<https://www.djangoproject.com/download/>

virtualenv venv | activate

pip install django

django-admin startproject quiz\_backend

python manage.py makemigrations

python manage.py migrate

python manage.py runserver

python manage.py createsuperuser

python manage.py startapp quiz

models:

class Quiz(models.Model):

owner = models.ForeignKey(User, on\_delete=models.CASCADE)

question = models.CharField(max\_length=255)

answer\_1 = models.CharField(max\_length=255)

answer\_2 = models.CharField(max\_length=255)

answer\_3 = models.CharField(max\_length=255)

answer\_4 = models.CharField(max\_length=255)

correct\_answer = models.IntegerField()

class Meta:

verbose\_name\_plural = 'quizes'

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

return self.id

Daha sonra rest\_framework dokümantasyonuna bakarak kur.

Ardından swagger kur.

Swagger’da quiz görünmesi için:

Serializer.py oluştur:

class QuizSerializer(serializers.ModelSerializer):

owner = serializers.ReadOnlyField(source='owner.id')

class Meta:

model = Quiz

fields = '\_\_all\_\_'

views:

from django.db import transaction

from django.shortcuts import render

# Create your views here.

from rest\_framework import generics

from quiz.models import Quiz

from quiz.serializer import QuizGetSerializer, QuizPostSerializer

class QuizList(generics.ListCreateAPIView):

def get\_serializer\_class(self):

if self.request.method == 'GET':

return QuizSerializer

if self.request.method == 'POST':

return QuizSerializer

return QuizSerializer

def perform\_create(self, serializer):

with transaction.atomic():

serializer.save(owner=self.request.user)

queryset = Quiz.objects.all()

class QuizDetail(generics.RetrieveUpdateDestroyAPIView):

def get\_serializer\_class(self):

if self.request.method == 'GET':

return QuizSerializer

if self.request.method == 'POST':

return QuizSerializer

return QuizSerializer

queryset = Quiz.objects.all()

urls.py:

from django.urls import path

from rest\_framework.urlpatterns import format\_suffix\_patterns

from quiz import views

urlpatterns = [

path('', views.QuizList.as\_view()),

path('<str:pk>/', views.QuizDetail.as\_view()),

]

urlpatterns = format\_suffix\_patterns(urlpatterns)

büyük urls içinde:

url(r'^api/v1/quizes/', include('quiz.urls')),

Daha sonra aynı işleri match için de yap. Ancak detail kısmına ihtiyaç yok.

Daha sonra oluşturma kısmını düzenlemek lazım. POST atınca kullanıcıya 3 soru atayıp mevcut soruyu göndermeli. /ongoing atınca kullanıcının mevcut maçını göndermelidir.

Bunun için ilk olarak post request atıldığında oyun varsa tekrar oluşturmamalıdır. Views şöyle düzenlenmeli:

class MatchList(generics.ListCreateAPIView):

def get\_serializer\_class(self):

if self.request.method == 'GET':

return MatchSerializer

if self.request.method == 'POST':

return MatchSerializer

return MatchSerializer

def post(self, request, \*args, \*\*kwargs):

current\_match = Match.objects.filter(Q(user=request.user) & Q(is\_defeat=False) & Q(is\_victory=False))

if current\_match:

serialized\_data = MatchSerializer(current\_match, many=True)

return Response(serialized\_data.data, status=200)

return self.create(request, \*args, \*\*kwargs)

def perform\_create(self, serializer):

with transaction.atomic():

saved\_instance = serializer.save(user=self.request.user)

quiz\_id\_list = Quiz.objects.values\_list('id', flat=True)

random\_quiz\_id\_list = random.sample(list(quiz\_id\_list), 3)

query\_set = Quiz.objects.filter(id\_\_in=random\_quiz\_id\_list)

for q in query\_set:

newQuiz = QuizSet(match\_id=saved\_instance.id, quiz=q)

newQuiz.save()

queryset = Match.objects.all()

devam eden oyun ve cevaplama mekanizması geliştirilecek. Bunun için views kısmına:

class MatchOngoing(generics.ListCreateAPIView):

serializer\_class = MatchSerializer

queryset = Match.objects.all()

ve url içine de ongoing/ endpointini ekle.

QuizSetSerializer ekle ve içine

from rest\_framework import serializers

from match.serializer import MatchSerializer

from quiz.serializer import QuizSerializer

from quiz\_set.models import QuizSet

class QuizSetSerializer(serializers.ModelSerializer):

quiz = QuizSerializer()

class Meta:

model = QuizSet

fields = '\_\_all\_\_'

Son olarak MatchOngoing i güncelle:

class MatchOngoing(generics.ListCreateAPIView):

serializer\_class = QuizSetSerializer

queryset = QuizSet.objects.all()

def list(self, request, \*args, \*\*kwargs):

current\_match = Match.objects.filter(Q(user=request.user) & Q(is\_defeat=False) & Q(is\_victory=False))

if current\_match:

current\_quiz = QuizSet.objects.filter(match\_\_in=current\_match).filter(answer=None).first()

serializer = QuizSetSerializer(current\_quiz)

return Response(serializer.data)

else:

return Response(data={'info': 'Devam eden oyun bulunmamaktadir'})

def post(self, request, \*args, \*\*kwargs):

try:

answer\_id = self.request.data['answer\_id']

except KeyError:

current\_match = Match.objects.filter(Q(user=request.user) & Q(is\_defeat=False) & Q(is\_victory=False))

if current\_match:

current\_quiz = QuizSet.objects.filter(match\_\_in=current\_match).filter(answer=None).first()

serializer = QuizSetSerializer(current\_quiz)

return Response(serializer.data)

else:

return Response(data={'info': 'Devam eden oyun bulunmamaktadir'})

with transaction.atomic():

current\_match = Match.objects.filter(Q(user=request.user) & Q(is\_defeat=False) & Q(is\_victory=False)).first()

if current\_match:

current\_quiz = QuizSet.objects.filter(match=current\_match).filter(answer=None).first()

is\_correct = current\_quiz.quiz.correct\_answer == answer\_id

current\_quiz.answer = answer\_id

current\_quiz.is\_correct = is\_correct

current\_quiz.save()

if not is\_correct:

current\_match.is\_defeat = True

current\_match.save()

return Response(data={'info': 'Kaybettin'})

else: #Eger sirada yeni bir soru yoksa oyun kazanilmistir

current\_quiz = QuizSet.objects.filter(match=current\_match).filter(answer=None).first()

if not current\_quiz:

current\_match.is\_victory = True

current\_match.save()

return Response(data={'info': 'Kazandın'})

else:

serializer = QuizSetSerializer(current\_quiz)

return Response(serializer.data)

else:

return Response(data={'info': 'Devam eden oyun bulunmamaktadir'})

Daha sonra postmane geç ve içeriğini anlat.

Front-end

React sitesini aç ve aşamaları uygula.

Sonra login için 2 input 1 button’lu form oluştur. Sonra component haline getir. onSubmit’i de props haline getirip başlık at.

ongoingHandle, startHandle, selectionHandle, onLogin http requestleri yazılacak. Bir noktada use-http hookunu da göster.

En sonda da localStorage’i göster.