Collaborative Activity Tracker

Eerik Saksi - 2392230s

December 16, 2020

1 Status report

1.1 Proposal

1.1.1 Motivation

Inactivity is one of the modern leading killers, and despite all the science in favor of it, most do not exercise enough. This is largely due to the time commitment and discomfort caused by exercise. Many use activity trackers and apps to gamify exercise while others train with their friends. These make you accountable and make exercise funner.

1.1.2 Aims

Who's to say that gamified progress and group training are mutually exclusive? My RPG app lets teams fight an enemy together. Damage is calculated based on exercise performance and effort. Gamified progress makes exercise more fun and satisfying, but it also has more purpose beyond a score: it helps your team progress to the next enemy. There is also social obligation and accountability to stay consistent and do your part, as well as outgroup competition. There is also a certain time in which you need to defeat the enemy in, which creates time pressure.

1.2 Progress

- Postgres database with tables containing data on users, groups, their performance, etc.
- Set up Google authentication. Along side this, security policies on database rows based on identity
- Autogenerated GraphQL API from Postgres database using PostGraphile. Manually omitted some queries and fields, as well as created database functions that trigger when some queries are called.
- React Native (Expo) Typescript app that queries the PostGraphile API
- Login through server
- Groups that have an enemy of a certain level. Enemies have one of 8 different animated pixel sprites, as well as increasing health
- Profile updates (gender, weight)

- Relative strength calculations for hundreds of exercises given repetitions. This is used to calculate the damage that each attack deals.
- 6 different pixel animated avatars based on your relative strength (animated idle and attack).
- Workout tracking, which results in an animated fight with your character and the group's current enemy. The number of attacks your character does is based on workout difficulty. The damage is subtracted from your groups current enemies health, where going 0 or below takes your team to the next stage.
- Stats of all your teams workouts and personal records from the current battle, and every battle you have fought.

1.3 Problems and risks

1.3.1 Problems

- I am very glad I learned to use PostGraphile as it has very efficient performance and makes development super easy, but it also caused me most issues.
- It autogenerates queries such as deleteUser, which have no security by default. Reimplementing deleteUser with security would defeat the purpose of PostGraphile, as it is supposed to be automatic.
- I instead learned how to pass the user ID as context for the SQL query. I then implemented "row level security" which allows you to create rules for reading and writing to rows based on the passed user ID. As an example, anyone should be able to see how a group is doing, but only a validated user who belongs to that group should be able to update it.
- I also had to learn how to create database triggers. For example, instead of manually reimplementing createWorkout so that it also deals damage to the enemy, I created a trigger that subtracts damage from the current enemy whenever a workout is created.
- I swapped from pure React Native to Expo when I realized how annoying building and linking libraries would be.
- I also had some problems with the sprite sheets that I used. There was a lot of leftover whitespace around each art, which made it hard to hard to align elements. I tried for a while to find a programatic conclusion, but decided that since most spritesheets had minimal space around each frame, the best course of action would be to simply manually clip each row and column by the same amount for all sprites using a script that I wrote.

1.3.2 Risks

- So far I have done most of my testing with an Android emulator, and with the server running on localhost. **There will likely be some testing needed with real iOS and Android devices which connect to the server that is deployed on a cloud service.**
- The app might not be intuitive to use, there might be issues downloading it, and there might be issues with data collection. This is hard to say without human testing. **I should run a pilot study to make sure that the app is intuitive to use without my guidance.**

1.4 Plan

- Week 1: Prepare for a pilot study: make an ethics consent sheet, prepare questionares, make sure I can contact users through email or the app, so on.
- Weeks 2-3: I am already pretty close to a minimum viable product, and I do not feel that much pressure to make the best app as it isn't what is being tested. I will hopefully keep implementing "nice to have" features, testing and starting a pilot study with a couple people.
- Weeks 4-5: I should be done with the pilot study by now, so I will reserve this time to fix whatever issues arise from that, as well as any more "nice to have" features.
- Week 6-9: I will hopefully start sending out messages to some people asking to participate in the study. I will most likely not implement new features, but rather focus on making sure everything is running as expected and answering tech problems. I will also work on data analysis, coding responses and reviews, and so on. I will also focus on writing the report during this time.
- Week 10: Hopefully by this time I should have most the report finished. I should incorporate whatever new remaining findings from the data analysis over weeks 6-9.

1.5 Ethics and data

I have verified that the ethics checklist will apply to any evaluation I need to do. I will sign and complete the checklist. I will need to survey users after some time, asking them if they found the app and its collaborative aspects to be fun and motivating. I will also automatically track use and time spent on the app, as this will be helpful to see how much users engage with collaborative aspects of the app. I hope to implement some kind of chat which I will also log.