Meeting notes 04/12/2021:

- What the presentation is about.
- TODO: Do part 1 of assignment 3.
- We know dev/local setup/test is working.
- TODO: Check if Heroku is working? Yes.
- TODO: Upload brief meeting notes @Henry
- TODO: Create Assignment3.md with postmortem and tell them where to find it in readme.md. @Daniel
- TODO: Add the overview and readme stuff to github. @Henry
- Assigned slides to everyone
    - Henry: First 2 slides.
    - Daniel: Next 2
    - James: Next 2
    - Stew: Next 2
- Will meet on Tuesday to ensure slides are coherent and rehearsed.

A final overview of the features your project has implemented, including what purpose they serve.

- An elaboration of what these features do and how they work.

Features:

Note: Details of how to use the endpoints and examples are described in our readme.md:

- We can read a list of guidelines.
    - A GET express endpoint which will return all specified records in JSON format. Used to get all guideline ids in the db.
    - AC: I should be able to use the /<resource_type> endpoint to retrieve a list of all definition ids. Where <resource_type> is ActivityDefinition or PlanDefinition.
- We retrieve a specific guideline.
    - A get express endpoint which will return a specific record given an id in JSON format. Used to retrieve the specific information about a guideline such as it's description or version.
    - AC: I should be able to use the get /<resourceType>/<id> endpoint to retrieve the specific guideline of <id>
- We can add a new guideline.
    - A POST express endpoint to add a new guideline to the db. Used to insert a new guideline into the db.
    - AC: I should be able to use the post /<resourceType> to insert a new guideline record into the db
- We can update a guideline.
    - A PUT express endpoint to update the json of a specified guideline. Used to update the fields of the guideline.
    - AC: I should be able to use the /resourceType>/<id> to update the specific guideline of <id>
- We can delete a guideline.

    - A DELETE express endpoint to delete a specified guideline. Ensures a specific guideline can no longer be accessed, but won't be permanently removed in the database.
    - This is a soft-delete, and is not explicitly removed from the DB.
    - AC: I should be able to use the /resourceType>/<id> endpoint to delete the specific guideline of <id>
- Integration tests that tests the above.
    - Should verify below for both activity and plan definition:
        - adding a new guideline to DB
        - reading a list of all available guidelines
        - updating guideline's json
        - retrieve the guideline and verify contents
        - delete the guideline.
    - Responsible for ensuring the various features work as a cohesive whole.
    - *AC: The test case does exactly as we specified above. We should attempt 2-3 "atypical" edge cases similar to the respective unit tests.*
- We have a readme in our github repo, which documents what our project is about, what the various API calls do, how to use them and an example of each.
    - *AC: I can access the readme, which clearly states the purpose of the project and documents all API calls, how to use them and has an example.*

    - Create db/schemas and populate our downloaded guideline definitions (James)
        - Download a zip file of guidelines in json format from https://build.fhir.org/ig/HL7/cqf-recommendations/definitions.json.zip
        - Create a new database called "guidelinesdb"
        - Creates schemas
        - Create two collections for "plandefinitions" and "activitydefinitions"
        - Insert all of the plan/activity definitions from the zip file into the db
        - Verify that the db works by reading a few of the elements.
        - *AC: I can write a program that can add data to the plan or activity definition collection or read any definition data from the 'guidelinesdb' database.*
    - We should additionally verify the above functionality using manual testing with Postman (Henry/Daniel/Stew)
        - *AC: Using postman, I can try requesting all of the endpoints (with several different varying inputs) using manual testing and receive the expected response.*
    - Our test coverage at the end is >= 75% (Daniel/James, End of A3)
        - Run jest check the coverage.
            - If it's less than 75%, then write more unit tests until it reaches the threshold.
        - *AC: I can run jest with the --coverage flag and the result is >= 75% code coverage.*

- The acceptance criteria for these features and how that acceptance has been demonstrated. In particular, what verification criteria have been negotiated with

your industry partner, and what tests, processes, or documentation exist showing that those criteria have been met?

How have the acceptance criteria been met?

- The acceptance criterias for the endpoints have been met, because we have created the various endpoints specified in app.js and ensured that they work through manual testing with Postman. In addition, we have ensured that the functionality is correct through our unit testing and integration testing.
- The acceptance criteria for the unit tests and acceptance have been met, because we have created a typical test case which performs the specified action and verified that it works (ex. By creating a mock database and ensuring that the correct guidelines have been listed). Each test also has at least 2 “atypical” edge cases, such as an empty resource type or missing guideline id or adding duplicate guidelines. These tests can be found under api/tests.
- The acceptance criteria for the database has been met, because we have written the endpoints and unit tests to correctly read/write data from the database.
- The acceptance criteria for 75% code coverage has been met, because our docker test runs the coverage report which specifies > 75% code coverage for all files.
- The acceptance criteria for postman has been met, because we have already completed manual testing using a variety of typical and edge cases and found our code to work as intended.
- The acceptance criteria for the readme has been met, because we have updated our readme with the specified information.

How we negotiated with the validation criteria with our partner:

- We proposed the unit and integration testing features to the client and reviewed the acceptance criteria with them.
- In particular, we agreed that we will write unit tests for each endpoint and integration tests that include functionality from each endpoint.
- The client commented that our “several edge cases” criteria wasn’t specific enough and wanted us to change it to a concrete number. It was changed to “2 - 3 edge cases.”
- After this change, the client approved our unit/integration testing acceptance criteria and determined that “success” would be determined by completing all acceptance criterias.

To show the acceptance criterias have been met:

- We have the unit and integration tests stored in api/tests.
- Each test will test typical cases where the functionality works as required, such as making sure all guidelines are listed using the “list definitions endpoint” using a mock database.
- Each test also includes 2 - 3 atypical test cases, such as an empty resource type or missing guideline id or adding duplicate guidelines.
- Our conversation with our client has been documented in our meeting notes.

- What features of tasks were not delivered, if any, and documentation of how those decisions were made.

1. Stretch goal: We should be able to get the activity definition referenced by a plan definition (Whoever finishes early, No due date)
    - Create a router endpoint to retrieve an activity definition referenced by a plan definition.
    - This data can already be retrieved by getting the json data of the specified definition, which is why it's a stretch goal.
        - *AC: I should be able to use the <endpoint> to retrieve an activity definition for a specified plan definition.*
    - Create a unit test to verify its functionality.
        - *AC: I should be able to run the tests using docker, which runs all of our unit tests. The unit tests should include at least one test case for each "typical case" and about 2- 3 "edge cases" where appropriate. The group should review and accept all TCs at the end.*

Why wasn't it delivered?

- The feature wasn't a core feature of our product, as we already provide this functionality using the "get specific guideline" endpoint.
- Our team members were busy with other courses and couldn't allocate the time to finish this endpoint.
- We only planned this feature if "someone finishes early."

Readme.md

- What are your validation, verification, and acceptance criteria as agreed to with your industry partners, and where is your understanding of those criteria demonstrated in the code.

These points are discussed in Assignment3.md under "Features" and "How have the acceptance criteria been met?"