Legacy Code Improvement

Introduction and Context:

The client needs a more efficient way to assign TAs and graders each semester. They need an easier application process for TA and grader positions, plus a way for faculty to view applicants and make recommendations. Applicants should be able to accept or decline offers, and the client should be able to export data from our application to a CSV or similar file for HR. These are the core functionalities listed, our main objectives are to improve and implement these to detail and implement a user friendly UI.

Existing Code Base:

This project does not have a pre-developed APP, but rather some existing python code. These codes implement simple logic using only Numpy, Scipy and Pandas libraries from Python,

The code will input and parse several CSV files that contain name, ID, and their application to their desired positions. These files can contain student, professor, and course information. The algorithm provided to us will parse these parameters and return a file with student name, UIN, their assigned section and a calculated score to measure each student's compatibility.

Strategy for Learning:

Addition to learning Ruby, Rails and other tools that help us develop and deploy our APP, we will also need to learn the logic implemented by the currently provided code.

Our sponsor requires us to add additional functionality to existing code. In Addition to being able to assign TAs from a CSV sheet, TA positions need to be adaptive to changes such as people leaving due to sickness or job opportunities. This requires us to understand the current logic of the algorithm, and add function to add, remove, and change TA position on top of the assignment function. Another functionality we have to add is that all PhD students have to be hired. Since the current system assigns a score to every student, some master students can get assigned a higher score than a PhD student which is against the sponsor;'s desired functionality.

We plan to improve upon the previous legacy implementation by first solving the issues our client had with the legacy project. In order to do that we first need to fully understand the python script given to us.

Strategy for Improvement:

The main issue with the previous implementation was that the algorithm would need to be rerun with every change. For example if there are 20 vacancies after the first round of offers went out then those vacancies will need to be filled. However the algorithm will simply run and reassign Graders and TAs to the classes again without taking into account already filled positions. We will improve upon this function through python first since we can invoke the python files through a python API. Then we will try to migrate these to Ruby if conditions allow. The second user story mentioned regarding prioritizing PhD students can be purely done in ruby as we will parse the file again to select the PhD student with the highest score. The seventh

user story mentioned wanting to export all the information to HR with one click rather than manually needing to send each individual's information to HR which we can do by adding a functionality to the code.