6-Documentation of reviews and meetings with stakeholders

Meeting With Stakeholder

Meeting notes for January 18th

- **Brief summary of beap engine then goals and timeline**

Some research is on commercial wearable devices.

Intersection of measurements of heartrates and other stuff by some wearable devices, different devices have different measurements

Beap engine made to allow laymen person to upload their own data in a file format and it returns a csv.

Includes a machine learning algorithm that would produce an open source activity level estimate for say sitting, running, etc that would have a standardized output for Fitbit and Apple watches.

Project status: had a fully functional first implementation that never really was touched for a while due to life circumstances. He now wants to resume it.

He shows us video

Existing codebase is in 2 github repos and data hosted on drop ocean

Goal:

1. wants to get everything back up and running the existing webapp
2. extensive documentation is highly desired
3. requires feedback from the dev team on what made our job easier or harder.
4. add extra coverage for new devices (garmin)
5. potentially make an app that lives on your phone because its easier
6. Any kind of speed(performance) improvements are helpful because it was slow.

Meeting notes for January 24th

High-level Overview of what that R doing (will have recorded Zoom)

Training data: aggregate)fitbit_applewatch_jager.csv

activity_trimmed variable is what we are trying to predict

DataProcessor.R files hold the logic for parsing uploaded data

inputs: XML and json

Predictor.R holds all the ML magic

rds files are R specific files

We are given explicit permission to buy/spin up Ubuntu server on Digital Ocean

Need to decide on a license for this project. (High Priority)

https://choosealicense.com/

CC BY-NC https://creativecommons.org/share-your-work/cclicenses/

open to just about any other up-to-date license we can add to a repo

The ethics statement on the home page needs updating (Low Priority)

Peer Review Documentations

Document Inspection of Code Inspection Checklist

Reviewers: **Ardalan Askarian Ralph Gregorio**

Reviewed: **Kamal Zrein**

Brief Description:

Kamal Wrote down a Code Inspection Checklist where he put checklists for Java & TypeScript and goes through different checklist in details by checking the: 1- Variable Declaration 2- Methods and method signatures including return and input types 3- Class definitions and grouping into packages (Java) 4- Control flow Defects 5- Code style & practices. Moreover, he went through Typescript Review and Front End (TS & ReactJS) Checklist as well. His checklist is detailed and have all the necessary checks in it. The

only thing that was missing was having a Brief Description of Defects found section, but otherwise, it is really good.

Code Inspection for File Upload Page component

Reviewer: **Glenn Tanjoh Jong**

Reviewed: **Ralph Gregorio**

Pull Request Number: **30**

Brief Description:

A non-functional page that builds the basic UI of the setup File Upload Page. Renders the UI for the page. Also implemented Jest tests to check the component is rendered properly. There was appropriate use of Jest and Linting ensured consistent styling across pages.

1. Java & Typescript General Review Checklist

1. Variable Declaration
    - Are variable names informative Yes
    - Are variable names unique (not confusing or similar)? Yes
    - Are variable names following chosen capitalization conventions (camel case)? Yes
    - Are variables properly initialized? Yes
    - Are variables labelled as private or public based on their use? N/A
    - Is every declared variable used? Yes
    - Is there excessive use of unnecessary temporary variables? No
2. Methods and method signatures including return and input types
    - Do method names reflect method functionalities? Yes

- Do method expected return values match the intended use of the return value? Yes
- Do methods have safeguards for problematic/unexpected input? No
- Is there a high cohesion between the methods within the same class? N/A

3. Class definitions and grouping into packages (Java)
    - Do object classes reflect the required elements of the program? N/A
    - Are classes placed in the appropriate packages reflecting the nature of their use? N/A
    - Are classes in different packages loosely coupled? N/A
4. Control flow Defects
    - Are Switch cases used instead of if/else blocks when appropriate? N/A
    - Are While loops successfully terminated to avoid infinite loops? N/A
    - Are control flows used efficiently in the handling of erroneous input? N/A
    - Are loop variables declared properly so that their scopes are only as big as necessary? N/A
    - Are there checks for edge cases (out of bounds) for For loops? N/A
    - Are there else blocks used for every if condition to ensure no case goes unhandled? N/A
5. Code style & practices
    - Is code consistently indented, spaced, and formatted? Yes
    - Code is well documented using inline comments and (Answer: No)
    - Are Expensive operations minimized (shallow object copies replacing deep ones if possible)? No
    - Are generics used where possible to improve code readability & reduce complexity? N/A

Typescript Review Checklist

- Are type annotations and inference used? Yes
- Are strict Null checks in place? Yes

Use type inference, type annotation, and generics.

2. Front End (TS & ReactJS)

- Is the single responsibility principle applied to react components? Y
- Are container components used strictly for managing state and business logic? N/A
- Are presentational components used for UI rendering and logic strictly? Y
- Are related components, styles, and assets grouped within same directory? Y
- Are functional components used instead of class components if possible? N/A
- Are React hooks used to manage & control state & effects in functional components? N/A
- Are hooks called at the top level of functional components? N/A
- Are unnecessary re-renders avoided? Y
- Are local component states prioritized for UI-specific state matters? N/A

Code Inspection for React Router and navbar

Reviewer: **Kamal Zrein**

Reviewed: **Cameron Beattie**

Pull Request Number: **37**

Brief Description of Defects found:

Uncertain if it is best use to have the react route components written in both the navbar and the app components. It seems however that this code pattern was used with the intent of separating the protected and public routes at a level lower than the react router inside the app component which could be a plausible use.

1. Java & Typescript General Review Checklist

1. Variable Declaration
    - Are variable names informative Yes
    - Are variable names unique (not confusing or similar)? Yes
    - Are variable names following chosen capitalization conventions (camel case)? Yes
    - Are variables properly initialized? Yes
    - Are variables labelled as private or public based on their use? N/A
    - Is every declared variable used? Yes
    - Is there excessive use of unnecessary temporary variables? No
2. Methods and method signatures including return and input types
    - Do method names reflect method functionalities? N/A
    - Do method expected return values match the intended use of the return value? Yes
    - Do methods have safeguards for problematic/unexpected input? No
    - Is there a high cohesion between the methods within the same class? Yes
3. Class definitions and grouping into packages (Java)
    - Do object classes reflect the required elements of the program? N/A
    - Are classes placed in the appropriate packages reflecting the nature of their use? N/A
    - Are classes in different packages loosely coupled? N/A
4. Control flow Defects
    - Are Switch cases used instead of if/else blocks when appropriate? Yes
    - Are While loops successfully terminated to avoid infinite loops? Yes
    - Are control flows used efficiently in the handling of erroneous input? No

- Are loop variables declared properly so that their scopes are only as big as necessary? N/A
- Are there checks for edge cases (out of bounds) for For loops? No
- Are there else blocks used for every if condition to ensure no case goes unhandled? Yes

5. Code style & practices
    - Is code consistently indented, spaced, and formatted? Yes
    - Is code well documented using inline comments and docstrings? Yes
    - Are Expensive operations minimized (shallow object copies replacing deep ones if possible)? N/A
    - Are generics used where possible to improve code readability & reduce complexity? N/A
6. Typescript Review Checklist
    - Are type annotations and inference used? Yes
    - Are strict Null checks in place? Yes

    Use type inference, type annotation, and generics.

2. Front End (TS & ReactJS)

- Is the single responsibility principle applied to react components? Yes
- Are container components used strictly for managing state and business logic? Yes
- Are presentational components used for UI rendering and logic strictly? Yes
- Are related components, styles, and assets grouped within same directory? Yes
- Are functional components used instead of class components if possible? N/A
- Are React hooks used to manage & control state & effects in functional components? N/A
- Are hooks called at the top level of functional components? N/A

- Are unnecessary re-renders avoided? Yes
- Are local component states prioritized for UI-specific state matters? N/A

Brief conclusion: Cameron’s code follows most of the check marks dictated by the code inspection document. He has also done a good job of documenting and styling his code.