Runner Assist Dashboard

Group 6 -TRIMM - Responisive Web App

> This repo is NOT A CODE REPOSITORY this should never have code pushed to it. We are using this to unify documentation and maintain our old combigned codebase incase we end up needing it, or the git history of it. The code in this repository comes from submodules that can be cloned and worked on seprately.

Project

- runner-assist-frontend
  - Frontent module for the runner assist application
  - Built in React
  - Data management with Redux
  - Using materialize CSS framwork
- runner-assist-server
  - Backend module for the runner assist application
  - Writen in Java
  - Implementing RESTful services with Java Serverlets
  - Uses json data formating for HTTP request & response
  - Secured with json web tokening
- Documentation
  - Wiki for information on both modules
  - Contains all design documents
  - Contatins reflections on Sprints
- Issues
  - Story/task/bug tracking
  - Using Agile/Sprint planning
  - Current Sprint Overview

Development Notes

Contributing

- Working on frontend
  - `git clone git@git.snt.utwente.nl:kindadysfunctional/runner-assist-frontend.git`
- Working on backend
  - `git clone git@git.snt.utwente.nl:kindadysfunctional/runner-assist-server.git`
- Try to keep to a 1 feature per branch model
  - makes merging easier
  - makes looking at history easier
  - makes writing merge request easier
  - makes writing and passing unit tests simpler (if we ever have testing)
- Try to make your branch names relate to the feature that you are working on
  - create a new branch for a new feature
- Merge requests on master
  - Use an informative title that relates to the feature you are adding
  - Note all major changes that you have made
  - Always check the box to `squash commits` and `delete branch`
  - Everyone has the ability to merge their own merge requests, with this great power comes great responsibility.
    - Make sure that the application will compile and run with your changes
    - If applicable make sure that your changes are covered by testing and are passing said tests
- Don't break things.

Issue tracking

- Creating a new issue
  - tag it correctly
  - (possibly) add it to the current sprint by setting the milestone
  - add detail notes if applicable

- Bugs
  - use the bug tag
  - assign it to the applicable person
  - (possibly) let them know in discord
- Stories
  - same as issues
  - make sure it has a point value
  - add some sort of acceptance criteria

WIKI & Documentation

- API
  - Every api should have a wiki page detailing what it does
  - Should have what the call requires
  - Should have what the call returns

Testing

- Backend
  - GOAL - >50% unit test coverage on codebase
  - Goal - >80% unit test coverage of all API methods
  - Goal - setup git to aaaauto run testing on merge requests
  - Every api method should have a JUnit test
  - Current (minimum) every api endpoint should be tested with Postman before being merged with master
- Frontend
  - (minimum) visual testing
  - get a second opinion on UI changes, but don't expect it to be a educated opinion
  - If time explore frontend testing