Meeting Notes:

* Databases:
  + Relationship b/w protos and mongo
  + Schema not important -> pretend back-end in NOSQL schema, fields can be added and removed without issues
  + Current Schemas:
    - Users
      * UUID
      * Name
      * Goals: [ array of goals ]
        + Goal: See the protos for details
      * Everything else - Personal profile information
        + Birthday
        + Gender
        + Etc.
    - Meals - Each one is all 3 meals at a given point and at a given cafe
      * UUID
      * Food items
        + Schema is determined by the scraping team -> trim it down (Ishan wrote code for this)
        + Which meal
      * Cafe ID
    - Cafes / Dining halls
      * Name
      * UUID
    - Food Logs - Each entry is a meal consumed by specific user
      * UUID
      * Timestamp
      * User ID
      * Cafe ID
      * B/L/D
      * Meta data (freeform)
        + Ton of info in here
  + Recommendation: Represent tables using MongoDB
    - Ex: To get user meal information for a day, query food logs on User ID
  + Recommendation: Download Mongo shell, and see tutorial on documentation
    - Sufficient for what we need to do
  + Heroku plug-in (M-lab(?)) -> if it still works, useful
    - Can be used for deployment
    - Heroku plugin that gives you a mongodb production (really sandbox) database
    - For testing during development, use a local database on your own machine
* Process of setting everything up:
  + DL Mongo
  + Cmd: mongo d -> starts mongo db
    - Aside: Hierarchy
      * DB -> has multiple collections (analog of table)
      * Each collection holds documents (analog of rows)
    - Ex: Cafe Document
      * Cafe document:
        + { ABCD1 [this is the UUID],

Name: “CJL” }

* + - * + Valid JSON
  + Inserting into mongo db
    - Must use a driver for applicaiton (pymongo)
    - Construct python dictionary with the same shape as the document and then call the insert function
      * Must enforce the shape of the document at applcition level
      * Aside: Can open a mongo shell and throw shit in from there for testing
* Protos just act as ground truth for database structure -> should use them as a reference for how to create our mongodb db
  + Meals and Food Logs will probably be contained in JSON files
* 2 decisions to make in terms of storing the two weird fields (metadata in Food Log and the arrays in Meals)
* Processes: Client, Server, DB
  + Hard part: establish communication between the 3
  + General flow:
    - Server and DB comms. Done thru driver (pymongo)
* Note: In npm start command in package.json (?), ask it to build
* GET END TO END CONNECTIOn
  + Use create react app to create react app
  + Set up the Flask server
  + Set up communication by creating dummy endpoint and then something to grab that dummy endpoint in React