Stock market trading app

→ 1 Technology used:

django.contrib messages for message pop up
Django Framework
javascript for global search in table
Django REST Framework
Crispy forms
Bootstrap version 5.2 CDN for frontend
Django Temaplate for frontend
Vertual environment
EMAIL_BACKEND = "django.core.mail.backends.console
For mongodb used djongo for connect with django project ORM
OS:Ubuntu

→ Requirement for setup django with versions as giref==3.5.2

beautifuls oup4 = = 4.11.1

charset-normalizer == 2.1.1

configparser = = 5.3.0

$$Django==4.1.1$$

django-bootstrap4==22.2

django-crispy-forms==1.14.0

django-filter==22.1

djangorestframework==3.13.1

djangorestframework-simplejwt==5.2.0

$$djongo==1.3.6$$

dnspython==2.2.1

$$ez-setup==0.9$$

idna==3.4

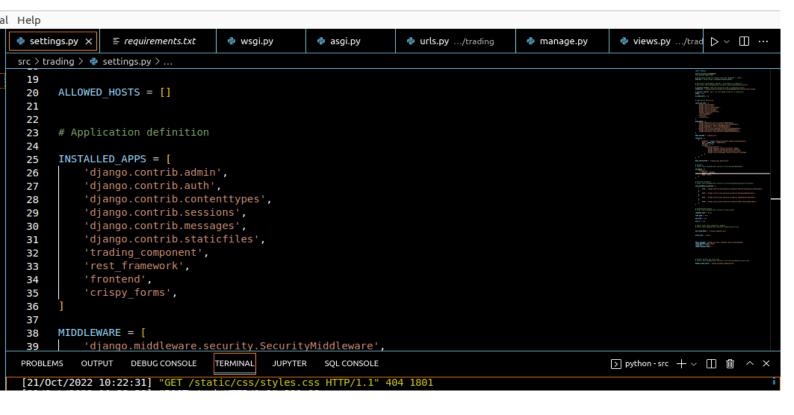
$$pytz = 2022.2.1$$

$$requests == 2.28.1$$

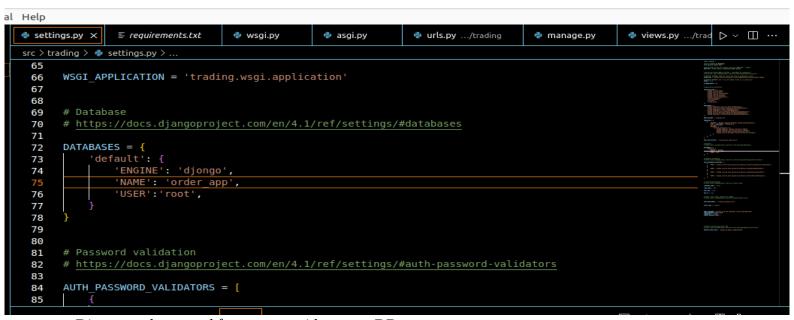
$$sqlparse==0.2.4$$

$$urllib3 = = 1.26.12$$

- → Django REST Framework
- 1. API Created
- → Order Create view
- → order List view
- → order Retrieve view(Detail view)
- → order update view
- → order Delete view
- → Used generic view for api
- → url for api : http://127.0.0.1:8000/api



Setting of installed app



Djongo package used for connect with mongo DB

```
al Help
  settings.py X

≡ requirements.txt

                                       wsgi.py
                                                      asgi.py
                                                                      urls.py .../trading
                                                                                           manage.py
                                                                                                           🕏 views.py .../trad
                                                                                                                          ▷ ~ □ …
  src > trading > 🥏 settings.py > ...
   111
          # Static files (CSS, JavaScript, Images)
   112
         # https://docs.djangoproject.com/en/4.1/howto/static-files/
   113
   114
   115
         AUTH USER MODEL = 'trading component.User'
   116
   117
   118
         STATIC URL = 'static/'
   119
   120
   121
   122
   123
   124
          EMAIL_BACKEND = "django.core.mail.backends.console.EmailBackend"
   125
         LOGIN REDIRECT URL="/home"
   126
          LOGIN_URL = "/login"
         LOGOUT_REDIRECT_URL="/"
   127
   128
   129
   130
   131
```

for Authuser used django auth User for Mail Django console emailbackend

```
al Help

≡ requirements.txt

                         wsgi.py
                                         asgi.py
                                                          urls.py .../trading
                                                                               manage.py
                                                                                                views.py .../trading_component X
                                                                                                                               ▷ ~ □ …
  src > trading_component > 💠 views.py > ...
          from django.shortcuts import render,reverse,redirect
          from .models import order
     2
     3
          from .serializers import orderSerializer
          from rest_framework.response import Response
          com rest_framework import status, generics
          import datetime
     8
     9
    10
    11
    12
    13
    14
          class orderView(generics.CreateAPIView):
              serializer_class = orderSerializer
    15
              queryset = order.objects.all()
    16
    17
              def post(self, request, format=None):
    serializer = orderSerializer(data=request.data)
    18
    19
                   if serializer.is valid():
    20
    21
                       serializer.save()
```

api GET and Post view

api GET and Post view

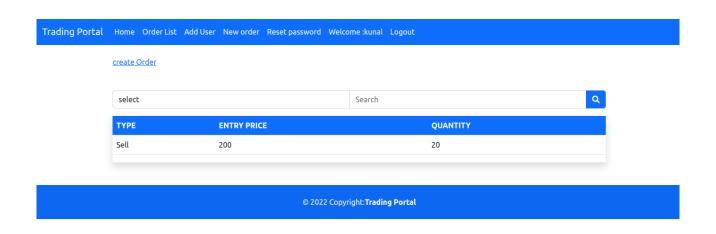
```
al Help
     ≡ requirements.txt
                                           asgi.py
                                                           urls.py .../trading
                                                                                 manage.py
                                                                                                  views.py .../trading_component X
                                                                                                                                  ▷ ~ □ …
                          wsgi.py
   src > trading_component > 💠 views.py > ...
    34
    35
          class OrderRetrieveView(generics.CreateAPIView):
    36
               serializer class = orderSerializer
    37
               queryset = order.objects.all()
    38
               def create(self, request, *args, **kwargs):
    39
                   queryset = order.objects.filter(pk=kwargs['pk']).first()
    40
                   if queryset:
    41
                        serializer = self.serializer_class(queryset)
    42
                    return Response({"status": True, "data": serializer.data}, status=status.HTTP_200_OK)
return Response({"status": False, "data": "Data not found"}, status=status.HTTP_404_NOT_FOUND
    43
    44
    45
    46
    47
          class OrderDetailUpdateView(generics.UpdateAPIView):
               serializer_class = orderSerializer
    48
    49
               def update(self, request, *args, **kwargs):
    50
                   queryset = order.objects.filter(pk=kwargs['pk']).first()
    51
                    if queryset:
                        serializer = self.serializer class(queryset, data=request.data, partial=True)
    52
    53
                        if serializer.is valid():
                             serializer.save()
```

Detail api view

```
al Help
           settings.py
                                                                         urls.py .../trading
                                                                                                    manage.py
                                                                                                                        views.py .../trading_component X
                                                                                                                                                                      □ …
                                wsgi.pv
                                                    asgi.py
    src > trading_component > 💠 views.py > .
                        return Response({"status": False, "data": "Data not found"}, status=status.HTTP 404 NOT FOUND
      58
      59
      61
             class OrderDetailDeleteView(generics.DestroyAPIView):
                  serializer class = orderSerializer
      62
                  def delete(self, request, *args, **kwargs):
    queryset = order.objects.filter(pk=kwargs['pk']).first()
      63
      64
      65
                         if queryset:
      66
                              serializer = self.serializer_class(queryset, data=request.data, partial=True)
      67
                              if serializer.is_valid():
      68
                                   queryset.delete()
                        return Response({"status": True, "data": "Data deleted successfully"}, status=status return Response({"status": False, "data": serializer.errors}, status=status.HTTP_400_BAD_return Response({"status": False, "data": "Data not found"}, status=status.HTTP_404_NOT_FOUND
      69
      70
      71
      72
73
74
75
```

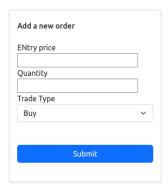
Trading Portal Login Signup				
	Username* Password*			
	Login			
	Don't have an account? Forgot password?			
© 2022 Copyright: Trading Portal				

Log in



Called table through get API

Go back to Home



© 2022 Copyright:**Trading Portal**

Add order through POST api

Username*						
Required. 150 characters or fewer.						
Letters, digits and @/./+/-/_ only.						
Password*						
Your password can't be too similar						
to your other personal						
information.						
 Your password must contain at 						
least 8 characters.						
 Your password can't be a 						
commonly used password.						
Your password can't be entirely						
numeric.						
Password confirmation*						
Enter the same password as before,						
for verification.						
Signup						

Signup

Django frontend api call view through django template

```
views.py .../frontend X
                                         settings.py
                        🕏 tests.py
                                                           wsgi.py
                                                                            asgi.py
                                                                                            urls.py .../trading
                                                                                                                   manage.py
                                                                                                                                 ▷ ~ □ …
src > frontend > ♦ views.py > ⊖ login_page
       def login_page(request):
  48
            form class=LoginForm
 49
  50
            message = ''
  52
            form=LoginForm
  53
            if request.method == 'POST':
  54
                 form = form class(request.POST)
  55
  56
                 if form.is_valid():
  57
                     user = authenticate(
                          username=form.cleaned_data['username'],
password=form.cleaned_data['password'],
  58
  59
  60
  61
                          login(request, user)
  62
                          message = 'Hello {user.username}! You have been logged in'
  63
  64
                          return redirect("/")
  65
            message = 'Login failed!'
return render(request, 'registration/login.html', context={ 'message': message, 'form':form})
  66
  67
  68
```

```
💠 tests.py
                                             settings.py
                                                                                                    urls.py .../trading
                                                                                                                            manage.py
                                                                                                                                           ▷ ~ □ …
views.py .../frontend X
                                                               wsgi.py
                                                                                  asgi.py
src > frontend > 💠 views.py > .
 71
        @login_required
 72
73
        def index(request):
             response=requests.get('http://127.0.0.1:8000/api', params=request.GET).json()
return render(request,'orders/order_list.html',{'response':response})
 74
 75
76
 77
        def home(request):
 78
             response=order.objects.all()
 79
 80
             print("Response :",response)
 81
             return render(request, 'books/book_list_student.html', {'response':response})
 82
 83
 84
```





```
→ views.py .../frontend X

→ tests.py

                                            settings.py
                                                              wsgi.py
                                                                                asgi.py
                                                                                                  urls.py .../trading
                                                                                                                         src > frontend > ♦ views.py > ⊖ orderview
        def create view(request):
 93
94
95
             entry_price = request.POST.get('entry_price')
             quantity = request.POST.get('quantity')
trade_type = request.POST.get('trade_type')
  96
            # entery_price = request.POST['entery_price']
payload = {
 97
 98
                  'entry_price': (entry_price),
'quantity': (quantity),
'trade_type': trade_type,
 99
 100
 101
                  'user': request.user.id
 102
103
104
105
             if request.method == 'POST':
 106
                  result = requests.post('http://127.0.0.1:8000/api', data=json.dumps(payload),
 107
108
109
                                              headers=get_header())
110
                  return redirect("/")
             return render(request, 'orders/order_create.html', {'response': r})
 113
```