import re  
from django.views.generic.base import View  
*# import wkhtmltopdf  
# from wkhtmltopdf.views import PDFTemplateResponse*import razorpay as razorpay  
import six  
from django.contrib.auth.decorators import login\_required  
from django.contrib.auth.forms import PasswordResetForm, UserCreationForm  
from django.contrib.auth.models import User  
from django.contrib.auth.tokens import PasswordResetTokenGenerator  
from django.contrib.sites.shortcuts import get\_current\_site  
from django.core.exceptions import ValidationError, ObjectDoesNotExist  
from django.core.mail import EmailMessage  
from django.forms import forms  
from django.http import HttpResponse, HttpResponseBadRequest, request  
from django.shortcuts import render, redirect  
from django.contrib.auth import login, authenticate, get\_user\_model  
from django.contrib import messages  
from django.template.loader import render\_to\_string  
from django.utils.encoding import force\_bytes, force\_str  
from django.utils.http import urlsafe\_base64\_encode, urlsafe\_base64\_decode  
from django.db.models.query\_utils import Q  
from django.utils.safestring import mark\_safe  
from django.views.decorators.csrf import csrf\_exempt  
from datetime import datetime  
from stream\_selector import settings  
from .forms import ProfileForm, SetPasswordForm  
from .models import UserBasicInfo, PaymentCheck, SectionFirst, SectionSecond, SectionThree, SectionFour, SectionFive  
from django.utils.datastructures import MultiValueDictKeyError  
  
*# from django\_xhtml2pdf.utils import pdf\_decorator*@csrf\_exempt  
@login\_required(login\_url='login/')  
def result(request):  
 if request.user.is\_authenticated:  
 user = request.user  
 userdetail = UserBasicInfo.objects.get(user\_id=user.id)  
 first = SectionFirst.objects.get(user\_id=user.id)  
 second = SectionSecond.objects.get(user\_id=user.id)  
 third = SectionThree.objects.get(user\_id=user.id)  
 four = SectionFour.objects.get(user\_id=user.id)  
 five = SectionFive.objects.get(user\_id=user.id)  
  
 *# condition for checking Non-Medical stream* if first.nineth\_marks == 'Between\_80%\_90%' or first.nineth\_marks == 'More\_than\_90%' and \  
 first.nineth\_marks\_math == 'Between\_80%\_90%' or first.nineth\_marks\_math == 'More\_than\_90%' and \  
 first.nineth\_marks\_science == 'Between\_80%\_90%' or first.nineth\_marks\_science == 'More\_than\_90%' and \  
 first.tenth\_marks == 'Between\_80%\_90%' or first.tenth\_marks == 'More\_than\_90%' or \  
 first.tenth\_marks == 'awaited' and first.tenth\_marks\_math == 'Between\_80%\_90%' or \  
 first.tenth\_marks\_math == 'More\_than\_90%' or first.tenth\_marks\_math == 'Result\_awaited' and \  
 first.tenth\_marks\_science == 'Between\_80%\_90%' or first.tenth\_marks\_science == 'More\_than\_90%' or \  
 first.tenth\_marks\_science == 'awaited' and first.most\_perfered\_sub == 'math' and second.study\_time\_spent == '4hr' and \  
 second.games\_time == '1hr' and second.screen\_time == 'Less\_than\_1 hr' and second.attempts == 'na' and \  
 second.edu\_gap == 'no' and five.father\_qual == 'doctorate' or five.father\_qual == 'post\_graduate' or five.father\_qual == 'graduate' or \  
 five.mother\_qual == 'doctorate' or five.mother\_qual == 'post\_graduate' or five.mother\_qual == 'graduate' and \  
 five.annual\_income == 'More\_than\_5' and third.math == 'yes' or third.physics == 'yes' or third.chemistry == 'yes' :  
 return render(request, 'result\_non\_medical.html', {'user\_detail': userdetail})  
  
 *# condition for checking Medical stream* elif first.nineth\_marks == 'Between\_80%\_90%' or first.nineth\_marks == 'More\_than\_90%' and \  
 first.nineth\_marks\_math == 'Between\_80%\_90%' or first.nineth\_marks\_math == 'More\_than\_90%' and \  
 first.nineth\_marks\_science == 'Between\_80%\_90%' or first.nineth\_marks\_science == 'More\_than\_90%' and \  
 first.tenth\_marks == 'Between\_80%\_90%' or first.tenth\_marks == 'More\_than\_90%' or \  
 first.tenth\_marks == 'awaited' and first.tenth\_marks\_math == 'Between\_80%\_90%' or \  
 first.tenth\_marks\_math == 'More\_than\_90%' or first.tenth\_marks\_math == 'awaited' and \  
 first.tenth\_marks\_science == 'Between\_80%\_90%' or first.tenth\_marks\_science == 'More\_than\_90%' or \  
 first.tenth\_marks\_science == 'awaited' and first.most\_perfered\_sub == 'math' and second.study\_time\_spent == '4hr' and \  
 second.games\_time == '1hr' and second.screen\_time == 'Less\_than\_1 hr' and second.attempts == 'na' and \  
 second.edu\_gap == 'no' and five.father\_qual == 'doctorate' or five.father\_qual == 'post\_graduate' or five.father\_qual == 'graduate' and \  
 five.mother\_qual == 'doctorate' or five.mother\_qual == 'post\_graduate' or five.mother\_qual == 'graduate' and \  
 five.annual\_income == 'More\_than\_5' and third.biology == 'yes' and third.math == 'no' or third.physics == 'yes' or third.chemistry == 'yes' :  
 return render(request, 'result\_medical.html', {'user\_detail': userdetail})  
  
 *# condition for checking Non-Medical and Medical stream* elif first.nineth\_marks == 'More\_than\_90%' and first.nineth\_marks\_math == 'More\_than\_90%' and \  
 first.nineth\_marks\_science == 'More\_than\_90%' and first.tenth\_marks == 'More\_than\_90%' or \  
 first.tenth\_marks == 'awaited' and first.tenth\_marks\_math == 'More\_than\_90%' or first.tenth\_marks\_math == 'awaited' \  
 and first.tenth\_marks\_science == 'More\_than\_90%' or first.tenth\_marks\_science == 'Result\_awaited' and \  
 first.most\_perfered\_sub == 'math' or first.most\_perfered\_sub == 'science' and second.study\_time\_spent == '4hr' and \  
 second.games\_time == '1hr' and second.screen\_time == '1hr' and second.attempts == 'na' and \  
 second.edu\_gap == 'no' and five.father\_qual == 'doctorate' or five.father\_qual == 'post\_graduate' or five.father\_qual == 'graduate' and \  
 five.mother\_qual == 'doctorate' or five.mother\_qual == 'post\_graduate' or five.mother\_qual == 'graduate' and \  
 five.annual\_income == 'More\_than\_5' and third.math == 'yes' and third.biology == 'yes' and \  
 third.math == 'no' or third.physics == 'yes' or third.chemistry == 'yes' :  
 return render(request, 'result\_both.html', {'user\_detail': userdetail})  
  
 *# condition for checking Arts stream* elif first.nineth\_marks == 'Between\_70%\_80%' or first.nineth\_marks == 'Less\_than\_60 %' and \  
 first.tenth\_marks == 'Between\_70%\_60%' or first.tenth\_marks == 'Less\_than\_60%' or first.tenth\_marks == 'awaited' \  
 and first.most\_perfered\_sub == 'social\_science' and second.study\_time\_spent == '3hr' and second.games\_time == '3hr' and \  
 second.screen\_time == '2hr' and second.attempts == 'na' and second.edu\_gap == 'no' or second.edu\_gap == 'yes' and \  
 third.history == 'yes' or third.geography == 'yes' and four.political\_science == 'yes':  
 return render(request, 'result\_arts.html', {'user\_detail': userdetail})  
  
 *# condition for checking Commerce stream* elif first.nineth\_marks == 'Between\_70%\_80%' or first.nineth\_marks == 'Less\_than\_60%' and \  
 first.nineth\_marks\_math == 'Between\_70%\_80%' or first.nineth\_marks\_math == 'Less\_than\_80%' and \  
 first.tenth\_marks == 'Between\_70%\_60%' or first.tenth\_marks == 'Less\_than\_60%' or first.tenth\_marks == 'awaited' \  
 and first.most\_perfered\_sub == 'math' and second.study\_time\_spent == '3hr' or second.study\_time\_spent == '2hr' and \  
 second.games\_time == '3hr' and second.screen\_time == '2hr' or second.screen\_time == '3hr' and third.commerce == 'yes' or \  
 third.accounts == 'yes' or third.statistics == 'yes':  
 return render(request, 'result\_commerce.html', {'user\_detail': userdetail})  
  
 elif first.nineth\_marks == 'Less\_than\_60%' and first.tenth\_marks == 'Less\_than\_60%' or first.tenth\_marks == 'awaited' and \  
 second.study\_time\_spent == '2hr' and second.games\_time == '3hr' and second.screen\_time == '2hr' and four.typewriting == 'yes' or \  
 four.stenography == 'yes' or four.beautician == 'yes' or four.library\_asst=='yes' or four.secretarial\_roles == 'yes':  
 return render(request, 'result\_vocational.html', {'user\_detail': userdetail})  
  
 elif first.nineth\_marks == 'Between\_70%\_60%' or first.nineth\_marks == 'Less\_than\_60%' and first.tenth\_marks == 'Between\_70%\_60%' or \  
 first.tenth\_marks == 'Less\_than\_60%' or first.tenth\_marks == 'awaited' and first.most\_perfered\_sub == 'math' or first.most\_perfered\_sub == 'science' and \  
 second.study\_time\_spent == '3hr' or second.study\_time\_spent == '2hr' and second.games\_time == '3hr' and second.screen\_time == '2hr' \  
 and third.math == 'yes' or four.computer == 'yes' or third.accounts == 'yes' or third.statistics == 'yes':  
 return render(request, 'result\_vocational.html', {'user\_detail': userdetail})  
  
 else:  
 return render(request, 'home.html', {'user\_detail': userdetail})  
  
 return render(request, 'home.html')  
  
  
@csrf\_exempt  
@login\_required(login\_url='login/')  
def home(request):  
 if request.user.is\_authenticated:  
 users = UserBasicInfo.objects.all().count()  
 candidate = SectionFive.objects.all().count()  
 payment = PaymentCheck.objects.all().count()  
 return render(request, 'home.html', {'users': users,  
 'candidate': candidate,  
 'payment': payment  
 })  
  
  
def index(request):  
 return render(request, "index.html")  
  
@csrf\_exempt  
def register(request):  
 if request.method == "POST":  
 *# Get the post parameters* username = request.POST['username']  
 email = request.POST['email']  
 fname = request.POST['fname']  
 lname = request.POST['lname']  
 pass1 = request.POST['pass1']  
 pass2 = request.POST['pass2']  
  
 *# check for errorneous input* if len(username) < 5:  
 messages.error(request, " Username should more than 5 characters")  
 return redirect('register')  
  
 if not username.isalnum():  
 messages.error(request, " User name should contain letters and numbers")  
 return redirect('register')  
  
 if not re.fullmatch(r'[A-Za-z0-9@#$%^&+=]{8,}', pass1):  
 messages.error(request,  
 "Password Must contain atleast one letter, one number,one special character. Minimum length should be 8 characters")  
 return redirect('register')  
  
 if (pass1 != pass2):  
 messages.error(request, " Passwords do not match")  
 return redirect('register')  
  
 if username == "" and pass1 == "" and email == "" and fname == "" and lname == "":  
 messages.error(request, "Kindly fill the fields")  
 return redirect("register")  
  
 *# Create the user* if not (User.objects.filter(username=username).exists() or User.objects.filter(email=email).exists()):  
 myuser = User.objects.create\_user(username, email, pass1)  
 myuser.first\_name = fname  
 myuser.last\_name = lname  
 myuser.save()  
 messages.success(request, " Your account has been successfully created")  
 return redirect('login')  
 else:  
 messages.error(request, "Looks like a username with that email or password already exist")  
 return redirect('register')  
 return render(request, 'register.html', context)  
 else:  
 return render(request, 'register.html')  
  
@csrf\_exempt  
def user\_login(request):  
 if request.method == 'POST':  
 username = request.POST['username']  
 password = request.POST['password']  
 if username == "" and password == "":  
 messages.error(request, "Kindly fill the fields")  
 return redirect("login")  
 user = authenticate(request, username=username, password=password)  
 if user is not None:  
 login(request, user)  
 messages.success(request, "Successfully Logged In")  
 return redirect("home")  
 else:  
 messages.error(request, "Invalid credentials! Please try again")  
 return redirect("login")  
 else:  
 return render(request, 'login.html')  
  
  
@csrf\_exempt  
@login\_required(login\_url='login/')  
def profile(request):  
 if request.method == 'POST':  
 name = request.POST['sname']  
 fathername = request.POST['fathername']  
 mothername = request.POST['mothername']  
 dob = request.POST['dob']  
 gender = request.POST['gender']  
 category = request.POST['category']  
 address = request.POST['address']  
 state = request.POST['state']  
 district = request.POST['district']  
 city = request.POST['city']  
 pin = request.POST['pin']  
 area = request.POST['area']  
 board = request.POST['board']  
 school = request.POST['school']  
 school\_type = request.POST['school\_type']  
 mobile = request.POST['number']  
 anumber = request.POST['anumber']  
 *# photo = request.FILES['image']* if name == "" and fathername == "" and mothername == "" and dob == "" and gender == "" and category == "" and \  
 school\_type == "" and state == "" and city == "" and district == "" and pin == "" and area == "" and board == "" and school == "" and mobile == "" and anumber == "":  
 messages.error(request, "Kindly fill the fields")  
 return redirect("profile")  
  
 if (mobile == anumber):  
 messages.error(request, " Both numbers should be different")  
 return redirect('profile')  
  
 user = request.user  
 if request.user.is\_authenticated:  
 profilemodel = UserBasicInfo()  
 profilemodel.full\_name = name  
 profilemodel.father\_name = fathername  
 profilemodel.mother\_name = mothername  
 profilemodel.dob = datetime.strptime(dob, "%Y-%m-%d")  
 profilemodel.gender = gender  
 profilemodel.category = category  
 profilemodel.address = address  
 profilemodel.state = state  
 profilemodel.district = district  
 profilemodel.city = city  
 profilemodel.pin = pin  
 profilemodel.area = area  
 profilemodel.board = board  
 profilemodel.school\_name = school  
 profilemodel.school\_type = school\_type  
 profilemodel.mobile\_num = mobile  
 profilemodel.parents\_num = anumber  
 profilemodel.user\_id = user.id  
 profilemodel.check\_alerts = "yes"  
 profilemodel.save()  
  
 *# profilemodel = UserBasicInfo.objects.filter(full\_name=name).first()  
 # userimage = UserImage()  
 # userimage.name = profilemodel  
 # userimage.user\_image = photo  
 # userimage.user\_image\_ext = photo.name.split('.')[-1]  
 # userimage.save()* messages.success(request, f'Your data has been added. Now you can take Stream Selection test')  
 return redirect('profile')  
 else:  
 messages.error(request, f'There is some error in your form. Kindly check and fill it again.')  
 return redirect('profile')  
 else:  
 return render(request, 'profile.html')  
  
@csrf\_exempt  
@login\_required(login\_url='login/')  
def section\_first(request):  
 user = request.user  
 try:  
 uid = UserBasicInfo.objects.get(user\_id=user.id)  
 if not SectionFirst.objects.filter(user\_id=user.id).exists():  
 if request.method == 'POST':  
 try:  
 nineth\_marks = request.POST['nineth\_marks']  
 math\_nineth\_marks = request.POST['math\_nineth\_marks']  
 sci\_nineth\_marks = request.POST['sci\_nineth\_marks']  
 tenth\_marks = request.POST['tenth\_marks']  
 math\_tenth\_marks = request.POST['math\_tenth\_marks']  
 sci\_tenth\_marks = request.POST['sci\_tenth\_marks']  
 math\_olympiad = request.POST['math\_olympiad']  
 sci\_olympiad = request.POST['sci\_olympiad']  
 sci\_workshop = request.POST['sci\_workshop']  
 most\_preferred\_sub = request.POST['most\_preferred\_sub']  
 least\_preferred\_sub = request.POST['least\_preferred\_sub']  
  
 if nineth\_marks == "" and math\_nineth\_marks == "" and sci\_nineth\_marks == "" and tenth\_marks == "" and math\_tenth\_marks == "" and sci\_tenth\_marks == "" and \  
 math\_olympiad == "" and sci\_olympiad == "" and sci\_workshop == "" and most\_preferred\_sub == "" and least\_preferred\_sub == "":  
 messages.error(request, "Kindly fill the fields")  
 return redirect("section\_first")  
 if request.user.is\_authenticated:  
 firstmodel = SectionFirst()  
 firstmodel.nineth\_marks = nineth\_marks  
 firstmodel.nineth\_marks\_math = math\_nineth\_marks  
 firstmodel.nineth\_marks\_science = sci\_nineth\_marks  
 firstmodel.tenth\_marks = tenth\_marks  
 firstmodel.tenth\_marks\_math = math\_tenth\_marks  
 firstmodel.tenth\_marks\_science = sci\_tenth\_marks  
 firstmodel.math\_olampaid = math\_olympiad  
 firstmodel.sci\_olampaid = sci\_olympiad  
 firstmodel.workshop = sci\_workshop  
 firstmodel.most\_perfered\_sub = most\_preferred\_sub  
 firstmodel.least\_perfered\_sub = least\_preferred\_sub  
 firstmodel.user\_id = user.id  
 firstmodel.save()  
  
 *# check errors  
 # success message redirect to result page* messages.success(request, f'Your data has been added.')  
 return redirect('section\_second')  
 else:  
 messages.error(request, f'Some error in the form.')  
 return redirect('section\_first')  
 except MultiValueDictKeyError:  
 messages.success(request, f'Already filled the previous section.')  
 return redirect('section\_second')  
 else:  
 messages.error(request, f'Already filled the Section 1. Kindly fill section 2')  
 return redirect('section\_second')  
 except UserBasicInfo.DoesNotExist:  
 messages.error(request, f'You forgot to fill Student Information form. Kindly fill it first.')  
 return redirect('profile')  
  
 return render(request, 'section\_first.html')  
  
@csrf\_exempt  
@login\_required(login\_url='login/')  
def section\_second(request):  
 user = request.user  
 try:  
 uid = UserBasicInfo.objects.get(user\_id=user.id)  
 if not SectionSecond.objects.filter(user\_id=user.id).exists():  
 if request.method == 'POST':  
 try:  
 study\_method = request.POST['study\_method']  
 study\_environment = request.POST['study\_environment']  
 study\_time\_spent = request.POST['time\_spent']  
 games\_time = request.POST['games\_time']  
 screen\_time = request.POST['screen\_time']  
 role\_model = request.POST['role\_model']  
 attempts = request.POST['attempts']  
 attendance = request.POST['attendance']  
 scholarship = request.POST['scholarship']  
 edu\_gap = request.POST['edu\_gap']  
  
 if study\_method == "" and study\_environment == "" and study\_time\_spent == "" and games\_time == "" and screen\_time == "" and \  
 role\_model == "" and attempts == "" and attendance == "" and edu\_gap == "" and scholarship == "":  
 messages.error(request, "Kindly fill the fields")  
 return redirect("section\_second")  
 if request.user.is\_authenticated:  
 secondmodel = SectionSecond()  
 secondmodel.study\_method = study\_method  
 secondmodel.study\_environment = study\_environment  
 secondmodel.study\_time\_spent = study\_time\_spent  
 secondmodel.games\_time = games\_time  
 secondmodel.screen\_time = screen\_time  
 secondmodel.role\_model = role\_model  
 secondmodel.attempts = attempts  
 secondmodel.attendance = attendance  
 secondmodel.scholarship = scholarship  
 secondmodel.edu\_gap = edu\_gap  
 secondmodel.user\_id = user.id  
 secondmodel.save()  
  
 *# check errors  
 # success message redirect to result page* messages.success(request, f'Your data has been added.')  
 return redirect('section\_three')  
 else:  
 messages.error(request, f'Some error in the form.')  
 return redirect('section\_second')  
 except MultiValueDictKeyError:  
 messages.success(request, f'Already filled the previous section.')  
 return redirect('section\_three')  
 else:  
 messages.error(request, f'Already filled the Section 2. Kindly fill section 3')  
 return redirect('section\_three')  
 except UserBasicInfo.DoesNotExist:  
 messages.error(request, f'You forgot to fill Student Information form. Kindly fill it first.')  
 return redirect('profile')  
 return render(request, 'section\_second.html')  
  
@csrf\_exempt  
@login\_required(login\_url='login/')  
def section\_three(request):  
 user = request.user  
 try:  
 uid = UserBasicInfo.objects.get(user\_id=user.id)  
 if not SectionThree.objects.filter(user\_id=user.id).exists():  
 if request.method == 'POST':  
 try:  
 math = request.POST['math']  
 physics = request.POST['physics']  
 chemistry = request.POST['chemistry']  
 biology = request.POST['biology']  
 history = request.POST['history']  
 geography = request.POST['geography']  
 commerce = request.POST['commerce']  
 accounts = request.POST['accounts']  
 statistics = request.POST['statistics']  
 language = request.POST['language']  
  
 if math == "" and history == "" and biology == "" and chemistry == "" and physics == "" and \  
 statistics == "" and accounts == "" and commerce == "" and geography == "" and language == "":  
 messages.error(request, "Kindly fill the fields")  
 return redirect("section\_three")  
 if request.user.is\_authenticated:  
 thirdmodel = SectionThree()  
 thirdmodel.math = math  
 thirdmodel.history = history  
 thirdmodel.biology = biology  
 thirdmodel.physics = physics  
 thirdmodel.chemistry = chemistry  
 thirdmodel.geography = geography  
 thirdmodel.commerce = commerce  
 thirdmodel.accounts = accounts  
 thirdmodel.statistics = statistics  
 thirdmodel.language = language  
 thirdmodel.user\_id = user.id  
 thirdmodel.save()  
  
 *# check errors  
 # success message redirect to result page* messages.success(request, f'Your data has been added.')  
 return redirect('section\_four')  
 else:  
 messages.error(request, f'Some error in the form.')  
 return redirect('section\_three')  
 except MultiValueDictKeyError:  
 messages.success(request, f'Already filled the previous section.')  
 return redirect('section\_four')  
 else:  
 messages.error(request, f'Already filled the Section 3. Kindly fill section 4')  
 return redirect('section\_four')  
 except UserBasicInfo.DoesNotExist:  
 messages.error(request, f'You forgot to fill Student Information form. Kindly fill it first.')  
 return redirect('profile')  
  
 return render(request, 'section\_three.html')  
  
@csrf\_exempt  
@login\_required(login\_url='login/')  
def section\_four(request):  
 user = request.user  
 try:  
 uid = UserBasicInfo.objects.get(user\_id=user.id)  
 if not SectionFour.objects.filter(user\_id=user.id).exists():  
 if request.method == 'POST':  
 try:  
 political\_science = request.POST['political\_science']  
 home\_science = request.POST['home\_science']  
 environment\_science = request.POST['environment\_science']  
 physical\_edu = request.POST['physical\_edu']  
 computers = request.POST['computer']  
 typewriting = request.POST['typewriting']  
 stenography = request.POST['stenography']  
 beautician = request.POST['beautician']  
 library\_asst = request.POST['library\_asst']  
 secretarial\_roles = request.POST['secretarial\_roles']  
 if political\_science == "" and home\_science == "" and environment\_science == "" and physical\_edu == "" and computers == "" and \  
 typewriting == "" and stenography == "" and beautician == "" and library\_asst == "" and secretarial\_roles == "":  
 messages.error(request, "Kindly fill the fields")  
 return redirect("section\_four")  
 if request.user.is\_authenticated:  
 fourthmodel = SectionFour()  
 fourthmodel.political\_science = political\_science  
 fourthmodel.home\_science = home\_science  
 fourthmodel.environment\_science = environment\_science  
 fourthmodel.physical\_edu = physical\_edu  
 fourthmodel.computers = computers  
 fourthmodel.typewriting = typewriting  
 fourthmodel.stenography = stenography  
 fourthmodel.beautician = beautician  
 fourthmodel.library\_asst = library\_asst  
 fourthmodel.secretarial\_roles = secretarial\_roles  
 fourthmodel.user\_id = user.id  
 fourthmodel.save()  
  
 *# check errors  
 # success message redirect to result page* messages.success(request, f'proceed with next section.')  
 return redirect('section\_five')  
  
 else:  
 messages.error(request, f'Some Errors')  
 return redirect('section\_four')  
 except MultiValueDictKeyError:  
 messages.success(request, f'Already filled the previous section.')  
 return redirect('section\_five')  
 else:  
 messages.error(request, f'Already filled the Section 4. Kindly fill section 5')  
 return redirect('section\_five')  
 except UserBasicInfo.DoesNotExist:  
 messages.error(request, f'You forgot to fill Student Information form. Kindly fill it first.')  
 return redirect('profile')  
 return render(request, 'section\_four.html')  
  
@csrf\_exempt  
@login\_required(login\_url='login/')  
def section\_five(request):  
 user = request.user  
 try:  
 uid = UserBasicInfo.objects.get(user\_id=user.id)  
 if not SectionFive.objects.filter(user\_id=user.id).exists():  
 if request.method == 'POST':  
 try:  
 curricular = request.POST['curricular']  
 performance\_level = request.POST['performance\_level']  
 father\_qual = request.POST['father\_qual']  
 mother\_qual = request.POST['mother\_qual']  
 sibling\_qual = request.POST['sibiling\_qual']  
 father\_job = request.POST['father\_job']  
 mother\_job = request.POST['mother\_job']  
 sibling\_job = request.POST['sibling\_job']  
 annual\_income = request.POST['annual\_income']  
  
 if curricular == "" and performance\_level == "" and father\_qual == "" and mother\_qual == "" and \  
 sibling\_qual == "" and father\_job == "" and mother\_job == "" and sibling\_job == "" and annual\_income == "":  
 messages.error(request, "Kindly fill the fields")  
 return redirect("section\_five")  
 if request.user.is\_authenticated:  
 fifthmodel = SectionFive()  
 fifthmodel.curricular = curricular  
 fifthmodel.performance\_level = performance\_level  
 fifthmodel.father\_qual = father\_qual  
 fifthmodel.mother\_qual = mother\_qual  
 fifthmodel.sibling\_qual = sibling\_qual  
 fifthmodel.father\_job = father\_job  
 fifthmodel.mother\_job = mother\_job  
 fifthmodel.sibling\_job = sibling\_job  
 fifthmodel.annual\_income = annual\_income  
 fifthmodel.user\_id = user.id  
 fifthmodel.save()  
  
 *# check errors  
 # success message redirect to result page* messages.success(request,  
 f'You had given your test successfully. Now you can proceed with payment for result')  
 return redirect('result')  
 else:  
 messages.error(request, f'Some error in the form.')  
 return redirect('section\_five')  
 except MultiValueDictKeyError:  
 messages.success(request, f'Already fill the section.')  
 return redirect('result')  
 else:  
 messages.error(request, f'Already given the test. Kindly check result')  
 return redirect('result')  
 except UserBasicInfo.DoesNotExist:  
 messages.error(request, f'You forgot to fill Student Information form. Kindly fill it first.')  
 return redirect('profile')  
  
 return render(request, 'section\_five.html')  
  
  
def section\_six(request):  
 return render(request, 'section\_six.html')  
  
  
def section\_seven(request):  
 return render(request, 'section\_seven.html')  
  
  
def section\_eight(request):  
 return render(request, 'section\_eight.html')  
  
  
def section\_nine(request):  
 return render(request, 'section\_nine.html')  
  
  
def section\_ten(request):  
 return render(request, 'section\_ten.html')  
  
@csrf\_exempt  
@login\_required  
def password\_change(request):  
 user = request.user  
 if request.method == 'POST':  
 form = SetPasswordForm(user, request.POST)  
 if form.is\_valid():  
 form.save()  
 messages.success(request, "Your password has been changed")  
 return redirect('login')  
 else:  
 for error in list(form.errors.values()):  
 messages.error(request, error)  
  
 form = SetPasswordForm(user)  
 return render(request, 'password\_reset\_confirm.html', {'form': form})  
  
  
class ActivationTokenGenerator(PasswordResetTokenGenerator):  
 def \_make\_hash\_value(self, user, timestamp):  
 return six.text\_type(user.pk) + six.text\_type(timestamp) + six.text\_type(user.username)  
  
  
account\_activation\_token = ActivationTokenGenerator()  
  
@csrf\_exempt  
def password\_reset(request):  
 if request.method == 'POST':  
 form = PasswordResetForm(request.POST)  
 if form.is\_valid():  
 user\_email = form.cleaned\_data['email']  
 associated\_user = get\_user\_model().objects.filter(Q(email=user\_email)).first()  
 if associated\_user:  
 subject = "Password Reset request"  
 message = render\_to\_string("template\_reset\_password.html", {  
 'user': associated\_user,  
 'domain': get\_current\_site(request).domain,  
 'uid': urlsafe\_base64\_encode(force\_bytes(associated\_user.pk)),  
 'token': account\_activation\_token.make\_token(associated\_user),  
 "protocol": 'https' if request.is\_secure() else 'http'  
 })  
 email = EmailMessage(subject, message, to=[associated\_user.email])  
 if email.send():  
 messages.success(request,  
 """  
 <h2>Password reset sent</h2><hr>  
 <p>  
 We've emailed you instructions for setting your password, if an account exists with the email you entered.   
 You should receive them shortly.<br>If you don't receive an email, please make sure you've entered the address   
 you registered with, and check your spam folder.  
 </p>  
 """  
 )  
 else:  
 messages.error(request, "Problem sending reset password email, <b>SERVER PROBLEM</b>")  
  
 return redirect('home')  
  
 form = PasswordResetForm()  
 return render(request, "password\_reset.html", context={"form": form})  
  
@csrf\_exempt  
def password\_reset\_confirm(request, uidb64, token):  
 User = get\_user\_model()  
 try:  
 uid = force\_str(urlsafe\_base64\_decode(uidb64))  
 user = User.objects.get(pk=uid)  
 except:  
 user = None  
  
 if user is not None and account\_activation\_token.check\_token(user, token):  
 if request.method == 'POST':  
 form = SetPasswordForm(user, request.POST)  
 if form.is\_valid():  
 form.save()  
 messages.success(request, "Your password has been set. You may go ahead and <b>log in </b> now.")  
 return redirect('homepage')  
 else:  
 for error in list(form.errors.values()):  
 messages.error(request, error)  
  
 form = SetPasswordForm(user)  
 return render(request, 'password\_reset\_confirm.html', {'form': form})  
 else:  
 messages.error(request, "Link is expired")  
  
 messages.error(request, 'Something went wrong, redirecting back to Homepage')  
 return redirect("home")  
  
@csrf\_exempt  
@login\_required  
def checkout(request):  
 user = request.user  
  
 currency = 'INR'  
 amount = 10000 *# Rs. 100  
  
 # Create a Razorpay Order* razorpay\_order = razorpay\_client.order.create(dict(amount=amount,  
 currency=currency,  
 payment\_capture='0'))  
  
 *# order id of newly created order.* razorpay\_order\_id = razorpay\_order['id']  
 callback\_url = 'payment\_handler/'  
  
 *# we need to pass these details to frontend.* context = {}  
 context['razorpay\_order\_id'] = razorpay\_order\_id  
 context['razorpay\_merchant\_key'] = settings.RAZOR\_KEY\_ID  
 context['razorpay\_amount'] = amount  
 context['currency'] = currency  
 context['callback\_url'] = callback\_url  
  
 return render(request, 'checkout.html', context=context)  
  
  
*# authorize razorpay client with API Keys.*razorpay\_client = razorpay.Client(  
 auth=(settings.RAZOR\_KEY\_ID, settings.RAZOR\_KEY\_SECRET))  
  
  
*# we need to csrf\_exempt this url as  
# POST request will be made by Razorpay  
# and it won't have the csrf token.*@csrf\_exempt  
def payment\_handler(request):  
 *# only accept POST request.* if request.method == "POST":  
 user = request.user  
 try:  
 *# get the required parameters from post request.* payment\_id = request.POST.get('razorpay\_payment\_id', '')  
 razorpay\_order\_id = request.POST.get('razorpay\_order\_id', '')  
 signature = request.POST.get('razorpay\_signature', '')  
 params\_dict = {  
 'razorpay\_order\_id': razorpay\_order\_id,  
 'razorpay\_payment\_id': payment\_id,  
 'razorpay\_signature': signature  
 }  
  
 *# verify the payment signature.* result = razorpay\_client.utility.verify\_payment\_signature(  
 params\_dict)  
 if result is not None:  
 amount = 10000 *# Rs. 100* try:  
  
 *# capture the payemt* razorpay\_client.payment.capture(payment\_id, amount)  
 check = PaymentCheck()  
 check.user = user.id  
 check.order\_id = payment\_id  
 payment\_status = "Success"  
 *# render success page on successful caputre of payment* return render(request, 'paymentsuccess.html')  
 except:  
  
 *# if there is an error while capturing payment.* return render(request, 'paymentfailure.html')  
 else:  
  
 *# if signature verification fails.* return render(request, 'paymentfailure.html')  
 except:  
  
 *# if we don't find the required parameters in POST data* return HttpResponseBadRequest()  
 else:  
 *# if other than POST request is made.* return HttpResponseBadRequest()  
  
@csrf\_exempt  
def error\_404\_view(request, exception):  
 *# we add the path to the the 404.html file  
 # here. The name of our HTML file is 404.html* return render(request, '404.html')