**SOURCE CODE**

Buyer user Side views.py

**from** django.shortcuts **import** render,HttpResponse, redirect  
**from** django.contrib **import** messages  
**from** .forms **import** BuyerUserRegistrationForm  
**from** .models **import** BuyerUserRegistrationModel, BuyerCropCartModels,BuyerTransactionModels,BlockChainTransactionModel  
**from** sellers.models **import** FarmersCropsModels  
**from** .utility.BlockChainImpl **import** Blockchain  
**from** django.db.models **import** Sum  
**import** random  
  
blockchain = Blockchain()  
*# Create your views here.***def** BuyerUserRegisterActions(request):  
 **if** request.method == **'POST'**:  
 form = BuyerUserRegistrationForm(request.POST)  
 **if** form.is\_valid():  
 print(**'Data is Valid'**)  
 form.save()  
 messages.success(request, **'You have been successfully registered'**)  
 form = BuyerUserRegistrationForm()  
 **return** render(request, **'BuyerUserRegistrations.html'**, {**'form'**: form})  
 **else**:  
 messages.success(request, **'Email or Mobile Already Existed'**)  
 print(**"Invalid form"**)  
 **else**:  
 form = BuyerUserRegistrationForm()  
 **return** render(request, **'BuyerUserRegistrations.html'**, {**'form'**: form})  
**def** BuyerUserLoginCheck(request):  
 **if** request.method == **"POST"**:  
 loginid = request.POST.get(**'loginname'**)  
 pswd = request.POST.get(**'pswd'**)  
 print(**"Login ID = "**, loginid, **' Password = '**, pswd)  
 **try**:  
 check = BuyerUserRegistrationModel.objects.get(loginid=loginid, password=pswd)  
 status = check.status  
 print(**'Status is = '**, status)  
 **if** status == **"activated"**:  
 request.session[**'id'**] = check.id  
 request.session[**'loggeduser'**] = check.name  
 request.session[**'loginid'**] = loginid  
 request.session[**'email'**] = check.email  
 print(**"User id At"**, check.id, status)  
 cartin = checkCartCount(loginid)  
 **return** render(request, **'buyers/BuyerUserHome.html'**, {**'count'**:cartin})  
 **else**:  
 messages.success(request, **'Your Account Not at activated'**)  
 **return** render(request, **'BuyerLogin.html'**)  
 **except** Exception **as** e:  
 print(**'Exception is '**, str(e))  
 **pass** messages.success(request, **'Invalid Login id and password'**)  
 **return** render(request, **'BuyerLogin.html'**, {})  
**def** BuyerUserHome(request):  
 loginid = request.session[**'loginid'**]  
 cartin = checkCartCount(loginid)  
 **return** render(request, **'buyers/BuyerUserHome.html'**, {**'count'**:cartin})  
  
**def** BuyerSearchProductsForm(request):  
 loginid = request.session[**'loginid'**]  
 cartin = checkCartCount(loginid)  
 **return** render(request,**"buyers/BuyerSearchProducts.html"**,{**'count'**:cartin})  
  
**def** BuyerSearchCropsAction(request):  
 **if** request.method==**'POST'**:  
 crpname = request.POST.get(**'cropname'**)  
 search\_data = FarmersCropsModels.objects.filter(cropname\_\_icontains=crpname)  
 loginid = request.session[**'loginid'**]  
 cartin = checkCartCount(loginid)  
 **return** render(request, **'buyers/BuyerSearchResults.html'**,{**'data'**:search\_data,**'count'**:cartin})  
  
  
  
**def** BuyerAddCropsToCart(request):  
 crop\_id = request.GET.get(**'cropid'**)  
 crop = FarmersCropsModels.objects.get(id=crop\_id)  
 sellername = crop.sellername  
 cropname = crop.cropname  
 price = crop.price  
 description = crop.description  
 file = crop.file  
 buyerUser = request.session[**'loginid'**]  
 buyeremail = request.session[**'email'**]  
 cartStatus = **'waiting'** BuyerCropCartModels.objects.create(buyerusername=buyerUser,buyeruseremail=buyeremail,sellername=sellername,cropname=cropname, description=description, price=price, file=file,status=cartStatus)  
 print(**"Seller name "**,sellername)  
 search\_data = FarmersCropsModels.objects.filter(cropname\_\_icontains=cropname)  
 cartin = checkCartCount(buyerUser)  
 print(**"Cart Count = "**,cartin)  
 loginid = request.session[**'loginid'**]  
 cartin = checkCartCount(loginid)  
 **return** render(request, **'buyers/BuyerSearchResults.html'**, {**'data'**: search\_data,**'count'**:cartin})  
  
  
**def** checkCartCount(buyername):  
 cartin = BuyerCropCartModels.objects.filter(buyerusername=buyername,status=**'waiting'**).count()  
 **return** cartin  
  
  
**def** BuyyerCheckCartData(request):  
 buyerName =request.GET.get(**'buyerUser'**)  
 data = BuyerCropCartModels.objects.filter(buyerusername=buyerName, status=**'waiting'**)  
 **return** render(request,**"buyers/BuyerCheckInCart.html"**,{**'data'**:data})  
  
**def** BuyerDeleteanItemfromCart(request):  
 cropid = request.GET.get(**'cropid'**)  
 BuyerCropCartModels.objects.filter(id=cropid).delete()  
 buyerName = request.session[**'loginid'**]  
 cartin = checkCartCount(buyerName)  
 data = BuyerCropCartModels.objects.filter(buyerusername=buyerName, status=**'waiting'**)  
 **return** render(request, **"buyers/BuyerCheckInCart.html"**, {**'data'**: data,**'count'**:cartin})  
  
**def** startBlockChainProcess(request):  
 blockchain = Blockchain()  
 t1 = blockchain.new\_transaction(**"Satoshi"**, **"Mike"**, **'5 BTC'**)  
 blockchain.new\_block(12346)  
 t2 = blockchain.new\_transaction(**"Mike"**, **"Satoshi"**, **'1 BTC'**)  
 t3 = blockchain.new\_transaction(**"Satoshi"**, **"Hal Finney"**, **'5 BTC'**)  
 blockchain.new\_block(12345)  
 print(**"Genesis block: "**, blockchain.chain)  
 **return** HttpResponse(**"Block Chain Started"**)  
  
**def** BuyerTotalAmountCheckOut(request):  
 buyerName = request.GET.get(**'buyername'**)  
 cartstatuc = **'waiting'** total\_price = BuyerCropCartModels.objects.filter(buyerusername=buyerName, status=**'waiting'**).aggregate(Sum(**'price'**))  
 total\_price = total\_price[**'price\_\_sum'**]  
 print(**'Total Price '**,total\_price)  
 bank = (**'SBI Bank'**,**'Union Bank'**,**'ICICI Bank'**,**'Axis Bank'**,**'Canara Bank'**,**'HDFC Bank'**,**'FDI Bank'**,**'Chase Bank'**)  
 recipient = random.choice(bank)  
 **return** render(request, **'buyers/BuyerInitiateTransactionForm.html'**,{**'buyername'**:buyerName,**'totaPrice'**:total\_price,**'bank'**:recipient})  
  
**def** StartBlockChainTransaction(request):  
 **if** request.method==**'POST'**:  
 *## Block Chain Data* buyername = request.POST.get(**'buyername'**)  
 totalamount = request.POST.get(**'totalamount'**)  
 recipientnmae = request.POST.get(**'recipientnmae'**)  
  
 *#Transaction Data* cardnumber = request.POST.get(**'cardnumber'**)  
 nameoncard = request.POST.get(**'nameoncard'**)  
 cvv = request.POST.get(**'cvv'**)  
 cardexpiry = request.POST.get(**'cardexpiry'**)  
  
 t1 = blockchain.new\_transaction(buyername, recipientnmae, totalamount)  
 proofId = **''**.join([str(random.randint(0, 999)).zfill(3) **for** \_ **in** range(2)])  
 blockchain.new\_block(int(proofId))  
 print(**"Genesis block: "**, blockchain.chain)  
 print(**"T1 is "**,t1)  
 currentTrnx = blockchain.chain[-1]  
 previousTranx = blockchain.chain[-2]  
 *### Current Tranasction Details* c\_transactions = currentTrnx.get(**'transactions'**)  
 c\_tnx\_Dict = c\_transactions[0]  
  
 c\_index = currentTrnx.get(**'index'**)  
 c\_timestamp = currentTrnx.get(**'timestamp'**)  
 c\_sender = c\_tnx\_Dict.get(**'sender'**)  
 c\_recipient = c\_tnx\_Dict.get(**'recipient'**)  
 c\_amount = c\_tnx\_Dict.get(**'amount'**)  
 c\_proof = currentTrnx.get(**'proof'**)  
 c\_previous\_hash = currentTrnx.get(**'previous\_hash'**)  
  
 c\_dict\_rslt = {**'c\_index'**:c\_index,**'c\_timestamp'**:c\_timestamp,**'c\_sender'**:c\_sender,**'c\_recipient'**:c\_recipient,**'c\_amount'**:c\_amount,**'c\_proof'**:c\_proof,**'c\_previous\_hash'**:c\_previous\_hash}  
  
 *# previous Transaction* p\_dict\_rslt = {}  
 p\_transactions = previousTranx.get(**'transactions'**)  
 **if**(len(p\_transactions)!=0):  
 p\_tnx\_Dict = p\_transactions[0]  
  
 p\_index = previousTranx.get(**'index'**)  
 p\_timestamp = previousTranx.get(**'timestamp'**)  
 p\_sender = p\_tnx\_Dict.get(**'sender'**)  
 p\_recipient = p\_tnx\_Dict.get(**'recipient'**)  
 p\_amount = p\_tnx\_Dict.get(**'amount'**)  
 p\_proof = previousTranx.get(**'proof'**)  
 p\_previous\_hash = previousTranx.get(**'previous\_hash'**)  
  
 BuyerTransactionModels.objects.create(buyername=buyername, totalamount=totalamount,recipientname=recipientnmae,cradnumber=cardnumber,nameoncard=nameoncard,cvv=cvv, cardexpiry=cardexpiry)  
 p\_dict\_rslt = {**'p\_index'**: p\_index, **'p\_timestamp'**: p\_timestamp, **'p\_sender'**: p\_sender, **'p\_recipient'**: p\_recipient, **'p\_amount'**: p\_amount, **'p\_proof'**: p\_proof, **'p\_previous\_hash'**: p\_previous\_hash}  
 BlockChainTransactionModel.objects.create(c\_index=c\_index,c\_timestamp=c\_timestamp,c\_sender=c\_sender,c\_recipient=c\_recipient, c\_amount=c\_amount,c\_proof=c\_proof,c\_previous\_hash=c\_previous\_hash,p\_index=p\_index, p\_timestamp=p\_timestamp,p\_sender=p\_sender,p\_recipient=p\_recipient,p\_amount=p\_amount,p\_proof=p\_proof,p\_previous\_hash=p\_previous\_hash)  
 buyer\_name = request.session[**'loginid'**]  
 print(**'buyername ='**,buyer\_name)  
 qs = BuyerCropCartModels.objects.filter(buyerusername=buyer\_name).update(status=**'purchased'**)  
 **else**:  
 BuyerTransactionModels.objects.create(buyername=buyername, totalamount=totalamount,  
 recipientname=recipientnmae, cradnumber=cardnumber,  
 nameoncard=nameoncard, cvv=cvv, cardexpiry=cardexpiry)  
  
 BlockChainTransactionModel.objects.create(c\_index=c\_index, c\_timestamp=c\_timestamp, c\_sender=c\_sender,  
 c\_recipient=c\_recipient, c\_amount=c\_amount, c\_proof=c\_proof,  
 c\_previous\_hash=c\_previous\_hash, p\_index=**'p\_index'**,  
 p\_timestamp=**'p\_timestamp'**, p\_sender=**'p\_sender'**,  
 p\_recipient=**"p\_recipient"**, p\_amount=**"p\_amount"**, p\_proof=**"p\_proof"**,  
 p\_previous\_hash=**"p\_previous\_hash"**)  
 buyer\_name = request.session[**'loginid'**]  
 print(**'buyername ='**, buyer\_name)  
 qs = BuyerCropCartModels.objects.filter(buyerusername=buyer\_name).update(status=**'purchased'**)  
 **return** render(request, **'buyers/TransactionResults.html'**,{**'c\_dict\_rslt'**:c\_dict\_rslt,**'p\_dict\_rslt'**:p\_dict\_rslt})  
  
**def** BuyerViewPurchasedDetails(request):  
 buyer\_name = request.session[**'loginid'**]  
 cartin = checkCartCount(buyer\_name)  
 data = BuyerCropCartModels.objects.filter(buyerusername=buyer\_name,status=**'purchased'**)  
 **return** render(request, **'buyers/BuyersViewPurchasedData.html'**,{**'data'**:data,**'count'**:cartin})  
  
**def** BuyerViewTransactinDetails(request):  
 bd\_name = request.session[**'loginid'**]  
 print(**'buyer\_name'**,bd\_name)  
 data = BuyerTransactionModels.objects.filter(buyername = **' '**+bd\_name)  
 cartin = checkCartCount(bd\_name)  
 **return** render(request, **'buyers/BuyersViewTransactionDetails.html'**,{**'data'**:data,**'count'**:cartin})

BlockChain.py

**import** hashlib  
**import** json  
**from** time **import** time  
  
  
**class** Blockchain(object):  
 **def** \_\_init\_\_(self):  
 self.chain = []  
 self.pending\_transactions = []  
  
 self.new\_block(previous\_hash=**"The Times 03/Oct/2020 A Study of Blockchain Technology in Farmer’s Portal."**,proof=100)  
  
 *# Create a new block listing key/value pairs of block information in a JSON object. Reset the list of pending transactions & append the newest block to the chain.* **def** new\_block(self, proof, previous\_hash=**None**):  
 block = {  
 **'index'**: len(self.chain) + 1,  
 **'timestamp'**: time(),  
 **'transactions'**: self.pending\_transactions,  
 **'proof'**: proof,  
 **'previous\_hash'**: previous\_hash **or** self.hash(self.chain[-1]),  
 }  
 self.pending\_transactions = []  
 self.chain.append(block)  
  
 **return** block  
  
 *# Search the blockchain for the most recent block.* @property  
 **def** last\_block(self):  
 **return** self.chain[-1]  
  
 *# Add a transaction with relevant info to the 'blockpool' - list of pending tx's.* **def** new\_transaction(self, sender, recipient, amount):  
 transaction = {  
 **'sender'**: sender,  
 **'recipient'**: recipient,  
 **'amount'**: amount  
 }  
 self.pending\_transactions.append(transaction)  
 **return** self.last\_block[**'index'**] + 1  
  
 *# receive one block. Turn it into a string, turn that into Unicode (for hashing). Hash with SHA256 encryption, then translate the Unicode into a hexidecimal string.* **def** hash(self, block):  
 string\_object = json.dumps(block, sort\_keys=**True**)  
 block\_string = string\_object.encode()  
  
 raw\_hash = hashlib.sha256(block\_string)  
 hex\_hash = raw\_hash.hexdigest()  
  
 **return** hex\_hash

Buyer Models:

**from** django.db **import** models  
  
*# Create your models here.***class** BuyerUserRegistrationModel(models.Model):  
 name = models.CharField(max\_length=100)  
 loginid = models.CharField(unique=**True**, max\_length=100)  
 password = models.CharField(max\_length=100)  
 mobile = models.CharField(unique=**True**, max\_length=100)  
 email = models.CharField(unique=**True**, max\_length=100)  
 locality = models.CharField(max\_length=100)  
 address = models.CharField(max\_length=1000)  
 city = models.CharField(max\_length=100)  
 state = models.CharField(max\_length=100)  
 status = models.CharField(max\_length=100)  
  
 **def** \_\_str\_\_(self):  
 **return** self.loginid  
  
 **class** Meta:  
 db\_table = **'BuyersRegistrations'  
  
  
  
class** BuyerCropCartModels(models.Model):  
 buyerusername = models.CharField(max\_length=100)  
 buyeruseremail = models.CharField(max\_length=100)  
 sellername = models.CharField(max\_length=100)  
 cropname = models.CharField(max\_length=100)  
 description = models.CharField(max\_length=100000)  
 price = models.FloatField()  
 file = models.FileField(upload\_to=**'files/'**)  
 cdate = models.DateTimeField(auto\_now\_add=**True**)  
 status = models.CharField(max\_length=50)  
  
 **def** \_\_str\_\_(self):  
 **return** self.buyerusername  
  
 **class** Meta:  
 db\_table = **"BuyerCartTable"  
  
  
class** BuyerTransactionModels(models.Model):  
 buyername = models.CharField(max\_length=100)  
 totalamount = models.FloatField()  
 recipientname = models.CharField(max\_length=100)  
 cradnumber = models.IntegerField()  
 nameoncard = models.CharField(max\_length=100)  
 cvv = models.IntegerField()  
 cardexpiry = models.CharField(max\_length=200)  
 trnx\_date = models.DateTimeField(auto\_now\_add=**True**)  
  
 **def** \_\_str\_\_(self):  
 *#return self.id* **return** self.buyername  
  
 **class** Meta:  
 db\_table = **"BuyerTransactionTable"  
  
class** BlockChainTransactionModel(models.Model):  
 c\_index = models.CharField(max\_length=100)  
 c\_timestamp = models.CharField(max\_length=100)  
 c\_sender = models.CharField(max\_length=100)  
 c\_recipient = models.CharField(max\_length=100)  
 c\_amount = models.CharField(max\_length=100)  
 c\_proof = models.CharField(max\_length=100)  
 c\_previous\_hash = models.CharField(max\_length=100)  
 p\_index = models.CharField(max\_length=100)  
 p\_timestamp = models.CharField(max\_length=100)  
 p\_sender = models.CharField(max\_length=100)  
 p\_recipient = models.CharField(max\_length=100)  
 p\_amount = models.CharField(max\_length=100)  
 p\_proof = models.CharField(max\_length=100)  
 p\_previous\_hash = models.CharField(max\_length=100)  
  
 **def** \_\_str\_\_(self):  
 **return** self.id  
  
 **class** Meta:  
 db\_table = **"BlockChainTransactiontable"**

Sellers side views.py

**from** django.shortcuts **import** render,HttpResponse  
**from** django.contrib **import** messages  
**from** .forms **import** SellerUserRegistrationForm  
**from** .models **import** SellerUserRegistrationModel, FarmersCropsModels  
**from** django.core.files.storage **import** FileSystemStorage  
**from** buyers.models **import** BuyerCropCartModels  
  
*# Create your views here.***def** SellerUserRegisterActions(request):  
 **if** request.method == **'POST'**:  
 form = SellerUserRegistrationForm(request.POST)  
 **if** form.is\_valid():  
 print(**'Data is Valid'**)  
 form.save()  
 messages.success(request, **'You have been successfully registered'**)  
 form = SellerUserRegistrationForm()  
 **return** render(request, **'SellerUserRegistrations.html'**, {**'form'**: form})  
 **else**:  
 messages.success(request, **'Email or Mobile Already Existed'**)  
 print(**"Invalid form"**)  
 **else**:  
 form = SellerUserRegistrationForm()  
 **return** render(request, **'SellerUserRegistrations.html'**, {**'form'**: form})  
**def** SellerUserLoginCheck(request):  
 **if** request.method == **"POST"**:  
 loginid = request.POST.get(**'loginname'**)  
 pswd = request.POST.get(**'pswd'**)  
 print(**"Login ID = "**, loginid, **' Password = '**, pswd)  
 **try**:  
 check = SellerUserRegistrationModel.objects.get(loginid=loginid, password=pswd)  
 status = check.status  
 print(**'Status is = '**, status)  
 **if** status == **"activated"**:  
 request.session[**'id'**] = check.id  
 request.session[**'loggeduser'**] = check.name  
 request.session[**'loginid'**] = loginid  
 request.session[**'email'**] = check.email  
 print(**"User id At"**, check.id, status)  
 **return** render(request, **'sellers/SellerUserHome.html'**, {})  
 **else**:  
 messages.success(request, **'Your Account Not at activated'**)  
 **return** render(request, **'SellerLogin.html'**)  
 **except** Exception **as** e:  
 print(**'Exception is '**, str(e))  
 **pass** messages.success(request, **'Invalid Login id and password'**)  
 **return** render(request, **'SellerLogin.html'**, {})  
**def** SellerUserHome(request):  
 **return** render(request, **'sellers/SellerUserHome.html'**, {})  
  
**def** SellerAddItemsForm(request):  
 **return** render(request, **'sellers/SellerAddItems.html'**,{})  
  
**def** SellerAddItemsAction(request):  
 **if** request.method==**'POST'**:  
 cropname = request.POST.get(**'cropname'**)  
 price = request.POST.get(**'price'**)  
 description = request.POST.get(**'description'**)  
 image\_file = request.FILES[**'file'**]  
  
 *# let's check if it is a csv file* **if not** image\_file.name.endswith(**'.jpg'**):  
 messages.error(request, **'THIS IS NOT A JPG FILE'**)  
  
 fs = FileSystemStorage()  
 filename = fs.save(image\_file.name, image\_file)  
 detect\_filename = fs.save(image\_file.name, image\_file)  
 uploaded\_file\_url = fs.url(filename)  
 loginid = request.session[**'loginid'**]  
 email = request.session[**'email'**]  
 FarmersCropsModels.objects.create(sellername=loginid, selleremail=email, cropname=cropname,price=price, description=description,file=uploaded\_file\_url)  
 messages.success(request, **'Crop Data Addedd Success'**)  
 **return** render(request, **'sellers/SellerAddItems.html'**, {})  
  
**def** SellersCommodities(request):  
 loginid = request.session[**'loginid'**]  
 data = FarmersCropsModels.objects.filter(sellername=loginid)  
 **return** render(request, **'sellers/SellersCommoditiesData.html'**,{**'data'**:data})  
  
**def** SellerUpdateProducts(request):  
 cropid = request.GET.get(**'cropid'**)  
 data = FarmersCropsModels.objects.get(id=cropid)  
 **return** render(request, **'sellers/CropsUpdatesbySeller.html'**, {**'data'**: data})  
  
 **return** HttpResponse(**"Update products Working Success"**)  
  
**def** SellerDeleteProducts(request):  
 cropid = request.GET.get(**'cropid'**)  
 FarmersCropsModels.objects.filter(id=cropid).delete()  
 loginid = request.session[**'loginid'**]  
 data = FarmersCropsModels.objects.filter(sellername=loginid)  
 **return** render(request, **'sellers/SellersCommoditiesData.html'**, {**'data'**: data})  
  
  
**def** SellerCropUpdateAction(request):  
 *#MyModel.objects.filter(pk=some\_value).update(field1='some value')* cropname = request.POST.get(**'cropname'**)  
 price = request.POST.get(**'price'**)  
 cropid = request.POST.get(**'cropid'**)  
 description = request.POST.get(**'description'**)  
 image\_file = request.FILES[**'file'**]  
 *# let's check if it is a csv file* **if not** image\_file.name.endswith(**'.jpg'**):  
 messages.error(request, **'THIS IS NOT A JPG FILE'**)  
  
 fs = FileSystemStorage()  
 filename = fs.save(image\_file.name, image\_file)  
 detect\_filename = fs.save(image\_file.name, image\_file)  
 uploaded\_file\_url = fs.url(filename)  
 FarmersCropsModels.objects.filter(id=cropid).update(cropname=cropname, price=price, description=description, file=uploaded\_file\_url)  
 loginid = request.session[**'loginid'**]  
 data = FarmersCropsModels.objects.filter(sellername=loginid)  
 **return** render(request, **'sellers/SellersCommoditiesData.html'**, {**'data'**: data})  
  
**def** SellerViewCarts(request):  
 sellername = request.session[**'loginid'**]  
 data = BuyerCropCartModels.objects.filter(sellername=sellername)  
 **return** render(request,**'sellers/SellersViewCart.html'**,{**'data'**:data})

Sellers Side models.py

**from** django.db **import** models  
  
*# Create your models here.***class** SellerUserRegistrationModel(models.Model):  
 name = models.CharField(max\_length=100)  
 loginid = models.CharField(unique=**True**, max\_length=100)  
 password = models.CharField(max\_length=100)  
 mobile = models.CharField(unique=**True**, max\_length=100)  
 email = models.CharField(unique=**True**, max\_length=100)  
 locality = models.CharField(max\_length=100)  
 address = models.CharField(max\_length=1000)  
 city = models.CharField(max\_length=100)  
 state = models.CharField(max\_length=100)  
 status = models.CharField(max\_length=100)  
  
 **def** \_\_str\_\_(self):  
 **return** self.loginid  
  
 **class** Meta:  
 db\_table = **'SellerRegistrations'  
  
class** FarmersCropDataModels(models.Model):  
 sellername = models.CharField(max\_length=100)  
 selleremail = models.CharField(max\_length=100)  
 cropname = models.CharField(max\_length=100)  
 price = models.FloatField()  
 description = models.CharField(max\_length=100000)  
 file = models.FileField(upload\_to=**'files/'**)  
 cdate = models.DateTimeField(auto\_now\_add=**True**)  
  
 **def** \_\_str\_\_(self):  
 **return** self.loginid  
  
 **class** Meta:  
 db\_table = **"Farmerscroptable"  
  
class** FarmersCropsModels(models.Model):  
 sellername = models.CharField(max\_length=100)  
 selleremail = models.CharField(max\_length=100)  
 cropname = models.CharField(max\_length=100)  
 price = models.FloatField()  
 description = models.CharField(max\_length=100000)  
 file = models.FileField(upload\_to=**'files/'**)  
 cdate = models.DateTimeField(auto\_now\_add=**True**)  
  
 **def** \_\_str\_\_(self):  
 **return** self.loginid  
  
 **class** Meta:  
 db\_table = **"FarmersCrops"**

Base.html

<!DOCTYPE **html**>  
{%load static%}  
<**html lang="en"**>  
 <**head**>  
 <**title**>Study of Blockchain</**title**>  
 <**meta charset="utf-8"**>  
 <**meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"**>  
  
 <**link href="https://fonts.googleapis.com/css?family=Poppins:200,300,400,500,600,700,800&display=swap" rel="stylesheet"**>  
 <**link href="https://fonts.googleapis.com/css?family=Lora:400,400i,700,700i&display=swap" rel="stylesheet"**>  
 <**link href="https://fonts.googleapis.com/css?family=Amatic+SC:400,700&display=swap" rel="stylesheet"**>  
  
 <**link rel="stylesheet" href="{%static 'css/open-iconic-bootstrap.min.css'%}"**>  
 <**link rel="stylesheet" href="{%static 'css/animate.css'%}"**>  
  
 <**link rel="stylesheet" href="{%static 'css/owl.carousel.min.css'%}"**>  
 <**link rel="stylesheet" href="{%static 'css/owl.theme.default.min.css'%}"**>  
 <**link rel="stylesheet" href="{%static 'css/magnific-popup.css'%}"**>  
  
 <**link rel="stylesheet" href="{%static 'css/aos.css'%}"**>  
  
 <**link rel="stylesheet" href="{%static 'css/ionicons.min.css'%}"**>  
  
 <**link rel="stylesheet" href="{%static 'css/bootstrap-datepicker.css'%}"**>  
 <**link rel="stylesheet" href="{%static 'css/jquery.timepicker.css'%}"**>  
  
  
 <**link rel="stylesheet" href="{%static 'css/flaticon.css'%}"**>  
 <**link rel="stylesheet" href="{%static 'css/icomoon.css'%}"**>  
 <**link rel="stylesheet" href="{%static 'css/style.css'%}"**>  
 </**head**>  
 <**body class="goto-here"**>  
  
 <**nav class="navbar navbar-expand-lg navbar-dark ftco\_navbar bg-dark ftco-navbar-light" id="ftco-navbar"**>  
 <**div class="container"**>  
 <**a class="navbar-brand" href="{%url 'index'%}"**>Study of Blockchain Technology in Farmer’s</**a**>  
 <**button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#ftco-nav" aria-controls="ftco-nav" aria-expanded="false" aria-label="Toggle navigation"**>  
 <**span class="oi oi-menu"**></**span**> Menu  
 </**button**>  
  
 <**div class="collapse navbar-collapse" id="ftco-nav"**>  
 <**ul class="navbar-nav ml-auto"**>  
 <**li class="nav-item active"**><**a href="{%url 'index'%}" class="nav-link"**>Home</**a**></**li**>  
 <**li class="nav-item"**><**a href="{%url 'SellerLogin'%}" class="nav-link"**>Seller</**a**></**li**>  
 <**li class="nav-item"**><**a href="{%url 'BuyerLogin'%}" class="nav-link"**>Buyer</**a**></**li**>  
 <**li class="nav-item"**><**a href="{%url 'AdminLogin'%}" class="nav-link"**>Admin</**a**></**li**>  
 <**li class="nav-item"**><**a href="{%url 'SellerRegister'%}" class="nav-link"**>SellerRegister</**a**></**li**>  
 <**li class="nav-item"**><**a href="{%url 'BuyerRegister'%}" class="nav-link"**>BuyerRegister</**a**></**li**>  
  
  
 </**ul**>  
 </**div**>  
 </**div**>  
 </**nav**>  
 *<!-- END nav -->* {%block contents%}  
  
 {%endblock%}  
 <**hr**>  
  
 <**footer class="ftco-footer ftco-section"**>  
 <**div class="container"**>  
 <**div class="row"**>  
  
 </**div**>  
  
 <**div class="row"**>  
 <**div class="col-md-12 text-center"**>  
 <**p**>  
 Copyright **&copy;**<**script**>document.write(new Date().getFullYear());</**script**> All rights reserved | This template is made with <**i class="icon-heart color-danger" aria-hidden="true"**></**i**> by <**a href="#" target="\_blank"**>Alex Corporations</**a**>  
  
 </**p**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**footer**>  
  
  
  
 *<!-- loader -->* <**div id="ftco-loader" class="show fullscreen"**><**svg class="circular" width="48px" height="48px"**><**circle class="path-bg" cx="24" cy="24" r="22" fill="none" stroke-width="4" stroke="#eeeeee"**/><**circle class="path" cx="24" cy="24" r="22" fill="none" stroke-width="4" stroke-miterlimit="10" stroke="#F96D00"**/></**svg**></**div**>  
  
  
 <**script src="{%static 'js/jquery.min.js'%}"**></**script**>  
 <**script src="{%static 'js/jquery-migrate-3.0.1.min.js'%}"**></**script**>  
 <**script src="{%static 'js/popper.min.js'%}"**></**script**>  
 <**script src="{%static 'js/bootstrap.min.js'%}"**></**script**>  
 <**script src="{%static 'js/jquery.easing.1.3.js'%}"**></**script**>  
 <**script src="{%static 'js/jquery.waypoints.min.js'%}"**></**script**>  
 <**script src="{%static 'js/jquery.stellar.min.js'%}"**></**script**>  
 <**script src="{%static 'js/owl.carousel.min.js'%}"**></**script**>  
 <**script src="{%static 'js/jquery.magnific-popup.min.js'%}"**></**script**>  
 <**script src="{%static 'js/aos.js'%}"**></**script**>  
 <**script src="{%static 'js/jquery.animateNumber.min.js'%}"**></**script**>  
 <**script src="{%static 'js/bootstrap-datepicker.js'%}"**></**script**>  
 <**script src="{%static 'js/scrollax.min.js'%}"**></**script**>  
 <**script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBVWaKrjvy3MaE7SQ74\_uJiULgl1JY0H2s&sensor=false"**></**script**>  
 <**script src="{%static 'js/google-map.js'%}"**></**script**>  
 <**script src="{%static 'js/main.js'%}"**></**script**>  
  
 </**body**>  
</**html**>

BuyerRegistrations.html

{%extends 'base.html'%}  
{%load static%}  
{%block contents%}  
<**section class="ftco-section ftco-no-pt ftco-no-pb py-5 bg-light"**>  
 <**div class="container py-4"**>  
 <**div class="row d-flex justify-content-center py-5"**>  
 <**div class="col-md-6"**>  
 <**center**>  
 <**h2 style="**font-size: 22px;**" class="mb-0"**>Buyer Register Form</**h2**>  
 <**span**>**&nbsp;**</**span**>  
  
 <**form action="{%url 'BuyerUserRegisterActions'%}" method="POST" class="text-primary"  
 style="**width:100%**"**>  
  
 {% csrf\_token %}  
 <**table**>  
 <**tr**>  
 <**td**></**td**>  
 <**td class="text-primary"**>Customer Name</**td**>  
 <**td**>{{form.name}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**>Login ID</**td**>  
 <**td**>{{form.loginid}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**>Password</**td**>  
 <**td**>{{form.password}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**>Mobile</**td**>  
 <**td**>{{form.mobile}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**>email</**td**>  
 <**td**>{{form.email}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**>Locality</**td**>  
 <**td**>{{form.locality}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**>Address</**td**>  
 <**td**>{{form.address}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**>City</**td**>  
 <**td**>{{form.city}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**>State</**td**>  
 <**td**>{{form.state}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**></**td**>  
 <**td**>{{form.status}}</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**></**td**>  
 <**td**>  
 <**button type="submit" value="Register" class="btn btn-primary"**>  
 Register  
 </**button**>  
 </**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>  
 <**div class="form-group mt-3"**>  
 <**span**>**&nbsp;**</**span**>  
 </**div**>  
 </**td**>  
 </**tr**>  
  
 {% if messages %}  
 {% for message in messages %}  
 <**font color='GREEN'**> {{ message }}</**font**>  
 {% endfor %}  
 {% endif %}  
  
 </**table**>  
  
 </**form**>  
 </**center**>  
  
 </**div**>  
  
 </**div**>  
 </**div**>  
</**section**>  
{%endblock%}