REGIONAL BOOKING PLATFORM FUNCTIONAL SPECIFICATIONS

GÜDLFT

THE WHY

We aim to create a more lightweight version of our flagship booking system for local and regional competition organizers. This should enable the organizer to focus less on admin and more on the logistics of the competition, ensuring events are safe and effective for all. The app will allow clubs to enroll athletes in competitions organized within the division.

THE HOW

Rather than athletes booking directly with the competition organizer, club secretaries will be able to book places for them. Only secretaries will have access to the app and be able to see how many places remain.

Currently, clubs earn points for assisting in the setup and running of competitions. Each club can see their current balance and redeem points to enroll athletes at future competitions at a rate of 1 point per entry. Each competition will have a limited number of entries, and each club can only enter a maximum of 12 athletes.

Sample walkthrough:

1. Secretary logs into the app.
2. Secretary identifies an upcoming event and clicks on it.
    a. Sees the number of entries available and whether or not more can be accepted.
    b. The secretary can then use their accrued points to purchase places for the competition.
        i. If there are enough points and space permits, they should see a confirmation message. The used points are then deducted from their account.
        ii. They should receive an error message if they do not have enough points, try to purchase more than 12 places, or the competition is full.
    c. Can view the list of points across other clubs.

PHASE 0

- Project kick-off
- Conference calls with organizations:
    - Need a way for secretaries to log in and view their current point balance.
    - Need to be able to use points to buy early access places.
    - Should avoid overbooking.
    - Need a board showing points for all clubs (there have been calls of favoritism).
- Using grayscale wireframes for demo.

PHASE 1

- Club secretaries for the organization will be able to use their email address to login and view a list of upcoming competitions.
    - They will then be able to select a competition and use their points to purchase places.
    - They should see a message either confirming the number of places purchased, or a message saying the competition is full. Used points should be deducted from the previous total.
    - They should not be able to book more places than available or more than 12 spots in one competition (to allow fairness to other clubs).
- Club secretaries will be able to logout.

PHASE 2

- For transparency, there should be a public, read-only point totals board showing the number of points available for each club. There should be no need to login to the site to see the page.
- Given the number of potential users, keep build and rendering times to a minimum; it should take no more than 5 seconds to retrieve a list of competitions, and no more than 2 seconds to update the points total.

ANNEX - GÜDLFT DEVELOPMENT HOW-TO

Lift your project to new heights by following our house coding style and the following rules. Standardizing the way we create projects and branches in GitHubmakes it easy for other people to read and understand project progress, whether it's QA, users, or other developers.

1. When creating your project, **always** add a README. This will help you and the rest of the team remember why you were here. Be sure to include things like:
   a. Links to external resources.
   b. Particular setups.
   c. Naming conventions.
   d. Anything that would require someone to come over and say, "I have a question about X."

2. The main (master) branch is your source of truth (finished code). Anyone should be able to clone from the main branch and run the project correctly.

3. If you want to add a feature, fix a bug, or anything that would break Rule # 2, **create** a branch. The branch name should cite the type of addition and follow the format <feature/bug/enhancement>/descriptive-name. For example, if you were to find a bug, you would create a branch called bug/name-of-bug, and fix it (obviously).

4. Testing
   a. Always be testing. If you haven't tested it, it's broken.
   b. We're framework agnostic-you can use pytest, unittest, or Morelia. No matter what you use, you should be able to run it from the command line.
   c. Please put all tests in a test folder. No one wants to hunt around code to find these.
   d. Unit tests >> integration tests >> functional tests. If nothing else, write a unit test.
   e. We aim to cover our code with at least 60% tests. More tests mean more sleep.
   f. Try to write twice as many unit tests than integration or functional tests. Unit tests let us know *how* things work and when they don't.
   g. To save eye pain, please group unit, integration, and functional tests into separate folders. You'll be glad you did.

5. Focus on one bug/feature/enhancement at a time (one per branch)! Working on multiple things can make your head hurt. Plus, it makes it hard for people to know what they can work on.

6. Merge back into the main branch only when all your tests pass. Remember, if it's not working, it doesn't belong in the main.

7. When you are ready for code review, create a QA branch from the main branch. Unlike the other branches, however, this does **not** merge back into the main.