SpringBoard - Capstone 1 - Proposal

by: Etienne Deneault

Idea #1:

Lightweight Athlete/Performance Artist Management System

Application Main Function

- Provide coaches and performance enhancement specialist with a lightweight system to manage their teams/athletes/performance artists.

Application Secondary Function

- Provide coaches and athletes quick access to the most useful tools for Day to Day use.

Problems Solved

- There are many athlete management systems avaialble in the market but most have a difficult barrier of access for coaches and smaller athletic organizations. These "barriers" are due to the following: cost of access, complexity of implementation, complex tooling that generates a significant amount of work for the user/administrator.
- Many of the athlete mamagement systems do not offer easy access to features that coaches use on a daily basis. The result of this issue is that coaches do not use the functionality available because in "real-world" time it is too difficult to integrate into their coaching workflows.

Target Users

- The target user fir the application are Coaches working with smaller team organizations or a coach with a "single" or "few" athletes in individual sports

Application Features

No Auth Access

- Quick Access - *workout selector*
- Quick Access - *workout timer example configurations*
    - Timers built with dynamic javasript **OOP class Timer extended classes HiitTimer, RoundTimer, CircuitTimer**

With Auth Access

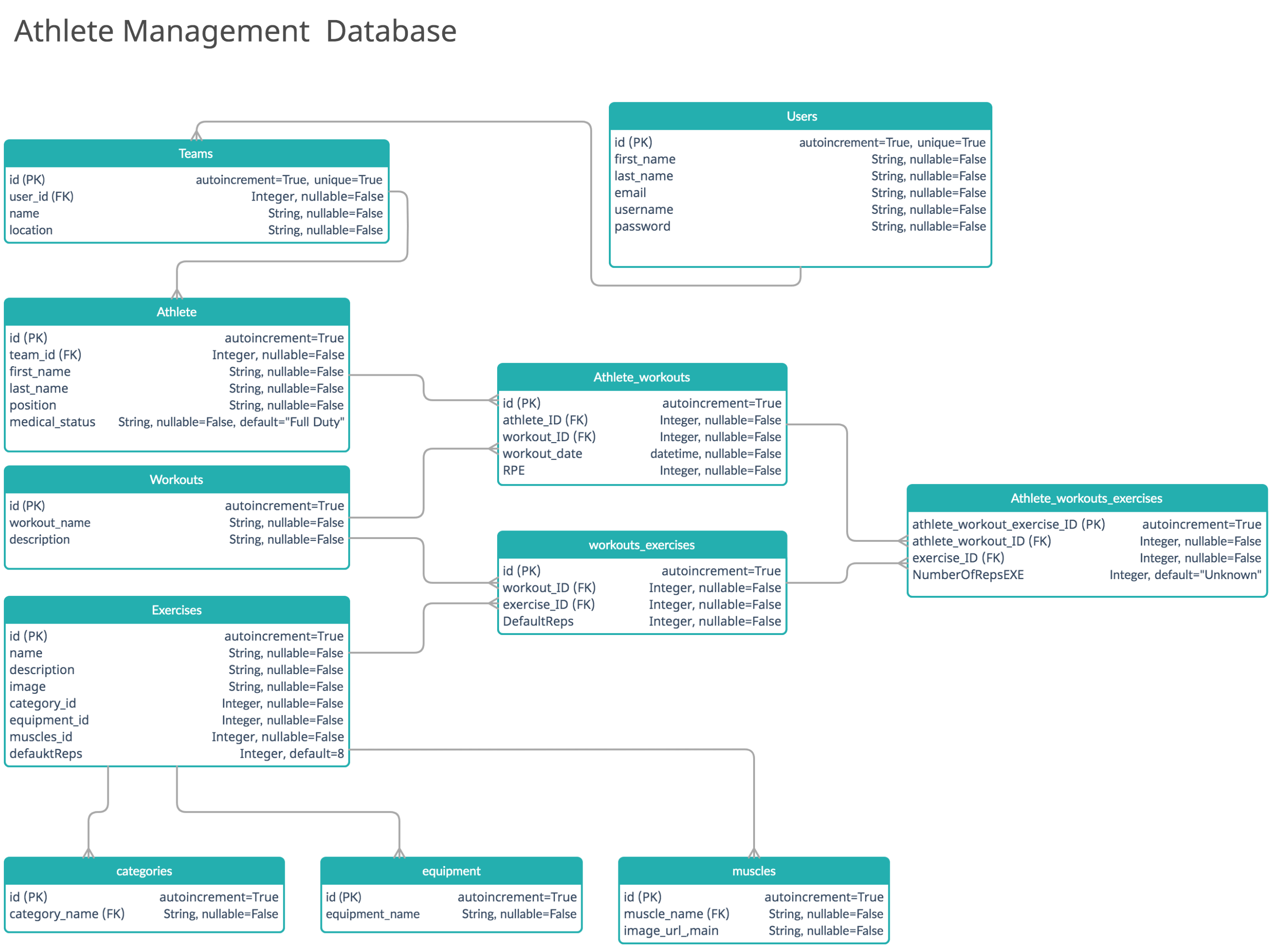
- Team/Athlete Managment (CRUD Teams, Athletes, Exercises, Workouts)

- Team/Athlete Management Dashboard - Team Data view and basic analysis (RPE, workload over time)
- CRUD workouts
- CRUD Timers
- CRUD training sessions
- CRUD training plans

API

- API to be used is: WEGR
  - > Public Endpoints to be used: exercise, exerciseinfo, exercisecategory, muscle, exercisecomment, exerciseimage, muscle, exerciseinfo, equipment

ATHLETE WORKOUT DATABASE SCHEMA


Athlete Management Database
Teams
id (PK) autoincrement=True, unique=True
user_id (FK) Integer, nullable=False
name String, nullable=False
location String, nullable=False
Users
id (PK) autoincrement=True, unique=True
first_name String, nullable=False
last_name String, nullable=False
email String, nullable=False
username String, nullable=False
password String, nullable=False
Athlete
id (PK) autoincrement=True
team_id (FK) Integer, nullable=False
first_name String, nullable=False
last_name String, nullable=False
position String, nullable=False
medical_status String, nullable=False, default="Full Duty"
Athlete_workouts
id (PK) autoincrement=True
athlete_ID (FK) Integer, nullable=False
workout_ID (FK) Integer, nullable=False
workout_date datetime, nullable=False
RPE Integer, nullable=False
Workouts
id (PK) autoincrement=True
workout_name String, nullable=False
description String, nullable=False
Athlete_workouts_exercises
athlete_workout_exercise_ID (PK) autoincrement=True
athlete_workout_ID (FK) Integer, nullable=False
exercise_ID (FK) Integer, nullable=False
NumberOfRepsEXE Integer, default="Unknown"
workouts_exercises
id (PK) autoincrement=True
workout_ID (FK) Integer, nullable=False
exercise_ID (FK) Integer, nullable=False
DefaultReps Integer, nullable=False
Exercises
id (PK) autoincrement=True
name String, nullable=False
description String, nullable=False
image String, nullable=False
category_id Integer, nullable=False
equipment_id Integer, nullable=False
muscles_id Integer, nullable=False
defauktReps Integer, default=8
categories
id (PK) autoincrement=True
category_name (FK) String, nullable=False
equipment
id (PK) autoincrement=True
equipment_name String, nullable=False
muscles
id (PK) autoincrement=True
muscle_name (FK) String, nullable=False
image_url_main String, nullable=False

- additional tables will be needed for medical_status, timers and training_plans.

API ENDPOINTS / ROUTES PLAN

Base

```
* GET /homepage
* 404 error_handler
```

No Auth - Timers and Workouts

```
* GET /timers
    * GET /timers/id
* GET /workouts
    * GET /workouts/id

***Authentification / Authorization***
* GET / POST /register
* GET / POST /login
* GET /logout
```

With Auth

```
* GET /dashboard
```

Users

```
* GET /users/username
* GET /users/logout
* GET / POST /users/username/add_team
```

Teams

```
* GET / POST teams
* GET teams/id
* GET / POST teams/id/edit
* GET / POST teams/id/delete
```

Athletes

```
* GET /POST athletes
* GET athletes/id
```

```
  * GET / POST athletes/id/edit
  * GET / POST athletes/id/delete
```

Exercises

```
  * GET / POST exercises
  * GET exercises/id
  * GET / POST exercises/id/edit
  * GET / POST exercises/id/delete
```

With Auth - Timers and Workouts

```
  * POST timers
  * GET / POST exercises/id/edit
  * GET / POST exercises/id/delete
  * POST workouts
  * GET / POST workouts/id/edit
  * GET / POST workouts/id/delete
```

Additional Routes for workouts, workouts_sesssions and training plans needed as well.

Technologies

- Python/Flask, PostgreSQL, QLAlchemy, Heroku, Jinja, RESTful APIs, JavaScript, HTML, CSS, WTForms, Bcryt