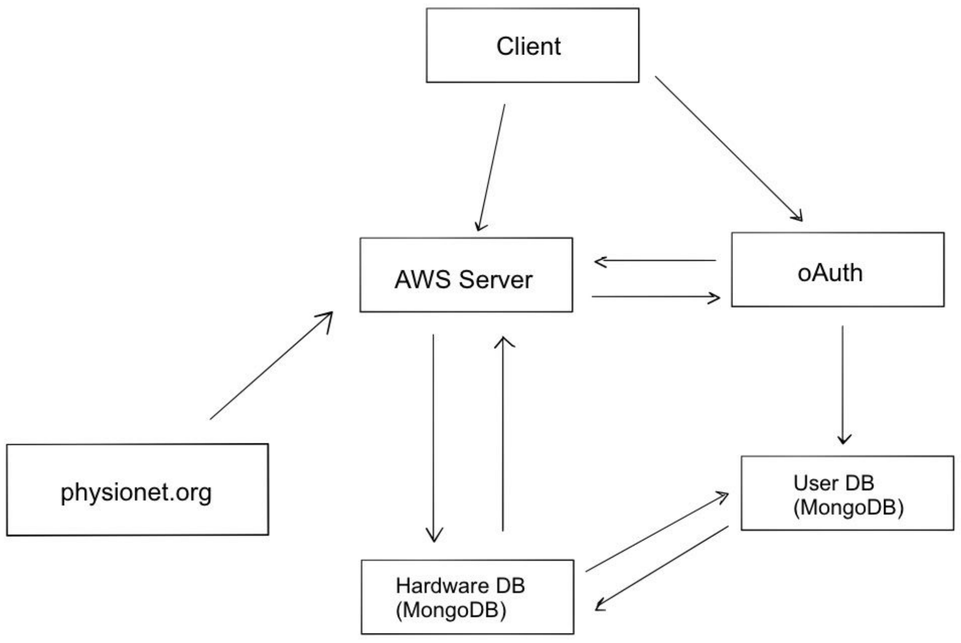
1. Project Summary

In this project, we will create a Web App that allows users to create a “Project” (to be defined later on), and request both hardware and data in order to complete said Project. Both the hardware and data will be abstract at this point, which means that they need to be a catch all for various types of hardware sets & data sets, that can be used to implement more specific types at any given point. Similarly, the “Project” will consist of some type of form, through which the user enters information about the type of project, use case, purpose, name, etc. The project will also be abstract, and can be implemented in a more specific way later on.

The team will use React.js for the client/front-end, Python and PyTest for the server/back-end & functionality, MongoDB for the database on the back-end, and AWS for the web application host/infrastructure.

2. Sketch



3. Tools / Approach

   Plan:

1. The team will begin by defining the features, use case, and overall summary of the Web App (referred to as “app” from here on out).
2. After defining use cases, MVP, etc, the team will begin working on the front-end of the app. This will be done using React. The main page (displayed once a user has already signed in) will be designed first, as it has the most UI elements and requires the most work aesthetically and functionally. Once this is complete, the team can create various side pages, such as the new project form, login page, new user page, settings, etc.
3. After the main page is created, the team can begin concurrently working on both the front-end (side pages), as well as begin establishing and connecting the back-end (Python and MongoDB). This will be done to store user information, keep track of hardware (in total, as well as in relation to each user), track projects, etc. Initially, the back-end database connection will be granted admin user privileges (database queries, access to all data).
4. After the main page is both established and connected to the back-end, the team can also begin developing (both in tandem or as a focused effort, given the backlog and to-do tasks at the time) the full functionality of the back-end API in Python. The API will be used to retrieve data from the back-end database in a secure way, so users are unable to access data that they do not have permission for.
5. As the back-end API is developed, the team will begin writing small tests via PyTest, and then scale those up to ensure that the API works as expected.
6. Finally, the team will polish and ensure the app looks professionally designed, and has all the necessary functionality. Once the app passes all tests and is up to standards on the local development server, the team will push to AWS. This will require additional tweaks to the software to ensure that all requests still work, as well as the UI components (ie. routing, CORS, UI screen size, etc).
7. The team will publish the site so anyone with the URL will be able to access it.

4. Features Summaries

   Features:
   Users will be able to login, sign out, create and delete accounts
   Users will be able to edit account info (name, password, bio, etc)
   Users will be able to create new projects, edit project info, and delete old projects
   Users will be able to check out and check in hardware sets
   Users will not be able to check out unavailable hardware sets.

1. As a user I need to be able to create an account with a secure login that can only be accessed with a specific username/password combination.
2. As a user, I need to be able to check out available hardware that I can use for a specific project.

3. As an admin for the app, I need the database to keep track of where the hardware is at all times (available or checked out).
4. As a user who has checked out hardware, I need to have access to datasets concerning my hardware.
5. As a user, I need to be able to create new projects so that I can checkout hardware and request data pertaining to it.
6. As a user, I need to be able to view my current projects so I can keep track of what I'm working on.
7. As a user, I need to be able to delete old projects so that I know I have completed them and they are no longer cluttering my workspace / taking up storage space.
8. As a user, I need to be able to check in Hardware sets after I have completed using them, so others can use them as well.
9. As an admin I need to notify users if hardware is unavailable so that multiple users are not requesting / using the same hardware set.
10. As an account holder I need to be able to easily locate site features (checking in/checking out hardware, viewing my projects, etc.).
11. As an admin, I need to be able to check how many users are currently using hardware sets, and for what.
12. As a user I need to securely log-out of my account in the case of using shared devices.
13. As a user, I should receive an error message with relevant information when I am unable to check out datasets or hardware sets.
14. As a user, I need to be able to access my user settings (username, password, bio, etc), in order to edit or change them.
15. As a user, I need to have the option to change my password for security/privacy concerns.

5. How to use Git
    a. Team members will meet at the beginning of each week (Monday) to discuss what tasks need to be completed. The members will meet again on Thursdays to provide updates on progress and receive feedback from other members. The duration of each meeting is set to be approximately one hour long but is flexible and can be longer to meet the task deadlines each week.
    b. Tasks and progress updates will be updated on the github project board so each member has real-time access to the overall progress of the project.
    c. The team will work on individual features of each aspect of the project (enumerated in the project plan) and upload contributions to git.
    d. Each member can update the project board with any immediate concerns so the whole team can address problems in a timely manner.
    e. Members are free to push/pull content from the repo but must talk to other members before making changes to the others' work. Since there's no one super user to authorize pull requests and merges, the team should decide together when to push, pull and merge.