Agenda:
- Feedback on Design Doc, Teamwork Plan, and Project Pitch (if any)
- Understand expectations for the whole project
- Discuss DB model (Application, Common, Custom, User, Student, Employer, Listing)
- Advice on development cycle
- Implementation Questions
  - We have an array of embedded documents and what we want to do is with one query, change a field in every single one of the embedded documents?
  - Is it bad practice to have a database object instantiated in our code that is instantiated every time the server restarts?
  - Is it better to create an empty common template and update or create an already filled common?
  - Escaping text for code injection
  - Go over routes

Progress Report:
- Started working on models and tests for DB
- Organized routes
- Milestones
  - Achieved: models and tests for Listing, User, Employer
  - Missed/In Progress: models and tests for Application, Common, Custom, Student
- Difficulties encountered:
  - Implementation Difficulties
    - Querying and updating question answer templates
    - Escaping text for code injection
  - Design
    - Application, Common, and Custom databases
    - Form translations
- Changes: N/A so far

Meeting Minutes:
- Feedback:
  - Purpose: well written
  - Concepts: well written
  - Data Model:
    - When an application is submitted = it's starred. But starred application can be withdrawn/rejected
    - Differentiate between student and employer status
    - Clarify that common app is only submitted once at the beginning
    - Common and Custom together make up the Application, which holds a status
  - Wire Frame
    - Make more modular parts of the app (box around)

- - ○ Challenges: well written
  - Understanding Expectations
    - ○ MVP needs to fulfill purposes of the beginning/solve the problem given the solution
    - ○ Final: semi polished app/contained within itself
      - ■ Cutting features if needed
  - Discuss DB Model
    - ○ Update so that Student holds commonCompleted boolean
  - Implementation Questions
    - ○ Write a Question Model???
    - ○ Can Update everything??? Async.parallel/Async.each/Async.map
      - ■ One callback for everything that's finished that updates everything
    - ○ Handlebars/templates already escape text by default
  - Routes
    - ○ GET students/application or employer/application
      - ■ Separate into student and employer

Decisions:
- Use a separate model for Question and check out async calls
- Separate routes into student and employer
- A student can sign up without filling out the Common App, but cannot access the rest of the student part of the web app without having filled out the Common App