Sport Relation Code and API Documentation

Introduction

Sport Relation API is an API to learn similarities between different sports and get a suggestion from API.

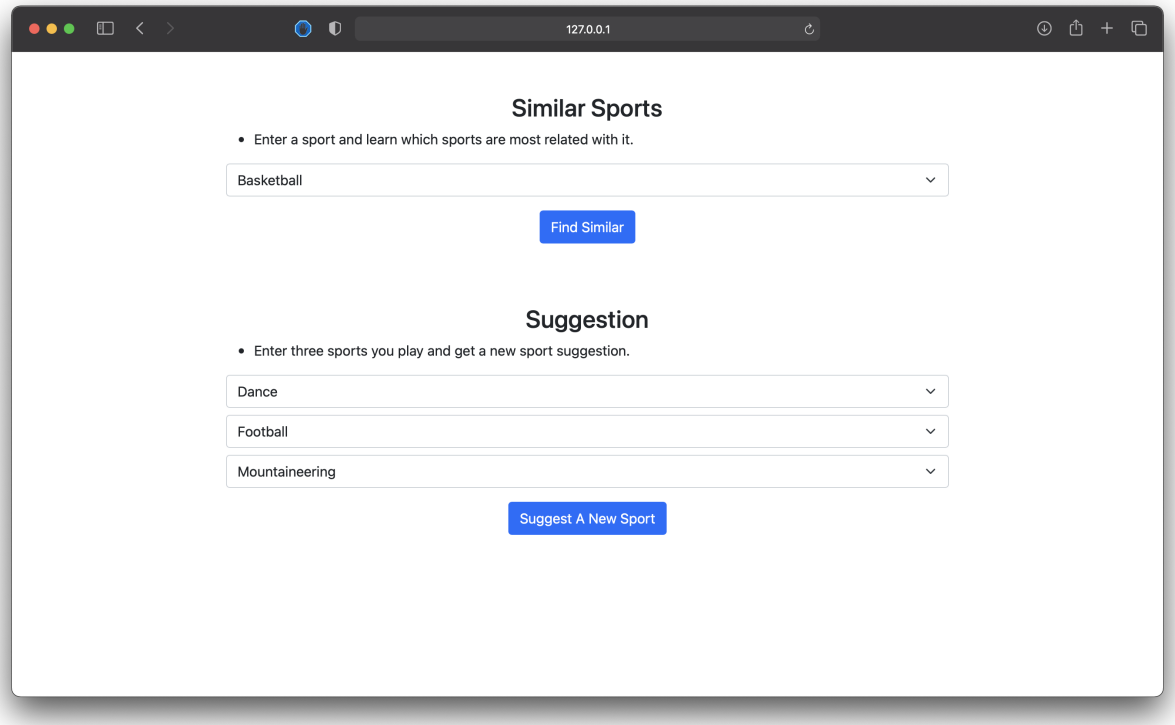
API is developed with Django Framework, and it uses PostgreSQL DBMS. Also, It uses Decathlon API.

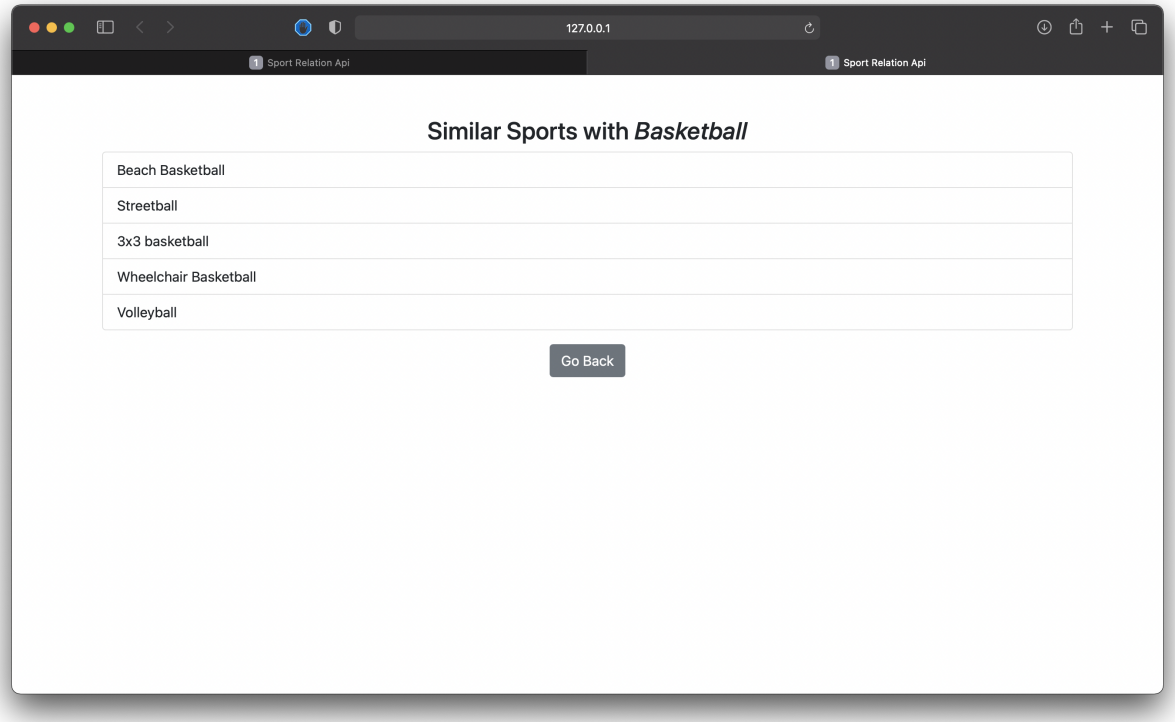
There are three capabilities of API.

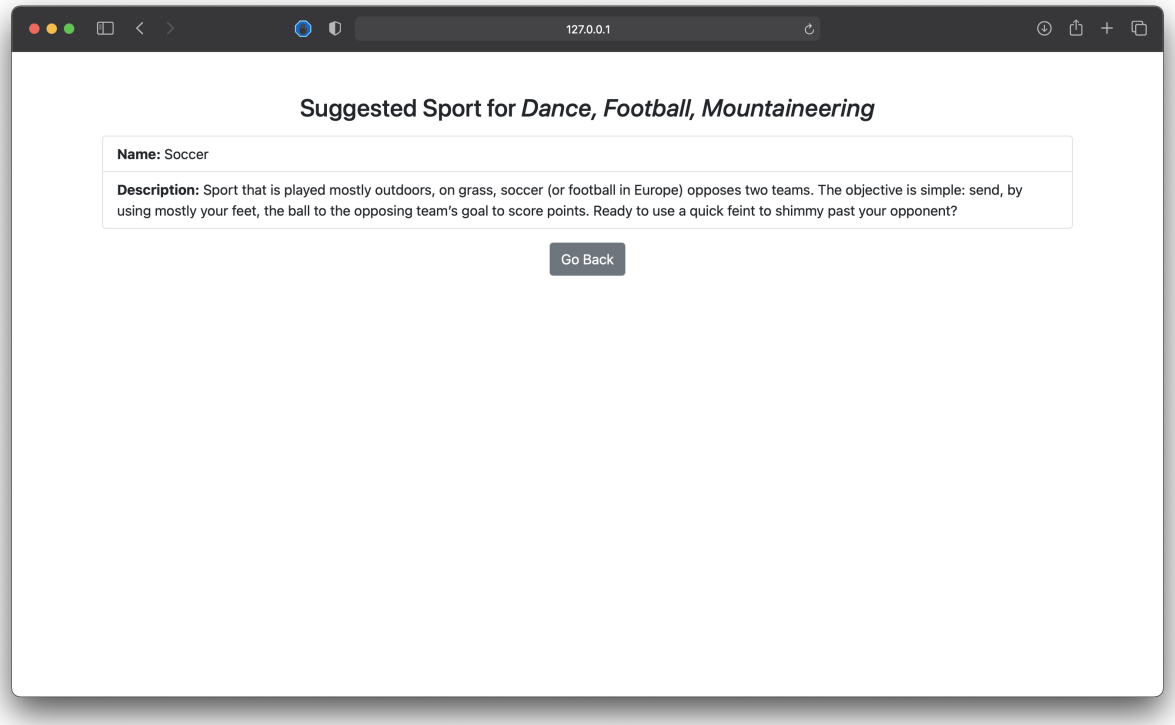
1. You can fetch a detailed sport information with a sport `pk`
2. You can learn top 5 similar sports to a sport by giving a sport `pk`
3. You can get a suggestion from API to learn which sport is the most relevant sport for you by giving three sports you interested in.

API has a user interface which developed with help of html templates rendering by Django. You can interact with API with help of UI.

Screenshots of UI:


127.0.0.1
Similar Sports
Enter a sport and learn which sports are most related with it.
Basketball
Find Similar
Suggestion
Enter three sports you play and get a new sport suggestion.
Dance
Football
Mountaineering
Suggest A New Sport


127.0.0.1
Sport Relation Api
Sport Relation Api
Similar Sports with Basketball
Beach Basketball
Streetball
3x3 basketball
Wheelchair Basketball
Volleyball
Go Back



How to run the app:

- Clone the repository
- Go to project folder
- Create and activate a virtual environment
- Install the requirements
- Set up a PostgreSQL database named `group5db`
- Run the command `python3 ./manage.py migrate` to create the tables in the database:
- Run the command `python3 ./manage.py sunserver` to start the app:
- Open the browser (http://127.0.0.1:8000/)
- Get suggestions and learn similarities between sports!
- Quit the app using ctrl+c
- You can test the app using the command `python3 ./manage.py test`

Sport Relation Code Documentation

```
class Sport(models.Model):
    link = models.CharField(max_length=2083, null=True)
    name = models.CharField(max_length=255, null=True)
    description = models.TextField(null=True)
    id = models.IntegerField(primary_key=True)
    slug = models.CharField(max_length=255, null=True)
    icon = models.CharField(max_length=2083, null=True)
```

Sport is an Django model which will be used to create a database table with column given as a field. It has necessary fields for a sport type.

```
class SportDetail(mixins.RetrieveModelMixin, generics.GenericAPIView):
    queryset = Sport.objects.all()
    serializer_class = SportSerializer

    def get(self, request, *args, **kwargs):
        return self.retrieve(request, *args, **kwargs)
```

`SportDetail` is an API View with `/sportRelation/api/sports/int:pk` endpoint. It has only one get method and it retrieve data from Sport model.

```
def get_related_sports(pk):
    url = base_url + str(pk)
    try:
        sports = requests.get(url).json()
        # if there is not any related sport (in this case decathlon api will not return a
 list for related field)
        if not isinstance(sports['data']['relationships']['related'], list):
            return []

        return [{
                **SportSerializer(Sport.objects.get(pk=x['data']['id'])).data,
```

```
                'weight': float(x['data']['weight'])
                } for x in sports['data']['relationships']['related']]
    except:  # If pk is not valid, return False
        return False
```

`get_related_sports(pk)` is a helper function which will be used for similar and suggest endpoints. It takes a p.k. and fetch related sports of a sport from Decathlon API. After, synchronize it with `Sport` model with help of `SportSerializer` . If there is not any sport for given `pk` value. It returns False.

```
class SimilarSport(APIView):  # Will return at most five similar sport with given pk

    def get(self, request, pk):
        related_sports = get_related_sports(pk)

        if related_sports == False:  # If pk is not valid, return Not Found
            return Response({"detail": "Not found."}, status=status.HTTP_404_NOT_FOUND)

        if len(related_sports) > 5:  # If there are more than five related sport return on
ly five of them
            return Response(related_sports[:5])

        return Response(related_sports)
```

`SimilarSport` is an API View with `/sportRelation/api/similar/int:pk` endpoint. It has only one get method and it returns at most five similar sport with given sport id.

```
class SuggestSport(APIView):

    def get(self, request, arg):
        pks = arg.split('-')
        suggestions = {}

        for pk in pks:  # Fetch similar sports one by one and add results to suggestions d
ictionary

            related_sports = get_related_sports(pk)

            if related_sports == False:  # If pk is not valid, return Not Found
                return Response({"detail": "Not found."}, status=status.HTTP_404_NOT_FOUN
D)
```

```
            for sport in related_sports:

                if sport['id'] in pks:  # we are looking for sports which are not choosen
 by user
                    continue

                # if we encounter with that sport before, update its weight
                if suggestions.get(sport['id']):
                    suggestions[sport['id']]['weight'] += sport['weight']
                else:  # if it is the first time we are encountering with that sport
                    suggestions[sport['id']] = sport

        suggestion = {}
        if suggestions:  # if suggestions dict is not an empty dict, return a suggestion w
ith maximum weight
            suggestion = max(suggestions.values(), key=lambda x: x['weight'])

        return Response(suggestion)
```

`SuggestSport` is an API View with `/sportRelation/api/suggest/int:pk` endpoint. It has only one get method and it takes three p.k. and get their similar sports. After, it combine them and returns sport which has maximum total weight.

```
class SaveSportListScript(APIView):
    permission_classes = [permissions.IsAdminUser]

    def post(self, request):
        url = base_url
        response = requests.get(url)  # fetch all sports
        sportlist = response.json()['data']

        sportlist = [{  # filter fields of sports
            'link': x['links']['self'],
            'name': x['attributes']['name'],
            'description': x['attributes']['description'],
            'id': x['id'],
            'slug': x['attributes']['slug'],
            'icon': x['attributes']['icon'],
        } for x in sportlist]

        ids = []

        for sport in sportlist:  # save fetched sports to datavase
            serializer = SportSerializer(data=sport)
            if serializer.is_valid():
                serializer.save()
                ids.append(sport['id'])
            else:  # if there is an array while saving the database, return HTTP_400
```

```
                return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

        # if all save operations are successfull, return their ids, with HTTP_201
        return Response({'AcceptedIds': ids}, status=status.HTTP_201_CREATED)
```

`SaveSportListScript` is an API View with `/sportRelation/api/save-sport-list-script` endpoint. It has only one post method and it is for only one time run. It can only be run by super admin to avoid possible security bug. It will fill the database with sports which are fetched from Decathlon API, and will filter necessary fields.

```
def index(request):  # Render home page
    sportlist = Sport.objects.order_by('name')

    context = {
        'sportlist': sportlist
    }

    return render(request, 'sport_relation/index.html', context)


def similar(request):  # Render similar sports page
    api_url = get_api_url(request.build_absolute_uri())
    pk = request.GET.get('sportlist', False)
    url = api_url + "similar/" + pk

    similar_sports = requests.get(url).json()
    root_sport = Sport.objects.get(pk=pk)

    context = {
        "similar_sports": similar_sports,
        "root_sport": root_sport
    }

    return render(request, 'sport_relation/similar.html', context)


def suggest(request):  # Render suggested sport page
    api_url = get_api_url(request.build_absolute_uri())
    pks = [request.GET.get('sportlist1', False), request.GET.get(
        'sportlist2', False), request.GET.get('sportlist3', False)]
    url = api_url + "suggest/" + "-".join(pks)

    suggestion = requests.get(url).json()
    root_sports = [Sport.objects.get(pk=pk) for pk in pks]

    context = {
        "suggestion": suggestion,
```

```
        "root_sports": root_sports
    }

    return render(request, 'sport_relation/suggest.html', context)
```

`index` `similar` and `suggest` are three view for user interface. They have similar jobs, they take a request fetch the data with given pk or pks, and give the data to html template in order to render the html to user.

Sport Relation API Documentation

Sport Detail

Endpoint: `GET /sportRelation/api/sports/:pk`

This endpoint retrieves a single sport. The pk parameter is a unique primary key for a sport.

Request Parameters

The only accepted (and required) parameter for this endpoint is the sport `pk`

Example for : `GET /sportRelation/api/sports/1`

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "name": "Duathlon",
    "description": "Because running and cycling are two perfectly complementary sports, du
athlon will make you work on different muscles while putting forward your endurance abilit
ies. Duathlon is a recreational and competitive solo or relay race, involving three segmen
ts: running, biking and running again. Will you have enough energy in your legs for the la
st portion?",
```

```
    "link": "/sports/1",
    "slug": "duathlon",
    "icon": null
}
```

Similar Sport

Endpoint: `GET /sportRelation/api/similar/:pk`

This endpoint retrieves at most five sports which are most similar sports with given sport in order. The pk parameter is a unique primary key for a sport.

Request Parameters

The only accepted (and required) parameter for this endpoint is the sport `pk`

Example for : `GET /sportRelation/api/similar/1`

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 391,
        "name": "Triathlon",
        "description": "Combining three endurance races, in order, swimming, cycling and r
unning, triathlon is practiced over various distances, includes various derivatives (cross
triathlon, winter triathlon, etc.) and allows you to take part in competitions alone or in
teams. It is an ideal sport to build up your whole body without ever getting bored. Becaus
e three sports are better than one!",
        "link": "/sports/391",
        "slug": "triathlon",
        "icon": "https://sports-api-production.s3.amazonaws.com/uploads/sport/icon/391/44
9.svg",
        "weight": 0.360548345171094
    },
    {
        "id": 900,
        "name": "Aquathlon",
        "description": "Running and swimming are the center of your sporting life? Why do
```

```
n’t you take part in an aquathlon race? This individual recreative or competitive two-stag
e race includes a swimming portion followed by a running portion. This is a perfect balanc
e to shape your legs and your arms!",
        "link": "/sports/900",
        "slug": "aquathlon",
        "icon": null,
        "weight": 0.25225590317391
    },
    {
        "id": 899,
        "name": "Winter triathlon",
        "description": null,
        "link": "/sports/899",
        "slug": "winter-triathlon",
        "icon": null,
        "weight": 0.238386339769658
    },
    {
        "id": 511,
        "name": "Long-distance running",
        "description": null,
        "link": "/sports/511",
        "slug": "long-distance-running",
        "icon": null,
        "weight": 0.215339218561191
    },
    {
        "id": 508,
        "name": "Adventure running",
        "description": null,
        "link": "/sports/508",
        "slug": "adventure-running",
        "icon": null,
        "weight": 0.215339218561191
    }
]
```


Suggest Sport

Endpoint: `GET /sportRelation/api/suggest/:pk1-pk2-pk3`

This endpoint retrieves a single sport which are the suggestion of the system (sport which has maximum total weight) for given three sport. The pk1-pk2-pk3 parameter is a string which contains three unique primary keys for three sports.

Request Parameters

The only accepted (and required) parameter for this endpoint is the `pk1-pk2-pk3`

Example for : `GET /sportRelation/api/suggest/1-4-5`

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 511,
    "name": "Long-distance running",
    "description": null,
    "link": "/sports/511",
    "slug": "long-distance-running",
    "icon": null,
    "weight": 0.790303418869382
}
```

Save Sport List Script

Endpoint: `POST /sportRelation/api/save-sport-list-script`

This endpoint saves necessary fields of all sports which retrieved from Decathlon API in order to fill database for future usages. It can only be called by super admin.

Example for : `POST /sportRelation/api/save-sport-list-script`

```
HTTP 201 Created
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "AcceptedIds": [
        282,
        284,
        285,
```

```
        224,
        286,
        ...
    ]
}
```