Agenda:

* ~~Goal by today was to get everyone set up; let’s begin development~~
* ~~Went through WFs with Dr. Cook on Tuesday - go over/provide notes and follow up with Julio today~~
* ~~Task/feature priorities~~
* ~~Go through & get clarification about the relevant portions of code~~

Notes:

* Instructor accounts: have to be generated with a code, but problem with backend API
  + contacted source who worked on it before, may need to contact different person
  + Julio gave us access to his account in the meantime
* Project code base runthrough:
  + error in dev tools console - trouble pulling user\_id - presumption is problem with live deployment database but shouldn't affect our development
  + to add data tab: uncomment /frontend/src/navigation/AdminNavigation.js and uncomment #39 and save
  + See PeerPresents/docs/
  + (Temporarily) window jitter: uncheck styles -> minimum height
  + Datasets page is fully functional
    - no way to delete dataset - would have to manually delete from DB
  + Add graph button should generate another graph (same set of dropdowns)
    - maybe new datasets should have 0 graphs (but for development, it currently shows one)
    - in theory: save graphs to each dataset (so next time you open it, same graphs are still there)
  + Datasets consist of sessions from [a course(s)]
    - on create data set screen, format is: CourseName (SessionName) +
    - session doesn't need to be from a particular course
  + Courses have [multiple] sessions made by students, sessions have multiple presentations made by teams of students; session tallies total number of questions from all of the presentations
    - a session == day of class (there would be a presentation day with multiple teams presenting)
  + Graphs:
    - Columns (for # of questions): each col's y-value is total number of questions, and each col is a session (can be multiple sessions)
      * takeaway: maybe this gives information on overall class participation for any given session (~'presentation day')
    - feed it data in the form of dictionaries with x, y values
    - VictoryGraphs (actual graph)
      * See line 142 - '(Object.values...' this is the line that actually generates the graph by mapping it like the existing graphData variable
      * .map(dataYouWantToGraph, indexIsOrderOfValues) to parameter data={[eachIndex, actualDataIntoASpecificSpotInGraph]}
      * aligns to middle, pushes leftward as new values are added (style lower prio)
  + GraphsContainer.js
    - main purpose is getting data from DB - probably do db calls here to get data the graphs should contain
    - from **GraphsContainer** pass to **Graph** in the render(return(...<Graph />)); component tag with props as attributes
      * **graphList** number of graphs to display (not exactly, but graphList.length would be actual number)
      * **sessionList** is initially empty but when you generate GraphsContainer you want to pull this from the db
      * **sessionQuestionsDict** used to pull tallies/total - order is important (idea is specifying key = index : value)
      * questionList may be obsolete, unused
  + DB calls
    - **getDataSet(**dataset\_id, thisDatasetDoesExistBoolean**)** use id - every dataset has a sessionList
    - **getSession(**session\_id, thisSessionDoesExistBoolean**)** use id - every session also has its own id in db and each has presentations
      * important to get title attribute for a session
    - **getPresentation(**resentation\_id, thisPresentationDoesExistBoolean**)**
  + Can refer to /storage/app/schema/data\_models.js for more information
  + setState({ nameOfStateAttributeToUpdate: valueOrReferenceToUpdateItAs });
    - note that just about everything in React is immutable, so you're not actually updating it - be sure to explicitly (copy the old values + something new you want to add) or else it'll just replace this state with (something new you want to add)
    - can do this manually with a separate array
    - can do this through hook and spread syntax (...oldThing, thingToAppend)
    - "states are universal" - scope & closures?
  + Recommend: read up on **states** and **props** - covered example usage
  + In Graph.js, graphData = [ **[ {},{},{} ]**, ..., [{},{},{}] ] where each of the bolded 2nd dimensional arrays is a session, each of the js objects in {} contains two dictionary references to x and y components
    - First level is the array[] of sessions
    - Second level is the array[] of an individual session's presentation counts (i think)
    - Third level are the js objects{} (basically, dictionaries) defining { x: integer, y: integer }
  + Components - roughly templates, may be used on any screen - we'll be working with:
    - Graph.js
    - GraphsContainer.js
  + Screens we'll be working with:
    - DataTab.js
    - CreateDatasetScreen.js
    - DatasetScreen.js
  + Hover feature not currently implemented at all
  + Window size, CSS for graphs *could* be improved if there's time, but prioritize functionality
    - the way x and y values are provided to the <VictoryBar> component can cause multiple columns to be pushed to the left, possibly related to using dictionary usage because dictionary is converted to array through map()
  + Worry about Export Data later - will need to talk to Dr. Cook about this; no WFs exist for this
* Each person could work on:
  + dropdown menu working (in general) for one choice, then figure out how to graph that choice
    - that person could then move on to displaying emoji reactions
    - at the same time another person could work on displaying a different type of data
  + coding together is potentially good approach

Action Items:

* Meet again next Tuesday at 4:05pm as scheduled
* Recommend look up how states, props work in React
* Should get Julio's instructor account login information in Discord soon
* Pass these notes to Andre