateField()  
 thumbnail = models.FileField()  
 video = models.FileField()  
  
  
class Serials(models.Model):  
 title = models.CharField(max\_length=50)  
 cast = models.CharField(max\_length=50)  
 duration = models.DurationField()  
 description = models.TextField()  
 director = models.CharField(max\_length=50)  
 language = models.CharField(max\_length=50)  
 thumbnail = models.ImageField()  
 episode = models.FileField()  
  
  
class Serials\_Episodes(models.Model):  
 serial = models.ForeignKey(Serials, on\_delete=models.CASCADE)  
 title = models.CharField(max\_length=200)  
 video = models.FileField(upload\_to='episodes/')  
 description = models.TextField()  
 episode\_number = models.IntegerField()  
  
  
class Movie\_Review(models.Model):  
 title = models.CharField(max\_length=400)  
 customer\_email = models.EmailField()  
 date = models.DateTimeField(auto\_now=True)  
 review = models.CharField(max\_length=150)  
 rating = models.CharField(max\_length=200)  
  
  
class Webseries\_Review(models.Model):  
 title = models.CharField(max\_length=400)  
 customer\_email = models.EmailField()  
 date = models.DateTimeField(auto\_now=True)  
 review = models.CharField(max\_length=150)  
 rating = models.CharField(max\_length=200)  
  
  
class Serial\_Review(models.Model):  
 title = models.CharField(max\_length=400)  
 customer\_email = models.EmailField()  
 date = models.DateTimeField(auto\_now=True)  
 review = models.CharField(max\_length=150)  
 rating = models.CharField(max\_length=200)  
  
  
class AdminLogin(models.Model):  
 email = models.CharField(max\_length=100)  
 password = models.CharField(max\_length=20)  
  
  
class Subscription(models.Model):  
 email = models.EmailField()  
 title = models.CharField(max\_length=100)  
 months = models.IntegerField()  
 cost = models.BigIntegerField()  
 discount = models.BigIntegerField()  
 date\_time = models.DateTimeField(auto\_now=True)  
  
  
class Subscribers(models.Model):  
 customer\_email = models.EmailField()  
 subscription = models.ForeignKey(Subscription, on\_delete=models.CASCADE)  
 startdate = models.DateField()  
 enddate = models.DateField()  
 holder\_name = models.CharField(max\_length=100)  
 card\_number = models.BigIntegerField()  
 cvv\_number = models.BigIntegerField()  
 bank\_name = models.CharField(max\_length=100)

Forms:

from django import forms  
from .models import Contact, Customer, Notification, Category, Movie, Webseries, Serials, Episode, Serials\_Episodes, \  
 Movie\_Review, Serial\_Review, Webseries\_Review, Subscription, Subscribers  
  
  
class ContactForm(forms.ModelForm):  
 class Meta:  
 model = Contact  
 fields = "\_\_all\_\_"  
  
  
class CustomerForm(forms.ModelForm):  
 class Meta:  
 model = Customer  
 fields = "\_\_all\_\_"  
 exclude = ['status','startdate','enddate']  
  
  
class NotificationForm(forms.ModelForm):  
 class Meta:  
 model = Notification  
 fields = "\_\_all\_\_"  
  
  
class CategoryForm(forms.ModelForm):  
 class Meta:  
 model = Category  
 fields = "\_\_all\_\_"  
  
  
class MovieForm(forms.ModelForm):  
 class Meta:  
 model = Movie  
 fields = "\_\_all\_\_"  
  
  
class WebseriesForm(forms.ModelForm):  
 class Meta:  
 model = Webseries  
 fields = "\_\_all\_\_"  
  
  
class SerialsForm(forms.ModelForm):  
 class Meta:  
 model = Serials  
 fields = "\_\_all\_\_"  
  
  
class EpisodeForm(forms.ModelForm):  
 class Meta:  
 model = Episode  
 fields = "\_\_all\_\_"  
  
  
class SerialEpisodeForm(forms.ModelForm):  
 class Meta:  
 model = Serials\_Episodes  
 fields = "\_\_all\_\_"  
  
  
class MovieReviewForm(forms.ModelForm):  
 class Meta:  
 model = Movie\_Review  
 fields = "\_\_all\_\_"  
  
  
class WebseriesReviewForm(forms.ModelForm):  
 class Meta:  
 model = Webseries\_Review  
 fields = "\_\_all\_\_"  
  
  
class SerialReviewForm(forms.ModelForm):  
 class Meta:  
 model = Serial\_Review  
 fields = "\_\_all\_\_"  
  
  
class SubscriptionForm(forms.ModelForm):  
 class Meta:  
 model = Subscription  
 fields = "\_\_all\_\_"  
  
class SubscribersForm(forms.ModelForm):  
 class Meta:  
 model = Subscribers  
 fields = ['holder\_name', 'card\_number', 'cvv\_number', 'bank\_name']

Templates:

Admin Add Movies:

{% load static %}  
{% include "admin\_header.html" %}  
  
<!--::breadcrumb part start::-->  
<section class="breadcrumb blog\_bg">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-12">  
 <div class="breadcrumb\_iner">  
 <div class="breadcrumb\_iner\_item">  
 <h2>Add Movie</h2>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</section>  
<!--::breadcrumb part start::-->  
  
<!-- ================ customer login section start ================= -->  
<section class="contact-section ">  
 <div class="container">  
 <h6 style="color:red;">{{ msg }}</h6>  
 <div class="d-none d-sm-block mb-5 pb-4"></div>  
 <div class="row">  
 <div class="col-12"></div>  
 <div class="col-lg-8">  
 <form method="post" enctype="multipart/form-data">  
 {% csrf\_token %}  
 <div class="row">  
 <div class="col-12">  
 <div class="form-group">  
 <label>Title:</label>  
 <input class="form-control placeholder hide-on-focus" name="title" id="title" type="text" placeholder="Movie Title" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="form-group">  
 <label>Cast:</label>  
 <input class="form-control placeholder hide-on-focus" name="cast" id="cast" type="text" placeholder="Enter Cast" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="form-group">  
 <label>Category:</label>  
 <input class="form-control placeholder hide-on-focus" name="category" id="category" type="text" placeholder="Movie Category" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="form-group">  
 <label>Description:</label>  
 <input class="form-control placeholder hide-on-focus" name="description" id="description" type="text" placeholder="Enter Description" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="form-group">  
 <label>Director:</label>  
 <input class="form-control placeholder hide-on-focus" name="director" id="director" type="text" placeholder="Director Name" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="form-group">  
 <label>Duration:</label>  
 <input class="form-control placeholder hide-on-focus" name="duration" id="duration" type="time" placeholder="Movie Duration"required>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="form-group">  
 <label>Release Date:</label>  
 <input class="form-control placeholder hide-on-focus" name="release\_date" id="release\_date" type="date" placeholder="Release Date" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="form-group">  
 <label>Language:</label>  
 <input class="form-control placeholder hide-on-focus" name="language" id="language" type="text" placeholder="Movie Language" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="form-group">  
 <label>Thumbnail:</label>  
 <input class="form-control placeholder hide-on-focus" name="thumbnail" id="thumbnail" type="file" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="form-group">  
 <label>Trailer:</label>  
 <input class="form-control placeholder hide-on-focus mb-3" name="movie\_video" id="movie\_video" type="file" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <button type="submit" class="button button-contactForm mb-4">Add Movie</button>  
 </div>  
 </div>  
 </form>  
 </div>  
 <div class="col-lg-4"></div>  
 </div>  
 </div>  
</section>  
<!-- ================ contact section end ================= -->  
  
{% include "footer.html" %}  
  
<!-- JavaScript to Disable Previous Dates -->  
<script>  
 document.addEventListener("DOMContentLoaded", function() {  
 // Get today's date in YYYY-MM-DD format  
 const today = new Date().toISOString().split('T')[0];  
 // Set the min attribute of the date input to today's date  
 document.getElementById('release\_date').setAttribute('min', today);  
 });  
</script>

Customer\_Profile:

{% load static %}  
{% include "customer\_header.html" %}  
  
<!--::breadcrumb part start::-->  
<section class="breadcrumb blog\_bg">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-12">  
 <div class="breadcrumb\_iner">  
 <div class="breadcrumb\_iner\_item">  
 <h2>Customer Profile</h2>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</section>  
<!--::breadcrumb part end::-->  
  
<!-- ================ Profile Section Start ================= -->  
<section class="contact-section">  
 <div class="container">  
 <div class="d-none d-sm-block mb-5 pb-4"></div>  
 <div class="row">  
 <div class="m-3">  
 <table class="table table-bordered">  
 <thead>  
 <tr>  
 <th>Name</th>  
 <th>Email</th>  
 <th>Gender</th>  
 <th>Age</th>  
 <th>Mobile</th>  
 <th>Address</th>  
 <th>Start Date</th>  
 <th>End Date</th>  
 <th>Action</th>  
 <th>Deactivate</th>  
 <th>History</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr>  
 <td>{{ x.name }}</td>  
 <td>{{ x.email }}</td>  
 <td>{{ x.gender }}</td>  
 <td>{{ x.age }}</td>  
 <td>{{ x.mobile }}</td>  
 <td>{{ x.address }}</td>  
 <!-- Display Start Date with status color -->  
 <td>  
 {% if active\_subscription %}  
 <span class="badge {% if active\_subscription.is\_active %}badge-success{% else %}badge-danger{% endif %}">  
 {{ active\_subscription.startdate }}  
 </span>  
 {% else %}  
 N/A  
 {% endif %}  
 </td>  
 <!-- Display End Date with status color -->  
 <td>  
 {% if active\_subscription %}  
 <span class="badge {% if active\_subscription.is\_active %}badge-success{% else %}badge-danger{% endif %}">  
 {{ active\_subscription.enddate }}  
 </span>  
 {% else %}  
 N/A  
 {% endif %}  
 </td>  
 <td>  
 <a href="{% url 'customer\_edit' %}"><b>Update</b></a>  
 </td>  
 <td>  
 <a href="{% url 'customer\_deactivate' id=x.id %}" onclick="return confirm('Really Want To Deactivate Your Account!?')"><b>Deactivate</b></a>  
 </td>  
 <td>  
 <a href="{% url 'customer\_view\_history' %}"><b>History</b></a>  
 </td>  
 </tr>  
 </tbody>  
 </table><br><br>  
  
 <!-- Display Expired Subscriptions History -->  
  
 </div>  
 </div>  
 </div>  
</section>  
  
<!-- ================ Profile Section End ================= -->  
  
{% include "footer.html" %}

Buy Plan:

{% load static %}  
{% include "customer\_header.html" %}  
  
<!--::breadcrumb part start::-->  
<section class="breadcrumb blog\_bg">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-12">  
 <div class="breadcrumb\_iner">  
 <div class="breadcrumb\_iner\_item">  
 <h2>Buy</h2>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</section>  
<!--::breadcrumb part start::-->  
  
<!-- ================ customer login section start ================= -->  
<section class="contact-section ">  
 <div class="container mt-3">  
 <h4 style="color:green;">{{ message }}</h4>  
 <div class="d-none d-sm-block mb-5 pb-4"></div>  
 <div class="row">  
 <div class="col-12">  
 <h2 class="contact-title">Buy</h2>  
 </div>  
 <div class="col-lg-8">  
 <form method="post">  
 {% csrf\_token %}  
 <div class="row">  
  
 <!-- Customer Email Field (Read-only) -->  
 <div class="col-12">  
 <div class="form-group">  
 <label for="customer\_email">Customer Email:</label>  
 <input class="form-control placeholder hide-on-focus" name="customer\_email" id="customer\_email" type="email" value="{{ email }}" readonly placeholder="Customer Email">  
 </div>  
 </div>  
  
 <!-- Subscription ID Field (Read-only) -->  
 <div class="col-12">  
 <div class="form-group">  
 <label for="subscription">Subscription ID:</label>  
 <input class="form-control placeholder hide-on-focus" name="subscription" id="subscription" type="number" value="{{ plans.id }}" readonly placeholder="Subscription ID">  
 </div>  
 </div>  
  
 <!-- Start Date Field (Read-only) -->  
 <div class="col-12">  
 <div class="form-group">  
 <label for="startdate">Start Date:</label>  
 <input class="form-control placeholder hide-on-focus" name="startdate" id="startdate" type="text" value="{{ startdate }}" readonly placeholder="Start Date" required>  
 </div>  
 </div>  
  
 <!-- End Date Field (Read-only) -->  
 <div class="col-12">  
 <div class="form-group">  
 <label for="enddate">End Date:</label>  
 <input class="form-control placeholder hide-on-focus" name="enddate" id="enddate" type="text" value="{{ enddate }}" readonly placeholder="End Date" required>  
 </div>  
 </div>  
  
 <!-- Holder Name Field -->  
 <div class="col-12">  
 <div class="form-group">  
 <label for="holder\_name">Card Holder Name:</label>  
 <input class="form-control placeholder hide-on-focus" name="holder\_name" id="holder\_name" type="text" placeholder="Card Holder Name" required>  
 </div>  
 </div>  
  
 <!-- Card Number Field -->  
 <div class="col-12">  
 <div class="form-group">  
 <label for="card\_number">Card Number:</label>  
 <input class="form-control placeholder hide-on-focus" name="card\_number" id="card\_number" type="number" placeholder="Card Number" required>  
 </div>  
 </div>  
  
 <!-- CVV Number Field -->  
 <div class="col-12">  
 <div class="form-group">  
 <label for="cvv\_number">CVV Number:</label>  
 <input class="form-control placeholder hide-on-focus" name="cvv\_number" id="cvv\_number" type="number" placeholder="CVV Number" required>  
 </div>  
 </div>  
  
 <!-- Bank Name Field -->  
 <div class="col-12">  
 <div class="form-group">  
 <label for="bank\_name">Bank Name:</label>  
 <input class="form-control placeholder hide-on-focus" name="bank\_name" id="bank\_name" type="text" placeholder="Bank Name" required>  
 </div>  
 </div>  
  
 <!-- Submit Button -->  
 <div class="col-12">  
 <div class="form-group mt-3">  
 <button type="submit" class="button button-contactForm">Buy</button>  
 </div>  
 </div>  
  
 </div>  
 </form>  
 </div>  
 <div class="col-lg-4"></div>  
 </div>  
 </div>  
</section>  
<!-- ================ contact section end ================= -->  
  
{% include "footer.html" %}

Customer View Customers:

{% load static %}  
{% include "admin\_header.html" %}  
  
  
 <!--::breadcrumb part start::-->  
 <section class="breadcrumb blog\_bg">  
 <div class="container">  
 <div class="row">  
 <div class="col-lg-12">  
 <div class="breadcrumb\_iner">  
 <div class="breadcrumb\_iner\_item">  
 <h2>View Customers</h2>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </section>  
 <!--::breadcrumb part start::-->  
  
 <!-- ================ profile section start ================= -->  
  
  
 <div class="m-3">  
 <table class="table table-bordered">  
 <thead>  
 <tr>  
 <th>Name</th>  
 <th>Email</th>  
 <th>Gender</th>  
 <th>Age</th>  
 <th>Mobile</th>  
 <th>Address</th>  
 <th>Status</th>  
 <th>Action</th>  
  
 </tr>  
 </thead>  
 {% for x in customer %}  
 <tbody>  
 <tr>  
 <td>{{x.name}}</td>  
 <td>{{x.email}}</td>  
 <td>{{x.gender}}</td>  
 <td>{{x.age}}</td>  
 <td>{{x.mobile}}</td>  
 <td>{{x.address}}</td>  
 <td>{{x.status}}</td>  
 <td>{% if x.status == 'pending' %}  
 <a href="{% url 'accept\_customer' id=x.id %}" class="btn btn-success" >Accept</a> <br><br>  
 <a class="btn btn-danger" href="{% url 'reject\_customer' id=x.id %}">Reject</a>  
 {% elif x.status == 'Accepted' %}  
 <button class="btn btn-warning" disabled="disabled" data-toggle="modal" data-target="#reply\_modal">Accepted</button>  
 {% else %}  
 <button class="btn btn-danger" disabled="disabled" data-toggle="modal" data-target="#reply\_modal">Rejected</button>  
 {% endif %}</td>  
  
 </tr>  
 </tbody>  
{% endfor %}  
</table><br></br>  
  
  
 </div>  
 </div>  
  
 </div>  
 </section>  
 <!-- ================ contact section end ================= -->  
  
{% include "footer.html" %}