**SAMPLE CODE**

**Urls.py**

from django.contrib import admin

from django.urls import path

from .views import index,users,agents,admins,usersignup,agentsignup,logout

from users.views import bituserregister,userlogincheck,StartUserTrading,UserBuyQuantity,UserBuyingCoins,UserTransactionsHistory,UserPredictionTest,UserPredictTestProcess

from agents.views import bitagentregister,agentlogincheck,AgentBuyCrypto,agentbuycurrency,AgentTransactions,AgentHadCoins,AgentLedgerStatus,AgentPredectionTest,AgentredictTestProcess

from admins.views import adminlogincheck,viewusers,viewagents,activatewaitedusers,activatewaitedagents,currentrate,updatecryptocurrency,AdminGetLedger

urlpatterns = [

path('admin/', admin.site.urls),

path('',index,name='index'),

path('index/', index, name='index'),

path('users/',users,name='users'),

path('agents/',agents,name='agents'),

path('admins/',admins,name='admins'),

path('usersignup/',usersignup,name='usersignup'),

path('agentsignup/',agentsignup,name='agentsignup'),

path('logout/',logout,name='logout'),

path('bituserregister/',bituserregister,name='bituserregister'),

path('userlogincheck/',userlogincheck,name='userlogincheck'),

path('StartUserTrading/',StartUserTrading,name='StartUserTrading'),

path('UserBuyQuantity/',UserBuyQuantity,name='UserBuyQuantity'),

path('UserBuyingCoins/',UserBuyingCoins,name='UserBuyingCoins'),

path('UserTransactionsHistory/',UserTransactionsHistory,name='UserTransactionsHistory'),

path('UserPredictionTest/',UserPredictionTest,name='UserPredictionTest'),

path('bitagentregister/',bitagentregister,name='bitagentregister'),

path('agentlogincheck/',agentlogincheck,name='agentlogincheck'),

path('AgentBuyCrypto/',AgentBuyCrypto,name='AgentBuyCrypto'),

path('agentbuycurrency/<currencyname>',agentbuycurrency,name='agentbuycurrency'),

path('AgentTransactions/',AgentTransactions,name='AgentTransactions'),

path('AgentHadCoins/',AgentHadCoins,name='AgentHadCoins'),

path('AgentLedgerStatus/',AgentLedgerStatus,name='AgentLedgerStatus'),

path('AgentPredectionTest/',AgentPredectionTest,name='AgentPredectionTest'),

path('AgentredictTestProcess/<value>',AgentredictTestProcess,name='AgentredictTestProcess'),

path('adminlogincheck/',adminlogincheck,name='adminlogincheck'),

path('viewusers/',viewusers,name='viewusers'),

path('viewagents/',viewagents,name='viewagents'),

path('activatewaitedusers/',activatewaitedusers,name='activatewaitedusers'),

path('activatewaitedagents/',activatewaitedagents,name='activatewaitedagents'),

path('currentrate/',currentrate,name='currentrate'),

path('updatecryptocurrency/<curr>',updatecryptocurrency,name='updatecryptocurrency'),

path('AdminGetLedger/',AdminGetLedger,name='AdminGetLedger'),

path('UserPredictTestProcess/<value>',UserPredictTestProcess,name='UserPredictTestProcess'),

]

Admin side **models.py**

from django.db import models

import datetime

from django.utils import timezone

# Create your models here.

class cryptcurrencyratemodel(models.Model):

currencytype=models.CharField(max\_length=100, primary\_key=True)

doller=models.FloatField()

rupee=models.FloatField()

originalprice = models.FloatField()

def \_\_str\_\_(self):

return self.currencytype

class Meta:

db\_table = 'currencyrate'

class CurrencyUpdateModel(models.Model):

id = models.AutoField(primary\_key=True)

currencyname = models.CharField(max\_length=100)

conversionRate = models.FloatField()

newCurrencyValue = models.FloatField()

originalCurrencyValue = models.FloatField()

chnageValue = models.FloatField()

profitorloss = models.CharField(max\_length=50)

changedate = models.DateTimeField()

def \_\_str\_\_(self):

return self.currencyname

class Meta:

db\_table = 'currencychnagetable'

unique\_together = ('currencyname', 'changedate',)

Admins Side **Views.py**

from django.shortcuts import render,HttpResponse

from django.contrib import messages

from users.models import BitUserRegisterModel,BlockChainLedger

from agents.models import BitAgentRegisterModel

from .models import cryptcurrencyratemodel,CurrencyUpdateModel

import string

import random

from datetime import date

from django.db.models import Sum

# Create your views here.

def adminlogincheck(request):

if request.method=='POST':

usrid = request.POST.get('adminid')

pswd = request.POST.get('pswd')

print("User ID is = ", usrid)

if usrid == 'admin' and pswd == 'admin':

return render(request, 'admins/adminhome.html')

else:

messages.success(request, 'Please Check Your Login Details')

return render(request, 'admins.html')

def viewusers(request):

dict = BitUserRegisterModel.objects.all()

return render(request, 'admins/userslist.html', {'objects': dict})

def viewagents(request):

dict = BitAgentRegisterModel.objects.all()

return render(request,'admins/agentslist.html',{'objects':dict})

def activatewaitedusers(request):

if request.method=='GET':

id = request.GET.get('uid')

status = 'activated'

print("PID = ", id, status)

authkey = genSecretKey(8)

BitUserRegisterModel.objects.filter(id=id).update(status=status,authkey=authkey)

registerusers = BitUserRegisterModel.objects.all()

return render(request, 'admins/userslist.html', {'objects': registerusers})

def activatewaitedagents(request):

if request.method=='GET':

id = request.GET.get('uid')

status = 'activated'

print("PID = ", id, status)

authkey = genSecretKey(8)

BitAgentRegisterModel.objects.filter(id=id).update(status=status, authkey=authkey)

registerusers = BitAgentRegisterModel.objects.all()

return render(request, 'admins/agentslist.html', {'objects': registerusers})

def genSecretKey(stringLength=8):

"""Generate a random string of letters and digits """

lettersAndDigits = string.ascii\_letters + string.digits

return ''.join(random.choice(lettersAndDigits) for i in range(stringLength))

def currentrate(request):

dict = cryptcurrencyratemodel.objects.all()

dict2 = CurrencyUpdateModel.objects.all()

return render(request,'admins/cryptoratecurrent.html',{'objects':dict,'objects1':dict2})

def updatecryptocurrency(request,curr):

rate = request.GET.get('rate')

print('Rate = ',type(rate),' Currency ',type(curr))

incrementRate = float(rate)

if incrementRate>0:

check = cryptcurrencyratemodel.objects.get(currencytype=curr)

currentRate = check.doller

currentRupee = check.rupee

originalDollerrate = check.originalprice

originalRupee = check.originalprice

newRupee = (incrementRate \* currentRupee) / 100

newCurrencyVal = (incrementRate \* currentRate) / 100

print('Updated Currency ', newCurrencyVal)

today = date.today()

print("Today's date:", today)

# changes = newCurrencyVal - originalDollerrate

changes = newCurrencyVal + currentRate

newRup = newRupee + currentRupee

print("Chnages is ", changes)

currencygain = ''

if changes > currentRate:

currencygain = 'Gain'

else:

currencygain = "loss"

print('Currency is ', currencygain)

CurrencyUpdateModel.objects.create(currencyname=curr, conversionRate=rate, newCurrencyValue=changes,

originalCurrencyValue=originalDollerrate, chnageValue=changes,

profitorloss=currencygain, changedate=today)

cryptcurrencyratemodel.objects.filter(currencytype=curr).update(doller=changes, rupee=newRup)

dict = cryptcurrencyratemodel.objects.all()

dict2 = CurrencyUpdateModel.objects.all()

return render(request, 'admins/cryptoratecurrent.html', {'objects': dict, 'objects1': dict2})

elif incrementRate==0:

print("Please Check Yhe Conversion rate")

else:

print("Currency Decrease Starts")

check = cryptcurrencyratemodel.objects.get(currencytype=curr)

currentRate = check.doller

currentRupee = check.rupee

originalDollerrate = check.originalprice

originalRupee = check.originalprice

newRupee = (abs(incrementRate) \* currentRupee) / 100

newCurrencyVal = (abs(incrementRate) \* currentRate) / 100

print('Updated Currency ', newCurrencyVal)

today = date.today()

print("Today's date:", today)

# changes = newCurrencyVal - originalDollerrate

changes =currentRate - newCurrencyVal

newRup = currentRupee - newRupee

print("Chnages is ", changes)

currencygain = ''

if changes > currentRate:

currencygain = 'gain'

else:

currencygain = "loss"

print('Currency is ', currencygain)

CurrencyUpdateModel.objects.create(currencyname=curr, conversionRate=rate, newCurrencyValue=changes,

originalCurrencyValue=originalDollerrate, chnageValue=changes,

profitorloss=currencygain, changedate=today)

cryptcurrencyratemodel.objects.filter(currencytype=curr).update(doller=changes, rupee=newRup)

dict = cryptcurrencyratemodel.objects.all()

dict2 = CurrencyUpdateModel.objects.all()

return render(request, 'admins/cryptoratecurrent.html', {'objects': dict, 'objects1': dict2})

def AdminGetLedger(request):

check = BlockChainLedger.objects.aggregate(Sum('blockchainmoney'))

x = check.get("blockchainmoney\_\_sum")

x = round(x, 2)

print('Totoal Ledger Sum ', x)

dict = BlockChainLedger.objects.all()

return render(request, 'admins/adminsblock.html', {'objects': dict, 'ledger': x})

agent **models.py**

from django.db import models

from django.utils import timezone

# Create your models here.

class BitAgentRegisterModel(models.Model):

id = models.AutoField(primary\_key=True)

email = models.CharField(max\_length=100, unique=True)

pswd = models.CharField(max\_length=100)

username = models.CharField(max\_length=100)

mobile = models.CharField(max\_length=100)

pan = models.CharField(max\_length=100)

state = models.CharField(max\_length=100)

location = models.CharField(max\_length=100)

cryptcurrency = models.CharField(max\_length=100)

status = models.CharField(max\_length=100, default='waiting')

authkey = models.CharField(max\_length=100, default='waiting')

# cdate = models.DateTimeField(auto\_now\_add=True)

cdate = models.DateTimeField()

def \_\_str\_\_(self):

return self.email

class Meta:

db\_table = 'agentregister'

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

''' On save, update timestamps '''

if not self.id:

self.cdate = timezone.now()

return super(BitAgentRegisterModel, self).save(\*args, \*\*kwargs)

class AgentHadCrypto(models.Model):

id = models.AutoField(primary\_key=True)

currencyName = models.CharField(max\_length=100)

useremail = models.CharField(max\_length=100)

quantity = models.IntegerField()

def \_\_str\_\_(self):

return self.useremail

class Meta:

db\_table = "agentscryptoquantity"

unique\_together = ('currencyName', 'useremail',)

class AgentBuyCryptoModel(models.Model):

id = models.AutoField(primary\_key=True)

agentName = models.CharField(max\_length=100)

agentemail = models.CharField(max\_length=100)

currencyname = models.CharField(max\_length=100)

currentprice = models.FloatField()

quantity = models.IntegerField()

payableammount = models.FloatField()

cardnumber = models.CharField(max\_length=100)

nameoncard = models.CharField(max\_length=100)

cardexpiry = models.CharField(max\_length=100)

cvv = models.IntegerField()

cdate= models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.id

class Meta:

db\_table = 'AgentBuyedTransactions'

Agent side **views.py**

from django.shortcuts import render, HttpResponse,redirect

from .models import BitAgentRegisterModel, AgentHadCrypto,AgentBuyCryptoModel

from django.contrib import messages

from admins.models import cryptcurrencyratemodel, CurrencyUpdateModel

from users.models import BlockChainLedger

from django.db.models import Sum

from django.conf import settings

import os

from users.lstmann import predictionstart

from users.algo.generatedata import GetData

# Create your views here.

def bitagentregister(request):

if request.method == 'POST':

email = request.POST.get('email')

pswd = request.POST.get('pswd')

username = request.POST.get('username')

mobile = request.POST.get('mobile')

pan = request.POST.get('pan')

state = request.POST.get('state')

location = request.POST.get('location')

crypttype = request.POST.get('cryptocurrencies')

print("Valid Form = ", email)

try:

rslts = BitAgentRegisterModel.objects.create(email=email, pswd=pswd, username=username, mobile=mobile,

pan=pan, state=state, location=location,

cryptcurrency=crypttype)

if rslts is None:

print("Invalid Data ", rslts)

messages.success(request, 'Email ID already exist, Registration Failed ')

else:

print("Valid Data ", rslts)

messages.success(request, 'Registration Success')

except:

messages.success(request, 'Email ID already exist, Registration Failed ')

return render(request, 'agents/agentsignup.html', {})

else:

print("Invalid Form Data")

messages.success(request, 'Email ID already exist, Registration Failed ')

return render(request, 'agents/agentsignup.html', {})

def agentlogincheck(request):

if request.method == "POST":

email = request.POST.get('email')

pswd = request.POST.get('pswd')

print("Email = ", email, ' Password = ', pswd)

try:

check = BitAgentRegisterModel.objects.get(email=email, pswd=pswd)

status = check.status

print('Status is = ', status)

if status == "activated":

request.session['id'] = check.id

request.session['loggedagent'] = check.username

request.session['email'] = check.email

print("User id At", check.id, status)

return render(request, 'agents/agentpage.html', {})

else:

messages.success(request, 'Your Account Not at activated')

return render(request, 'users.html')

# return render(request, 'user/userpage.html',{})

except Exception as e:

print('Exception is ', str(e))

pass

messages.success(request, 'Invalid Email id and password')

return render(request, 'agents.html', {})

def AgentBuyCrypto(request):

dict = cryptcurrencyratemodel.objects.all()

dict2 = CurrencyUpdateModel.objects.all()

return render(request, 'agents/buycurrencybyagent.html', {'objects': dict, 'objects1': dict2})

def agentbuycurrency(request, currencyname):

quntity = int(request.GET.get('quantity'))

check = cryptcurrencyratemodel.objects.get(currencytype=currencyname)

currentPrice = check.doller

payableAmount = quntity \* currentPrice

print("1 Bitcoint value = ", currentPrice, " Currency is = ", currencyname, " Quanity = ", quntity,

" Payable Ammount = ", payableAmount)

dict = {

"currentPrice": currentPrice,

"currencyname": currencyname,

"quntity": quntity,

"PayableAmmount": payableAmount

}

return render(request, 'agents/agentbuycrypto.html', dict)

def AgentTransactions(request):

if request.method == 'POST':

currencyname = request.POST.get('currencyname')

currentprice = float(request.POST.get('currentprice'))

quantity = int(request.POST.get('quantity'))

payableammount = float(request.POST.get('payableammount'))

cardnumber = request.POST.get('cardnumber')

nameoncard = request.POST.get('nameoncard')

cardexpiry = request.POST.get('cardexpiry')

cvv = int(request.POST.get('cvv'))

agentName = request.session['loggedagent']

email = request.session['email']

agentQuantities = checkusercrypto(email, currencyname)

print("Agents Quantity ", agentQuantities)

if agentQuantities == 0:

print("AM in IF block")

AgentHadCrypto.objects.create(currencyName=currencyname, useremail=email, quantity=quantity)

else:

totalQuanty = int(agentQuantities) + quantity

print("AM in else block ",totalQuanty )

AgentHadCrypto.objects.filter(currencyName=currencyname, useremail=email).update(quantity=totalQuanty)

AgentBuyCryptoModel.objects.create(agentName = agentName,agentemail=email,currencyname=currencyname,currentprice=currentprice,quantity = quantity,payableammount = payableammount,cardnumber = cardnumber,nameoncard = nameoncard,cardexpiry = cardexpiry,cvv= cvv)

dict1 = AgentHadCrypto.objects.filter(useremail=email)

dict2 = AgentBuyCryptoModel.objects.filter(agentemail=email)

return render(request, 'agents/agentbuyed.html', {"object1": dict1, 'object2': dict2})

def checkusercrypto(useremail, currencyname):

qty = 0

try:

obj = AgentHadCrypto.objects.get(currencyName=currencyname, useremail=useremail)

qty = obj.quantity

except Exception as e:

qty = 0

print('Error is ', str(e))

return qty

def AgentHadCoins(request):

email = request.session['email']

dict1 = AgentHadCrypto.objects.filter(useremail=email)

dict2 = AgentBuyCryptoModel.objects.filter(agentemail=email)

return render(request,'agents/agentbuyed.html',{"object1":dict1,'object2':dict2})

def AgentLedgerStatus(request):

email = request.session['email']

check = BlockChainLedger.objects.aggregate(Sum('blockchainmoney'))

x = check.get("blockchainmoney\_\_sum")

x = round(x, 2)

print('Totoal Ledger Sum ',x)

dict = BlockChainLedger.objects.filter(agentemail=email)

return render(request,'agents/agentblock.html',{'objects':dict,'ledger':x})

def AgentPredectionTest(request):

dict = {}

dirName = settings.MEDIA\_ROOT

listOfFile = getListOfFiles(dirName)

# print('List Files ',listOfFile)

count = 0;

for x in listOfFile:

count += 1

x1 = os.path.basename(x)

dict.update({count: x1})

print('List Of Files = ', dict)

return render(request, 'agents/agentpredictTest.html', {'dict': dict})

def getListOfFiles(dirName):

# create a list of file and sub directories

# names in the given directory

listOfFile = os.listdir(dirName)

allFiles = list()

# Iterate over all the entries

for entry in listOfFile:

# Create full path

fullPath = os.path.join(dirName, entry)

# If entry is a directory then get the list of files in this directory

if os.path.isdir(fullPath):

allFiles = allFiles + getListOfFiles(fullPath)

else:

allFiles.append(fullPath)

return allFiles

def AgentredictTestProcess(request,value):

print("FIle is ",value)

fileName = settings.MEDIA\_ROOT + "\\" + value

print('Dataset Name is ', fileName)

obj = GetData()

list = obj.generateTrading()

# print("List Data is ",list)

pPath = settings.MEDIA\_ROOT + "\\" + "predections.txt"

with open(pPath, 'a') as f:

# f.write("Date,Open,High,Low,Close,Volume,OpenInt")

# f.write('\n')

for item in list:

for x in item:

f.write("%s," % x)

f.write('\n')

predictionstart(fileName)

return redirect("AgentPredectionTest")

user side **models.py**

from django.db import models

import datetime

from django.utils import timezone

# Create your models here.

class BitUserRegisterModel(models.Model):

id = models.AutoField(primary\_key = True)

email = models.CharField(max\_length=100, unique=True)

pswd = models.CharField(max\_length=100)

username = models.CharField(max\_length=100)

mobile = models.CharField(max\_length=100)

pan = models.CharField(max\_length=100)

state = models.CharField(max\_length=100)

location = models.CharField(max\_length=100)

status = models.CharField(max\_length=100,default='waiting')

authkey = models.CharField(max\_length=100,default='waiting')

#cdate = models.DateTimeField(auto\_now\_add=True)

cdate = models.DateTimeField()

def \_\_str\_\_(self):

return self.email

class Meta:

db\_table = 'userregister'

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

''' On save, update timestamps '''

if not self.id:

self.cdate = timezone.now()

return super(BitUserRegisterModel, self).save(\*args, \*\*kwargs)

class CustomerHadCoins(models.Model):

id = models.AutoField(primary\_key=True)

currencyName = models.CharField(max\_length=100)

customeremail = models.CharField(max\_length=100)

quantity = models.IntegerField()

def \_\_str\_\_(self):

return self.customeremail

class Meta:

db\_table = "CustomerContainCoins"

unique\_together = ('currencyName', 'customeremail',)

class UserBuyingCryptoModel(models.Model):

id = models.AutoField(primary\_key=True)

customername = models.CharField(max\_length=100)

email = models.CharField(max\_length=100)

currencyname = models.CharField(max\_length=100)

quantity = models.IntegerField()

agentemail = models.CharField(max\_length=100)

singlecoingamount = models.FloatField()

payableammount = models.FloatField()

cardnumber = models.CharField(max\_length=100)

nameoncard = models.CharField(max\_length=100)

cardexpiry = models.CharField(max\_length=100)

cvv = models.IntegerField()

cdate= models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.id

class Meta:

db\_table = 'UserBuyingCryptoModel'

class BlockChainLedger(models.Model):

id = models.AutoField(primary\_key=True)

customeremail = models.CharField(max\_length=100)

agentemail = models.CharField(max\_length=100)

currencyname = models.CharField(max\_length=100)

quantity = models.IntegerField()

paidammout = models.FloatField()

blockchainmoney = models.FloatField()

def \_\_str\_\_(self):

return self.id

class Meta:

db\_table = "BlockChainLedger"

user side **views.py**

from django.shortcuts import render, HttpResponse,redirect

from .models import BitUserRegisterModel, CustomerHadCoins, UserBuyingCryptoModel,BlockChainLedger

from django.contrib import messages

from agents.models import AgentHadCrypto

from admins.models import cryptcurrencyratemodel

from django.conf import settings

import os

import pandas as pd

import datetime as dt

from datetime import datetime

import matplotlib.pyplot as plt

from .lstmann import predictionstart

from .algo.generatedata import GetData

# Create your views here.

def bituserregister(request):

if request.method == 'POST':

email = request.POST.get('email')

pswd = request.POST.get('pswd')

username = request.POST.get('username')

mobile = request.POST.get('mobile')

pan = request.POST.get('pan')

state = request.POST.get('state')

location = request.POST.get('location')

print("Valid Form = ", email)

try:

rslts = BitUserRegisterModel.objects.create(email=email, pswd=pswd, username=username, mobile=mobile,

pan=pan, state=state, location=location)

if rslts is None:

print("Invalid Data ", rslts)

messages.success(request, 'Email ID already exist, Registration Failed ')

else:

print("Valid Data ", rslts)

messages.success(request, 'Registration Success')

except:

messages.success(request, 'Email ID already exist, Registration Failed ')

return render(request, 'users/usersignup.html', {})

else:

print("Invalid Form Data")

messages.success(request, 'Email ID already exist, Registration Failed ')

return render(request, 'users/usersignup.html', {})

def userlogincheck(request):

if request.method == "POST":

email = request.POST.get('email')

pswd = request.POST.get('pswd')

print("Email = ", email, ' Password = ', pswd)

try:

check = BitUserRegisterModel.objects.get(email=email, pswd=pswd)

status = check.status

print('Status is = ', status)

if status == "activated":

request.session['id'] = check.id

request.session['loggeduser'] = check.username

request.session['email'] = check.email

print("User id At", check.id, status)

return render(request, 'users/userpage.html', {})

else:

messages.success(request, 'Your Account Not at activated')

return render(request, 'users.html')

# return render(request, 'user/userpage.html',{})

except Exception as e:

print('Exception is ', str(e))

pass

messages.success(request, 'Invalid Email id and password')

return render(request, 'users.html', {})

def StartUserTrading(request):

dict = AgentHadCrypto.objects.all()

return render(request, 'users/UserTrading.html', {'objects': dict})

def UserBuyQuantity(request):

quantity = request.POST.get('quantity')

currencyname = request.POST.get('currencyname')

agentemail = request.POST.get('agentemail')

print("Crypto = ", currencyname, ' Agent Email = ', agentemail, ' Quantity = ', quantity)

getDollers = cryptcurrencyratemodel.objects.get(currencytype=currencyname)

coinPrice = getDollers.doller

blockchain = 11.5

bitBlock = (coinPrice \* blockchain) / 100

print("Block Bit Money ", bitBlock)

bitMoney = bitBlock + coinPrice

print("paid for 1 Bit ", bitMoney)

pay = float(quantity) \* bitMoney

dict = {

'quantity': quantity,

'currencyname': currencyname,

'agentemail': agentemail,

'bitBlock': round(bitMoney, 2),

'payableAmmount': round(pay, 2)

}

return render(request, 'users/userbuytranscation.html', dict)

def UserBuyingCoins(request):

if request.method == 'POST':

currencyname = request.POST.get('currencyname')

quantity = int(request.POST.get('quantity'))

agentemail = request.POST.get('agentemail')

singlecoingamount = float(request.POST.get('singlecoingamount'))

payableammount = float(request.POST.get('payableammount'))

cardnumber = request.POST.get('cardnumber')

nameoncard = request.POST.get('nameoncard')

cardexpiry = request.POST.get('cardexpiry')

cvv = int(request.POST.get('cvv'))

customername = request.session['loggeduser']

email = request.session['email']

oneBlock = 11.5

fetchBit = payableammount/100

blockChainAmmount = fetchBit\*oneBlock

print("Ledger balance ",blockChainAmmount)

updateAgentCoins(agentemail,currencyname,quantity)

userQuantity = checkusercrypto(email, currencyname)

print("Agents Quantity ", userQuantity)

if userQuantity == 0:

print("AM in IF block")

CustomerHadCoins.objects.create(currencyName=currencyname, customeremail=email, quantity=quantity)

else:

totalQuanty = int(userQuantity) + quantity

print("AM in else block ", totalQuanty)

CustomerHadCoins.objects.filter(currencyName=currencyname, customeremail=email).update(quantity=totalQuanty)

UserBuyingCryptoModel.objects.create(customername=customername, email=email, currencyname=currencyname,quantity=quantity, agentemail=agentemail, singlecoingamount=singlecoingamount, payableammount=payableammount, cardnumber=cardnumber, nameoncard=nameoncard, cardexpiry=cardexpiry, cvv=cvv)

BlockChainLedger.objects.create(customeremail=email,agentemail=agentemail,currencyname=currencyname,quantity=quantity,paidammout=payableammount,blockchainmoney=blockChainAmmount)

dict1 = CustomerHadCoins.objects.filter(customeremail=email)

dict2 = UserBuyingCryptoModel.objects.filter(email=email)

return render(request, 'users/userbuyed.html', {"object1": dict1, 'object2': dict2})

def checkusercrypto(useremail, currencyname):

qty = 0

try:

obj = CustomerHadCoins.objects.get(currencyName=currencyname, customeremail=useremail)

qty = obj.quantity

except Exception as e:

qty = 0

print('Error is ', str(e))

return qty

def updateAgentCoins(agentemail,currencyname,quantity):

check = AgentHadCrypto.objects.get(currencyName=currencyname,useremail=agentemail)

availableCquantity = check.quantity

balannceQua = availableCquantity - quantity

AgentHadCrypto.objects.filter(currencyName=currencyname, useremail=agentemail).update(quantity=balannceQua)

return availableCquantity

def UserTransactionsHistory(request):

email = request.session['email']

dict1 = CustomerHadCoins.objects.filter(customeremail=email)

dict2 = UserBuyingCryptoModel.objects.filter(email=email)

return render(request, 'users/userbuyed.html', {"object1": dict1, 'object2': dict2})

def UserPredictionTest(request):

dict = {}

dirName = settings.MEDIA\_ROOT

listOfFile = getListOfFiles(dirName)

# print('List Files ',listOfFile)

count = 0;

for x in listOfFile:

count += 1

x1 = os.path.basename(x)

dict.update({count: x1})

print('List Of Files = ',dict)

return render(request,'users/predictTest.html',{'dict':dict})

def getListOfFiles(dirName):

# create a list of file and sub directories

# names in the given directory

listOfFile = os.listdir(dirName)

allFiles = list()

# Iterate over all the entries

for entry in listOfFile:

# Create full path

fullPath = os.path.join(dirName, entry)

# If entry is a directory then get the list of files in this directory

if os.path.isdir(fullPath):

allFiles = allFiles + getListOfFiles(fullPath)

else:

allFiles.append(fullPath)

return allFiles

def UserPredictTestProcess(request,value):

print('Dataset Name is ',value)

fileName = settings.MEDIA\_ROOT+"\\"+value

print('Dataset Name is ', fileName)

obj = GetData()

list = obj.generateTrading()

#print("List Data is ",list)

pPath = settings.MEDIA\_ROOT+"\\"+"predections.txt"

with open(pPath, 'a') as f:

#f.write("Date,Open,High,Low,Close,Volume,OpenInt")

#f.write('\n')

for item in list:

for x in item:

f.write("%s," % x)

f.write('\n')

predictionstart(fileName)

return redirect('UserPredictionTest')

**lstmandann.py**

from admins.models import CurrencyUpdateModel

from users.models import UserBuyingCryptoModel

from django.db.models import Sum

from django.conf import settings

import os

import traceback

import csv

class GetData:

def \_\_init\_\_(self):

print("Invoke Once")

def generateTrading(self):

print("Trading Started")

check = CurrencyUpdateModel.objects.all()

list2 = []

for fk in check:

list = []

cdate = fk.changedate

dat = cdate.strftime('%Y-%m-%d')

open = fk.originalCurrencyValue

#high = fk.newCurrencyValue

convRate = fk.conversionRate

high = 0.0

low = 0.0

openint = 0

if convRate >0:

high = open \*convRate

openint = 0

else:

low = open\*abs(convRate)

openint=1

close = fk.newCurrencyValue

cname = fk.currencyname

check = UserBuyingCryptoModel.objects.filter(currencyname=cname).aggregate(Sum('quantity'))

Volume = check.get("quantity\_\_sum")

print('Volume is ',Volume)

list.append(dat)

list.append(open)

list.append(high)

list.append(low)

list.append(close)

list.append(Volume)

list.append(openint)

list2.append(list)

path = settings.MEDIA\_ROOT+'\\'+"ram.txt"

#print("FIle Path is ",list2)

return list2

**Base.html**

{% load static%}

<!DOCTYPE html>

<!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->

<!--[if IE 7]> <html class="no-js lt-ie9 lt-ie8"> <![endif]-->

<!--[if IE 8]> <html class="no-js lt-ie9"> <![endif]-->

<!--[if gt IE 8]><!-->

<html class="no-js"> <!--<![endif]-->

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta name="description" content="Bingo One page parallax responsive HTML Template ">

<meta name="author" content="Themefisher.com">

<title>Bit-Bank</title>

<!-- Mobile Specific Meta

================================================== -->

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Favicon -->

<link rel="shortcut icon" type="image/x-icon" href="{%static 'images/bitcoinnetwork.png'%}" />

<!-- CSS

================================================== -->

<!-- Themefisher Icon font -->

<link rel="stylesheet" href="{%static 'plugins/themefisher-font.v-2/style.css'%}">

<!-- bootstrap.min css -->

<link rel="stylesheet" href="{%static 'plugins/bootstrap/dist/css/bootstrap.min.css'%}">

<!-- Slick Carousel -->

<link rel="stylesheet" href="{%static 'plugins/slick-carousel/slick/slick.css'%}">

<link rel="stylesheet" href="{%static 'plugins/slick-carousel/slick/slick-theme.css'%}">

<!-- Main Stylesheet -->

<link rel="stylesheet" href="{%static 'css/style.css'%}">

</head>

<body id="body">

<!--

Start Preloader

==================================== -->

<div id="preloader">

<div class="preloader">

<div class="sk-circle1 sk-child"></div>

<div class="sk-circle2 sk-child"></div>

<div class="sk-circle3 sk-child"></div>

<div class="sk-circle4 sk-child"></div>

<div class="sk-circle5 sk-child"></div>

<div class="sk-circle6 sk-child"></div>

<div class="sk-circle7 sk-child"></div>

<div class="sk-circle8 sk-child"></div>

<div class="sk-circle9 sk-child"></div>

<div class="sk-circle10 sk-child"></div>

<div class="sk-circle11 sk-child"></div>

<div class="sk-circle12 sk-child"></div>

</div>

</div>

<!--

End Preloader

==================================== -->

<!--

Fixed Navigation

==================================== -->

<section class="header navigation">

<div class="container">

<div class="row">

<div class="col-md-12">

<nav class="navbar navbar-expand-md">

<a class="navbar-brand" href="index.html">

<img src="{%static 'images/logo.png'%}" alt="" width="150px"> &nbsp; Cryptocurrency Price Analysis With Artificial Intelligence

</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="tf-ion-android-menu"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav ml-auto">

<li class="nav-item ">

<a class="nav-link" href="{%url 'index'%}">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="{%url 'users'%}">Users</a>

</li>

<li class="nav-item">

<a class="nav-link" href="{%url 'agents'%}">Agents</a>

</li>

<li class="nav-item">

<a class="nav-link" href="{%url 'admins'%}">Admin</a>

</li>

</ul>

</div>

</nav>

</div>

</div>

</div>

</section>

<!--

Welcome Slider

==================================== -->

{%block contents%}

{%endblock%}

<!--

Start Blog Section

=========================================== -->

<section class="blog section" id="blog">

<div class="container">

<div class="row">

<div class="col">

<div class="title text-center">

<h4>Our untold story</h4>

<h2>Bitcoin Knowledge Base.</h2>

<span class="border"></span>

<p>As for the LSTM model, it has a comparable performance with the ANN model in general, when predicting the one-day future prices of these cryptocurrencies, based on mean square error. It demonstrates that although ANN is lack of internal capability, it could effectively extract and use the useful information hidden in the historical price dynamics to predict a future price.</p>

</div>

</div>

</div>

<div class="row">

<!-- single blog post -->

<article class="col-12 col-md-6" >

<div class="post-item">

<div class="post-thumb">

<img class="img-fluid shadow rounded" src="{%static 'images/blog/post-1.jpg'%}" alt="Generic placeholder image">

</div>

<div class="post-title">

<h3 class="mt-0"><a href="">Ten things about Business</a></h3>

</div>

<div class="post-meta">

<span>By</span> <a href="" class="">Wang Yiying</a>

</div>

<div class="post-content">

<p>Nowadays, investing in cryptocurrencies, like Bitcoin, is one of the efficient ways of earning money. For example, the rate of Bitcoin significant rises in 2017, from a relatively low point 963 USD on January 1ST 2017, to its peak 19186 USD on December 17th 2017, and it closed with 9475 USD at the end of the year. Consequently, the rate of return of bitcoin investment for 2017 was over 880%, which is an impressive and surprising scenery for most investors. </p>

</div>

</div>

</article>

<!-- /single blog post -->

<!-- single blog post -->

<article class="col-12 col-md-6" >

<div class="post-item">

<div class="post-thumb">

<img class="img-fluid shadow rounded" src="{%static 'images/blog/post-2.jpg'%}" alt="Generic placeholder image">

</div>

<div class="post-title">

<h3 class="mt-0"><a href="">Something I need to tell you</a></h3>

</div>

<div class="post-meta">

<ul class="list-inline">

<li class="list-inline-item">

<span>By</span> <a href="" class="">Zang Yeze</a>

</li>

<li class="list-inline-item">

<span>By</span> <span> 978-1-7281-3430-7/19/$31.00 ©2019 IEEE</span>

</li>

</ul>

</div>

<div class="post-content">

<p>Cryptocurrency is the peer-to-peer digital moneyory and payment system that exist online via a controlled algorithem. When a miner cracks an algorithem to record a block of transactions to public ledger named blockchain and the cryptocurrency is created when the block is added to the blockchain. It allows people to store and transfer through encryption protocol and distributed network. </p>

</div>

</div>

</article>

<!-- end single blog post -->

</div> <!-- end row -->

</div> <!-- end container -->

</section> <!-- end section -->

<!--

Start Call To Action

==================================== -->

<section class="call-to-action section-sm">

<div class="container">

<div class="row">

<div class="col-md-12 text-center">

<h2>Open account for free and start trading Bitcoins now!</h2>

<p>The price data comprised of four elements namely opening, high, low, closing prices. In this study, we analyse the price of three of the most popuer cryptocurrencies: Bitcoin, Ethereum and Ripple.</p>

<a href="" class="btn btn-main">Get Started</a>

</div>

</div> <!-- End row -->

</div> <!-- End container -->

</section> <!-- End section -->

<footer id="footer" class="bg-one">

<div class="footer-bottom">

<h5>Copyright 2019. All Rights Reserved.</h5>

<h6>Design and Developed by <a href="">Alex Hales</a></h6>

</div>

</footer> <!-- end footer -->

<!-- Main jQuery -->

<script src="{%static 'plugins/jquery/dist/jquery.min.js'%}"></script>

<!-- Bootstrap 3.7 -->

<script src="{%static 'plugins/bootstrap/dist/js/popper.min.js'%}"></script>

<script src="{%static 'plugins/bootstrap/dist/js/bootstrap.min.js'%}"></script>

<!-- Owl Carousel -->

<script src="{%static 'plugins/slick-carousel/slick/slick.min.js'%}"></script>

<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>

<!-- Smooth Scroll js -->

<script src="{%static 'plugins/smooth-scroll/dist/js/smooth-scroll.min.js'%}"></script>

<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCC72vZw-6tGqFyRhhg5CkF2fqfILn2Tsw"></script>

<!-- Custom js -->

<script src="{%static 'js/script.js'%}"></script>

<script src="{% static 'js/state.js' %}"></script>

</body>

</html>