Team name: KPMGComeGetUs

Name: Sng Hong Yao **Student ID:** 17205050
Name: Ankish Raj Prajapati **Student ID:** 17202456
Name: Ronan Mascarenhas **Student ID:** 17379773

Setting up Environment

1. Open a terminal and navigate to the file where run.sh is.
2. Execute `./run.sh` (Uses maven to build the JAR, then Java to run the JAR)

Meeting Demands

Prerequisite

- On application startup, the first thing you will see is the Settings tab within the Setup page.
- Within that tab, you can
  - specify the number of students the random candidateSolution should generate,
  - toggle hard/soft constraints the fitness evaluation function uses,
  - toggle GPA consideration which will affect the fitness evaluation function, and CSVReader/Writer, and
  - toggle other application settings.
- Once you are done in the Settings tab, you can either:
  - 1. populate projects then students, or just
  - 2. populate students. (i.e. populating projects is now optional).
- At this point,
  - the solvers page on the left is disabled until students are loaded/generated (this is because you cannot do any computation until students exist).
  - This design is used to subtly guide users to use the application.

Load additional GPA information on students into the system

- If the ‘GPA’ column header is present in the first line of the CSV file, we will load it accordingly.
- Else, the gpa field for each student will be set to null.
- Essentially, GPA information is made optional, and they will be evaluated accordingly, i.e. if GPA is present, the soft constraint for GPA will be taken into consideration during fitness evaluation, else we simply ignore the evaluation of that soft constraint altogether.

Indicate the importance given to GPA via a slider in the GUI

- Within the Settings tab in the Setup page, under *‘GPA Consideration’* section, there is a slider for this purpose.
- Setting to 1.0 means fully enabling GPA consideration, and 0.0 means disabling GPA consideration altogether.

- The value from this slider is multiplied with our GPA fitness for each student. e.g. try changing it, the starting fitness/energy varies accordingly.

Visualize the quality of the current solution in the GUI (via a colour bar, a meaningful number, a dial, or so on)

- Within either of the solvers page, you will be presented with:
  - a graph,
  - processing thread's progress indicator,
  - control buttons,
  - the current/best solution with strength indicator, constraint violations, and their table.
- If you click 'Play' in the control buttons, the solver will use the specified parameters to try to obtain the solutions.
- If animation is:
  - enabled (slower), the graph, processing thread's progress indicator, solution strength indicator, constraint violations and table will be updated constantly.
  - disabled (faster), only the processing thread's progress indicator will be updated. Once computation is done, only then the graph and solutions' strength, constraints, and table will be updated.
- Once computation is done, solutions to the allocation problem will be populated in the table/sheet below.

See the progress of the current search for solutions on screen (via a progress bar, or some other dynamic GUI element)

- There is a progress bar, represented by the *'Processing thread progress'* progress bar below the graph, and above the control buttons in the solvers page.

Extras

- **Sample Input file by Tony.**
  - Tony gave us an Excel file, we exported it to CSV.
  - Unfortunately, all of us exported CSV files with similar defects, so we had to clean up the exported CSV file.
  - Defects include:
    - Extra columns. (each line shows extra commas at the end)
    - Extra rows. (last few lines have empty values, with just commas)
    - Similar student IDs.
  - These defects are fixed in the CSV file called, *SampleTestNoExtra_(unique ids)*.

- **Handling errors when loading students file.**
  - Error messages are now made even more verbose.
  - It now shows the line number that caused the error/exception, and the speculated problem that caused it.

- **How we can parse our original CSV file format and Tony's CSV file format.**
    - Previously we had a sophisticated way to guess which column represents what.
    - We now judge what the column represents based on the column headers.
    - i.e. Our application uses 'First Name' in our CSV headers, Tony's uses 'Student'. So when we parse the first line, i.e. the column headers, we check if they equals to 'first name' or 'student'. If so, then that entire column represents First Name in our system.
    - The same applies for:
        - 'Student Number' and 'ID'
        - '1' and 'Preference 1'
    - Essentially, our CSV reader relies heavily on the first line, i.e. column headers, to decide which column represents what. A small price to pay for modularity.

- **Clear button in Table/Sheet when Loading/Generating students/projects.**
    - During our weekly call with Tony, he mentioned the application should be flexible enough to take in only the students file.
    - This flexibility messes with our previous design choice, i.e. MUST load/generate projects first, then load/generate students.
    - Now the application is able to:
        1. load/generate projects first, then load/generate students, or
        2. load/generate students only.
    - The clear button is introduced to switch between the above two pathways.

- **About section.**
    - Mainly represents an in-application introduction to the application.

- **Other settings.**
    - Toggling animations.
    - Toggling dark theme.
        - We noticed many modern applications have this option, we figured it would be good for the eyes as well.
    - Toggling detailed GA visuals.
        - By default, GA's graph shows all populations' fitness per generation.
        - So as we increase the population size, the more sluggish the graph gets.
        - This setting is added so the user can toggle between the above, or just show the fittest solution in a generation.