Logan Roach
RabbitCard, as it stands now (8/13/2021)

PGA API
Here are the links to the Production and Test accounts for the PGA API:
PROD - https://tourapi.pgatourhq.com:9443/devportal/apis
TEST - https://api-test-sites.pgatourhq.com:9443/devportal/apis

The login credentials:
Username – "RabbitCard"
PW – "7NbDxFTvU457"

Inside this account, there are different applications. The "syndsctest" application is what the current access token applies to. All the APIs I have used thus far are housed inside this app.

In the GitHub repository, you can go into api/src/datagrab and see the JS files I'm using to grab a lot of the data. They are bucketed by what I believe will be the correct separation for making calls.

Basefillers.js
This is the file containing the API calls that I believe will need to be made least frequently. In here, I call data for tournaments, getting information such as where it is being played, information on that course, the world golf rankings for players that week, and close out last week's tournaments as completed. I imagine this will need to be called once maybe twice a week.

Dailyfillers.js
Inside this file, I update the field of players for that week's tournaments. Everyday throughout the week, new golfers will drop out, be added in, or qualify for the tournament, so this will be useful for that constant variation Monday - Wednesday before the tournament starts on Thursday

Weekend.js
Here, I grab the groups for a given tournament. This is something we can call on Tuesdays or Wednesdays when groupings of golfers are released for Thursday and Friday. They will need to be called again Friday and Saturday evening when the following day's groupings are released. An important aspect of this gaming app is picking between golfers for a certain task (i.e. better score, closest to the pin, etc), so groupings is kind of a two-fold category. For people wanting to play the game in real-time, picking between golfers playing together is a great option because you see the result in real-time, as the golfers are playing holes together. Another component could be creating pairings based on their rating (found in the worldgolfrankings table). This takes away the real-time result aspect of the game but could create more interesting and competitive matchups.

Realtime.js

Keeping real time stats is very important, so this file will manage that aspect. Getting golfers stats and scores on holes as they play them will keep the app engaging for users who want to play as the day of golf progresses. I believe a five or ten minute loop for this request will be necessary all day Thursday - Sunday.

For the API, the rate limit for requests is approximately 1/minute. When wanting to make successive requests, I've had to put in waiting periods of 65 seconds just to prevent a max out. I believe Heroku Schedule is powerful enough add-on to schedule these files to be run in the correct time intervals. Obviously a lot of calls and changes will be made to the database everyday. I was thinking caching might be a smart option to limit the work being done, but I'm not super familiar with doing this, so if it is a bad idea or doesn't work here, no worries in ruling it out. Also as a side note, the files I detailed above aren't super cleaned up and definitely have some hard-coded aspects as I was testing them, so that is something I can clean up and change after you all take a look.

Postgres DB

The schema for this database can be found in sql/init in the repository. As for explaining the purpose of each table:

Users, Roles, and the user_roles tables are created by the Sequelize user auth models I am using.

courses - information on each golf course - some courses are in here multiple times as they have changed in between tournaments played at them

golfers - information of individual golfers on the PGA tour

tournaments_completed - information on tournaments

tournament_field - table keeping track of which golfers competed at each event

winners - table connecting tournaments to golfers based on which golfer(s) won each tournament

groupings - table connecting grouping golfers into the groups that they played in during certain rounds at certain tournaments

tournament_hosts - table connecting tournaments to the course(s) that hosted it

tournament_holes - table connecting individual golf holes to the course they are at, as well as the tournament that course was used for (yardages and par-values) can change for different events.

scoreboard - table keeping a log of total scores for rounds from golfers at events

scorecards - table keeping log of hole by hole breakdown of scores from golfers at events

worldgolfrankings - table that has the most up date world golf rankings, year by year

cttp - table keeping track of the closest to the pin games. Each cttp game is a grouping of golfers from a given round of a given tournament. The plan here is each row is a golfer from a grouping during a round at a tournament with an identifier similar to the other golfers in his grouping and a boolean detailing if that golfer won his grouping's game or not

user_pick_cttp - table keeping track of users' picks for given cttp games

Table(s) to be added still: stats for games like cttp. I believe the infrastructure is already in place to support a match play style game.

NodeJS (api)

Sequelize is utilized within the Node application for user auth/signin. Beyond that, Express has been used to create a few REST endpoints for the database. If you look in the **graphql** folder though, there is a pretty developed graphql/joinmonster infrastructure if that is a route people are more interested in taking. Both are great options that complete the same task, and I'm good to pick between either, so you all let me know.

React-Native (rabbitfront1)

A very basic react-native frontend has been spun up in this folder. I took a crack at some patchwork user signin/signup functionality, but I don't believe it is a secure or efficient as it needs to be. Beyond that, the app works for moving in between pages. Just replacing the page content with our queried data will be the next step. Grant took a stab at correcting some of the dimensions of the pages, but I believe there is an easier and better way to do this. The JS files were converted from XD files using Fireblade, so a lot of their formatting and styling is very inefficient and odd.

Please reach out with any and all criticisms/changes that be made. I'm looking forward to working with you all.