Project Update #2

1. Project title

College Admissions Calculator

Github Repository

2. Team members and roles

Nnamdi Ede, Project Manager, nnamdi.ede2019@gmail.com

Ryan Essem, Tester, nolimitry1@gmail.com

Jim Chen, Analyst, jchen160@umd.edu

Amanda Hernandez, Researcher, avhernan@terpmail.umd.edu

3. Accomplishments since last report

What you have done since last report – bullet list of individual contributions; note if any tasks were not completed and why

- Amanda wrote code that returns whether or not a student will be accepted or rejected based on if they input a GPA and SAT score that is greater than or equal to several parameters.
    - Uses UMD specific data (average GPA and SAT score) from students admitted this year.
    - As a group, we also decided to change our original project idea (more on this in the next section).
    - Updated flowchart and readme to reflect new changes.
- Ryan also experimented with code and figured out how to color code a scatterplot.
    - Also found additional data we can use for GPAs.
- Delegated sections of code to each person
    - Amanda → Accepted students
    - Nnamdi → Rejected students
    - Ryan → Waitlisted students
    - Jim → Deferred students
        - Created smaller text files to use as a practice runs. We plan to expand the text file to about 1,000 students with varying GPAs and SAT scores.
- Spoke to Rankin about our overall idea and new developments/changes from our original plan.

4. Roadblocks, problems, challenges, risks, questions

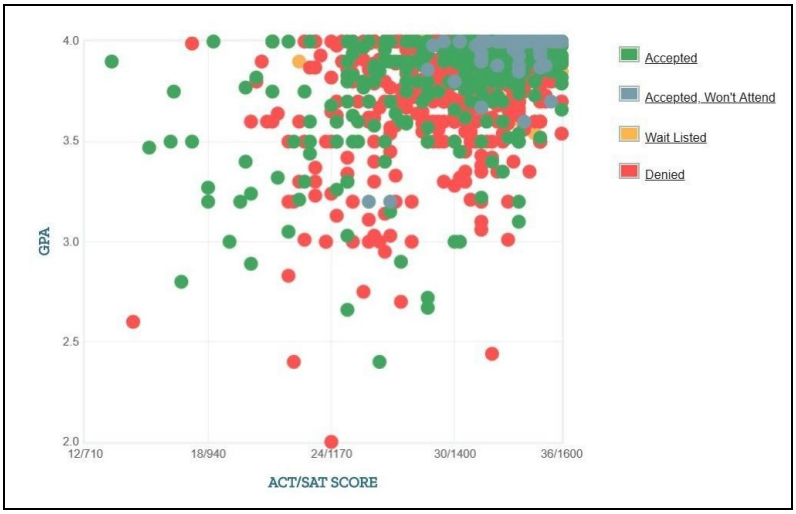
Bullet list of any concerns at this point – current or upcoming

- After our last update, we decided to change how the code/overall project would be structured as well as what information it would return to the user. Initially, we wanted to create a program that served as a college admissions calculator in the sense that it would calculate a user's chances of getting into five different schools based on only their GPA and SAT score.
    - Now, we have decided to deviate from the original idea and create a program that takes the user's input (GPA, SAT score) and returns whether they would be accepted, rejected,

waitlisted, or deferred at the University of Maryland based on Maryland's average GPA and SAT score of admitted students earlier this year.

- The program will also read a text file that has the names, GPAs, SAT scores, and number of activities involved in of about 1,000 randomized students and determine whether those students will be accepted, rejected, waitlisted, or deferred by comparing their information to the paramaters defined above.
  - Accepted decision range:
    - GPA >= 3.0
    - SAT >= 1250
  - Rejected decision range:
    - GPA <= 2.7
    - SAT <= 1100
    - Activities < 4
  - Waitlisted decision range:
    - GPA in range (2.7, 3.0)
    - SAT in range (1100, 1250)
    - Activities == 4
  - Deferred decision range:
    - GPA in range (2.7, 3.0)
    - SAT in range (1100, 1250)
    - Activities >= 5
- The program will then return four text files (one for each admission decision) with the names of students and their GPA, SAT, and the number of activities they're involved in. The idea behind this is that it will help the user understand how they compare against other students. This gives it a bit of a more realistic element as well.
- Lastly, the program will also return a scatterplot with the data in the returned text files. This will also give the user an idea of the average GPAs and SATs that are accepted, rejected, waitlisted, and deferred. The x-axis of the scatterplot will be SAT scores while the y-axis will be GPAs. The plot will also be color coded so each admissions decision is represented by an easily identifiable color.

Example of what the scatterplot would look like:



- How complex does the code need to be? (Do we need to have a certain amount of functions or other elements?)

5. Plan for the next sprint (i.e, until the next deliverable)

Your plan for the next sprint – bullet list of individual tasks, with tentative deadlines and assigned roles

- Over the next week or so, we will work on our designated sections of code and put together the larger text file of random students. As we are working on our individual code, we will also run test runs to ensure that our section of code works.
    - We will meet during this weekend to talk through the code we have written individually and figure out the most efficient way to put each piece together.
        - After this, we will also determine what extra credit parts we want to include.
    - Once we have the master code/program put together, we will make all the necessary updates to our repository, flowchart, and readme.