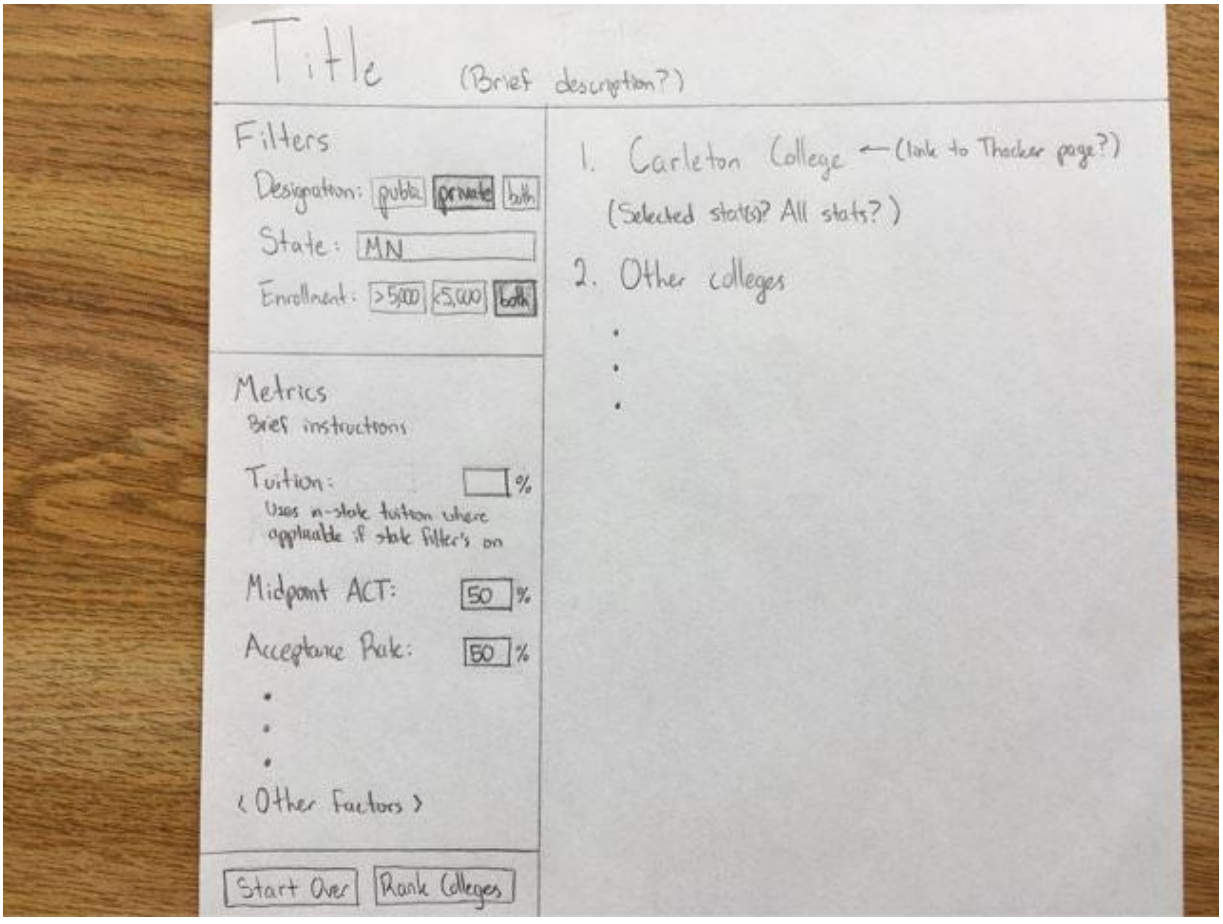
Kevin Christianson and Isaac Haseley

Today's college ranking systems garner nothing but complaints for their misguided use of metrics by which to evaluate institutions. Why ask administrators to judge universities they've never stepped foot on? Why weight standardized test scores at 7.5% of an institution's outcome? And where's the quality of food metric? We can't promise to fix that last one (yet), but our app will resolve common complaints by placing ranking metrics in the hands of its users. We'll start humble and use the factors from our API (see mockup below).

The MVC design pattern is the pattern for us. Modeling Colleges and a data structure to manage them (AllColleges) will allow us to reuse these classes upon future expansion of the user interface. Separating the design thusly will allow Kevin and me to work on different pieces of the project and reduce merge conflicts. When we begin reading resumes and our team expands, this conventional design pattern will be familiar to new recruits. Views will include nested AnchorPanes (or Regions or a close relative), buttons, text input fields, and text labels.

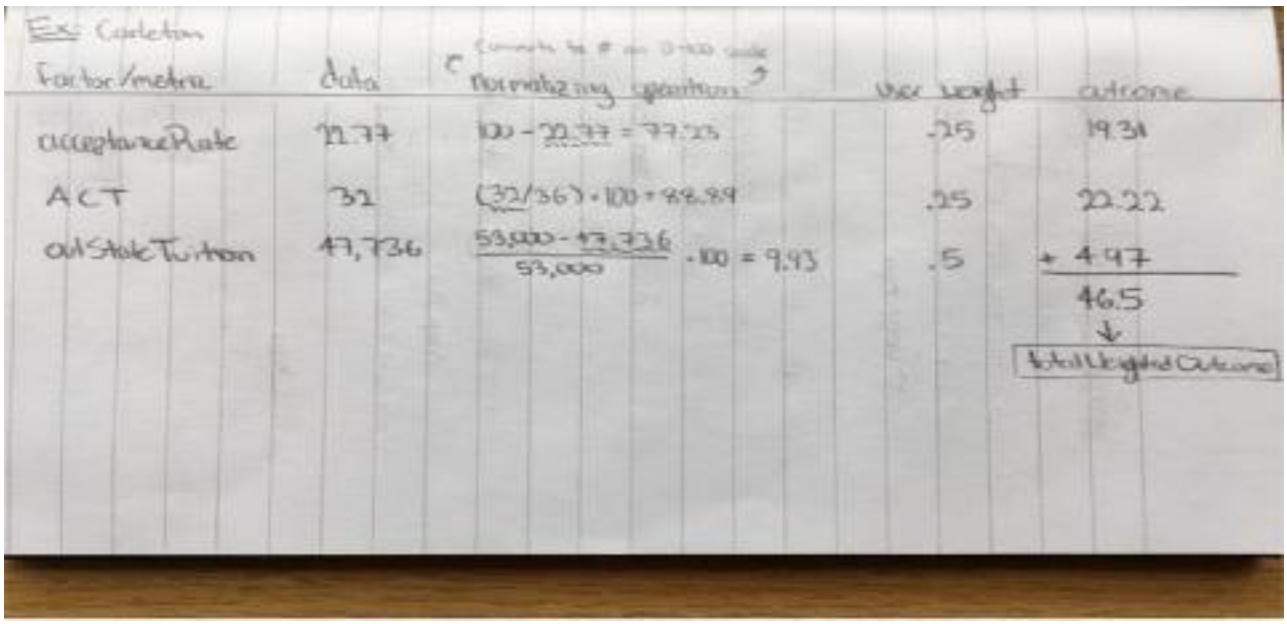
Potential mockup:



Models

- College
  - Instance variables for all attributes – We'll grab this data from our API.
  - double totalWeightedOutcome – See example below. We'll use this to sort.
  - Constructor will take in a dictionary, sets all instance variables except totalWeightedOutcome
  - Getters for all instance variables
- AllColleges:
  - collegeList – an ArrayList of Colleges
  - Constructor will hit API, construct each College from its dictionary, and add it to the list
  - rankByUserMetrics – ranks (sorts) CollegeList using user's weights
  - Getter for collegeList

Ranking algorithm: the following example assigns Carleton its totalWeightedOutcome metric, which we'll use to sort all colleges in the user's ranking list.



GSON installation guide:
GSON is a module for java created by Google, which parses JSON. We use it in our project as the results we get from our API are in JSON. All the files necessary are in our project, however to configure them with IntelliJ there are a few steps necessary:

1. Go to File->Project Structure (command ;)
2. Click the plus sign on top left to create a new project library.

3. Select Java as the source and then navigate to the lib folder inside the final folder.
4. Select all three files (they should all be .jar) and hit enter and then apply.