CSM Walton

6/11/2020

Stephen Thoemmes
Eric Schmidt
Cole Callihan
Jean Duong

Introduction:

Our assigned project was to develop a web app for automatic student group formation intended for use by professors. Our client, Dr. Walton is a geology and geological engineering professor at Colorado School of Mines and requested an upgrade from his self coded group former algorithm. Dr. Walton's group creator code relied on randomizing students in groups until set criteria was met. This method invoked a long processing time and required unique adjustments of the source code for each new set of student attributes. Because of this, he added his project to the field session project list. Dr. Walton requested for a product that would intelligently group students based on any input characteristics, would have an efficient algorithm, and would allow for easy interaction with a user. These were the three most important factors going into this project.

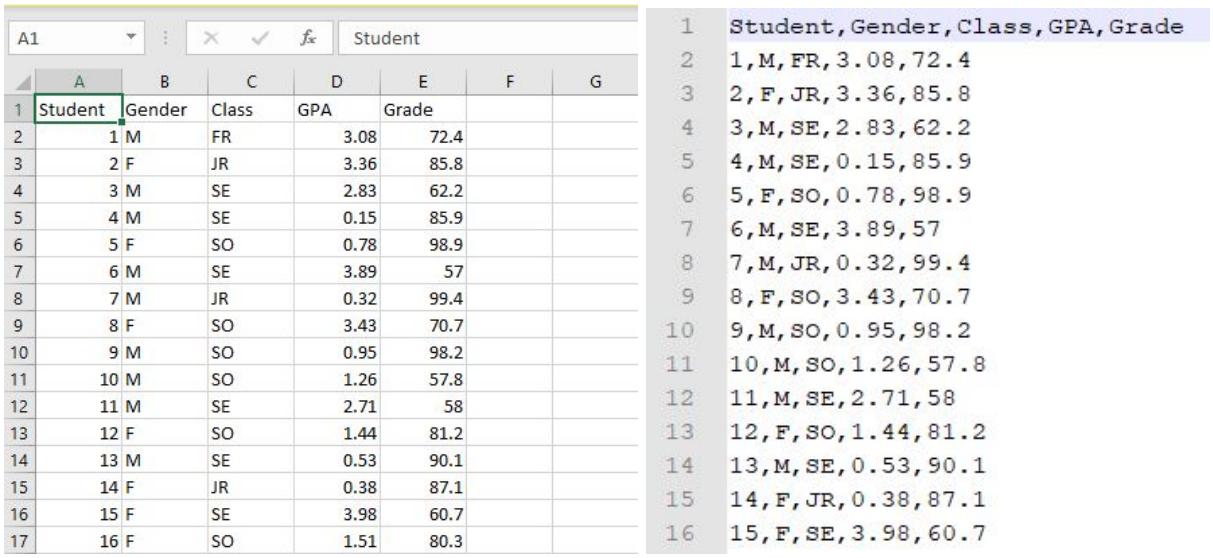
Requirements:

Product Vision:

- Our client requested a program that accepts data on students and uses it to form groups of students based on certain criteria defined by the user. While this input has been manual in the past, our client wanted the ability to upload an Excel file (.xlxs) of the listed students and some of their attributes (e.g Gender, class, test grade, GPA, ect). Next, have groups formed based off of group sizes and user set criteria. Our client also expressed interest in the accessibility of the program, making it execute quickly while also being easy to use. The last significant part of the product is weighing the different attributes such that the groups are formed based on the criteria described by the user. For example, if the user has a class of 39 students with a max group size of 4 and the average GPA of each group to be around 3.0, our program will try to satisfy these requirements to the best of its ability.

Functional Requirements:

- Given an Excel or CSV file in the standard (accessible/easy-to-learn) format specified, our project yields an output of student groups. **Figure 1** below shows an example of an acceptable input file. The accepted format of an input file is one whose first column contains a header followed by unique student identifiers (Names or Id numbers) as the first column; additional columns of data each has a descriptive header, and the data in each column may be numerical or categorical. These group assignments are optimized such that each group is internally diverse but externally similar: e.g the male:female ratio of each group matches that of the class as a whole, or the average GPA of each group matches that of the class as a whole within bounds.



	A	B	C	D	E
1	Student	Gender	Class	GPA	Grade
2	1	M	FR	3.08	72.4
3	2	F	JR	3.36	85.8
4	3	M	SE	2.83	62.2
5	4	M	SE	0.15	85.9
6	5	F	SO	0.78	98.9
7	6	M	SE	3.89	57
8	7	M	JR	0.32	99.4
9	8	F	SO	3.43	70.7
10	9	M	SO	0.95	98.2
11	10	M	SO	1.26	57.8
12	11	M	SE	2.71	58
13	12	F	SO	1.44	81.2
14	13	M	SE	0.53	90.1
15	14	F	JR	0.38	87.1
16	15	F	SE	3.98	60.7
17	16	F	SO	1.51	80.3

Figure 1: Acceptable input file in CSV format, opened in Excel (left) and opened in a text editor (right). A file in XLXS format would match the appearance of the CSV file when opened in Excel.

- The program offers options to ‘Distribute’, ‘Aggregate’,or ‘Cluster’ any given column of data. Then if an attribute column is strictly numeric, the ‘Balance’ operator will be available. Each grouping option just noted will be described in further detail in the “Back-end/Algorithm” section of this report.
- Our project is scalable to a large input file. A large input file may be due to an extensive number of students and/or an extensive number of student characteristics. However, an input file size limit is inplace to prevent overloading of the server or a DDOS attack.
- Our project can handle any specified group size or number of groups between one and the total number of students. In the case where even group sizes are not possible (e.g 10 students in 3 groups), the user may select whether groups may have an extra member or one less member.
- Our project also accommodates fringe cases and bad inputs files effectively. These may include files without headers, files without unique student identifiers, or files

Non-Functional Requirements:

- Performance
    - Our service must return groups determined by the algorithm in a short amount of time.
- Reliability/Repeatability
    - The product must always return groups in either a purely random fashion, or according to the algorithm, depending on user choice.
    - Output is displayed on screen, and output files are created in txt, csv, and zip formats.
    - The product must inform the user that the algorithm uses randomness and the same input will return different outputs.

    - If it is impossible for all requested grouping criteria to be met, groups will be made to the best of the product's ability, prioritizing rules in the order they are listed by the user.
    - Failure to meet all requested criteria is indicated to the user. Specifically, for any group that does not meet every requested criterion, a warning is printed for the user.
- Cost
    - The product has a budget of $0 and must utilize free open source software and code.
- Usability
    - The use of the product must be intuitive and provide example inputs.
    - Any mistakes in the input excel file must be caught and relayed to the user to resolve.
    - The program displays links to the source code for the project, as much as is secure to post publicly. (At least, there are links to GroupENG source code.)

Stretch Goals:

- Accessible to teachers across campus (via manual distribution of an executable or link, at least; possibly as a Trefny Center/ITS service).
- Talk to Trefny center and/or ITS at Mines to push this product to all faculty at Mines.
    - Make a web application to be a website hosted by Trefny.
    - Sleek website for easy navigation.
- Our program could be able to accept different formats such as JSON files and CSV files and still interpret them correctly.
- Our program could be scalable to an arbitrary number of students such as 1000 and still function correctly.
- Since Professors will be submitting sensitive student data such as usernames, gender, grades, and GPA, our web app will have security measures in place.
- Our program will be able to save the results of our algorithm and display them while also letting the user download the results in different types of files.
- Program can try to anticipate/calculate an expected wait time based on the input size, to display to the user.

Potential Project Risks:

- Our budget is $0, so we faced a risk when we needed services that required money, such as hosting a website for beta-testing, upgrading a server, or acquiring a HTTPS extension for security.
- Huge(>1500) inputs/variables might cause the grouping program to run slow, or may strain the host server.
    - The implemented algorithm is of low enough complexity that this is unlikely to be an issue.
- Everyone needed to learn programming for Django/HTML/CSS.
- Some members know python, others have to learn it.
- If the website is not secure, student information can be leaked which violates FERPA's PII (personal identification information).

System Architecture:

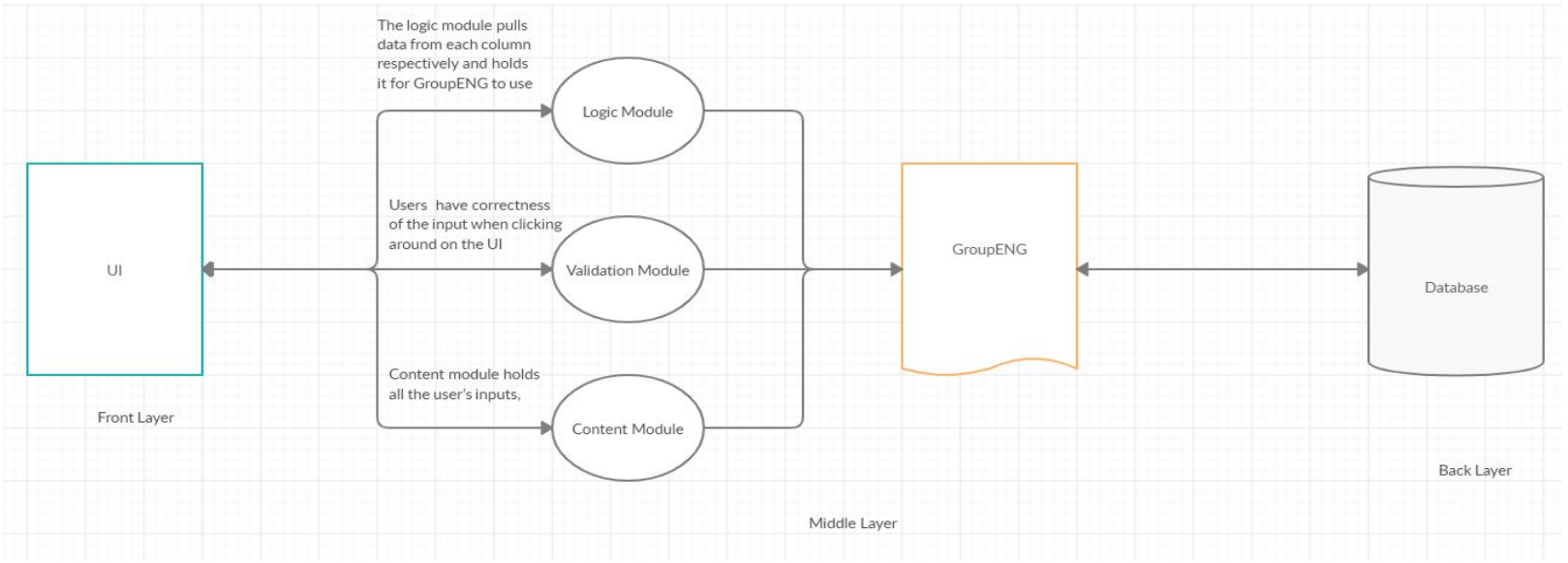


Figure 2: Top-Down Design

For the system architecture, **Figure 2** is a diagram which shows the user interface (front layer) connected to the middle layer which contains the logic module, the validation module, and the content module. All of those are connected to the GroupENG engine which then outputs to a database (back layer). First the user interface will ask the user to upload a file. The logic module then parses through the file and pulls data from each column respectively and holds it for GroupENG to use. Next, the validation module allows the user to be able to have correctness of the input when clicking around on the UI, and then the CNT which stands for content module holds all the user's inputs, like how they want to group their students, their groupsize, etc.

Technical Design:

Stretch Goal Related Design Issues:

- Familiarizing with resources such as Django and LTI
    - When our team met on 5/18/20, we decided that Wordpress would not be a good tool to utilize because it cost money to use, and also that it conflicted with our $0 budget. Django is the main driving force for building the website due to django

being dynamically compatible with python, which allows us to implement our program onto the website.

- Security and protecting student data is important. This means that we need to adhere to FERPA guidelines.

Basic Wireframe/Front End:

This wireframe includes 4 basic screens that will be shown to the user as seen in **Figure 3**. The first screen is the home page as well as the page that will accept a file type from the user. Once the file is submitted, our code will check the formatting and make sure that it can be interpreted correctly. If any of these tests fail, it will push the user to an error screen, forcing them to restart the process. If the tests pass, it will take the user to another page detailing the different columns they inputted, and it will allow the user to rank them based on which ones are most important. It will also describe the different operators which can be applied to each column: Distribute, Aggregate, Cluster, Balance, or Ignore. Once the user is satisfied, a final results screen will be displayed with the groups formed. An option will also allow the user to download the results, in case they want it in a more concrete format. Once they are done, they can either select remake groups to try and find a different combination with the same students, or upload a new file, completely restarting the process. For reference, please refer to **Figures A.1-A.5** in Appendix A to see the current web pages of the GroupENG app.

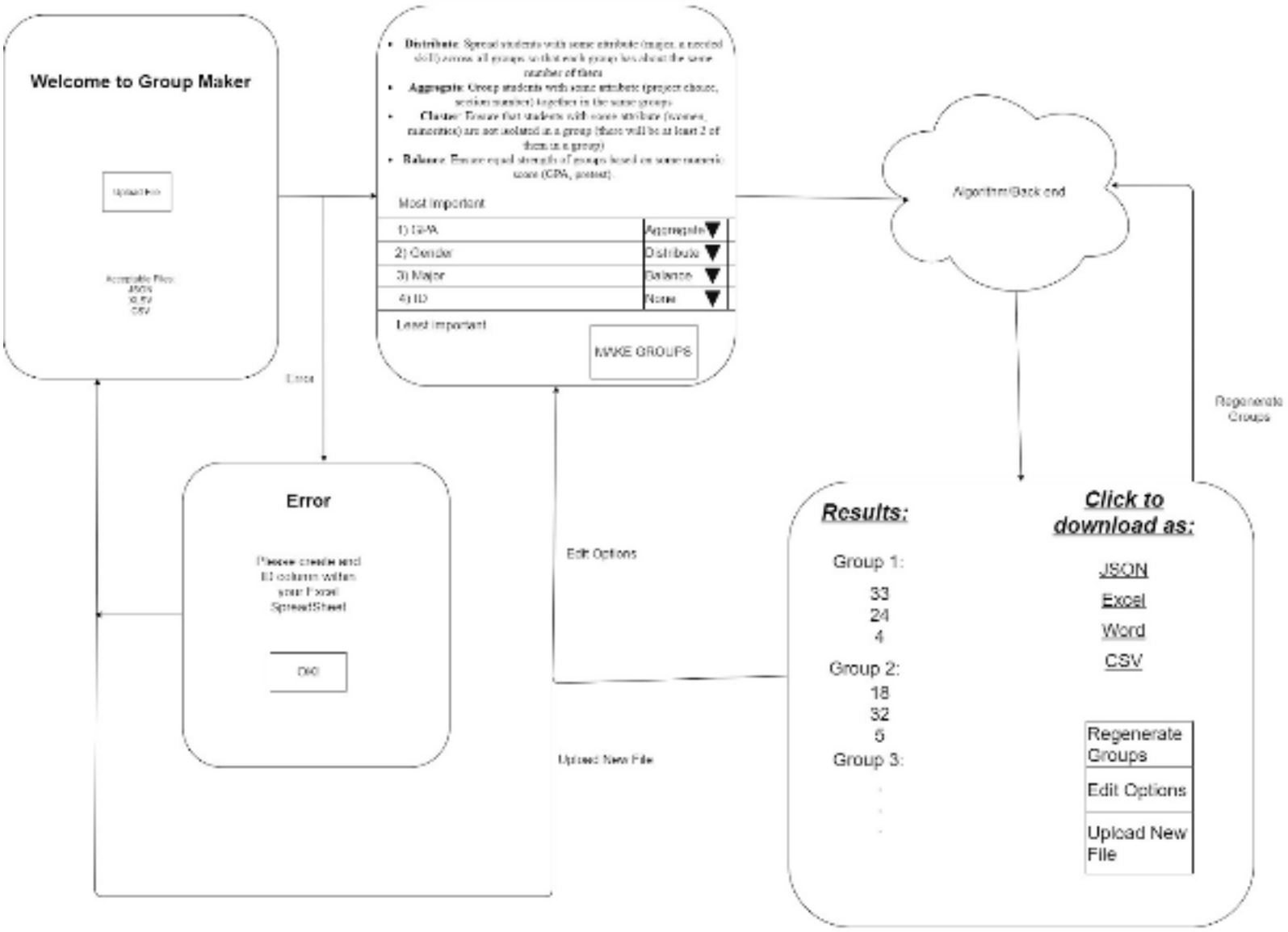


Figure 3: Wireframe of front end

Algorithm/Back End:

Figure 4 (below) shows a flowchart of the path the algorithm takes to form groups. The user (a faculty member, typically) chooses an operator for each column (each 'rule'). They also set a priority for each column by which the grouping is being done. These things combine to form the Set of Grouping Rules which are given to the GroupEng algorithm. Alongside these rules, the

algorithm is fed the Student Data File itself. With all of these inputs, the algorithm is able to produce the student group assignments, as well as some summary data about the groups.

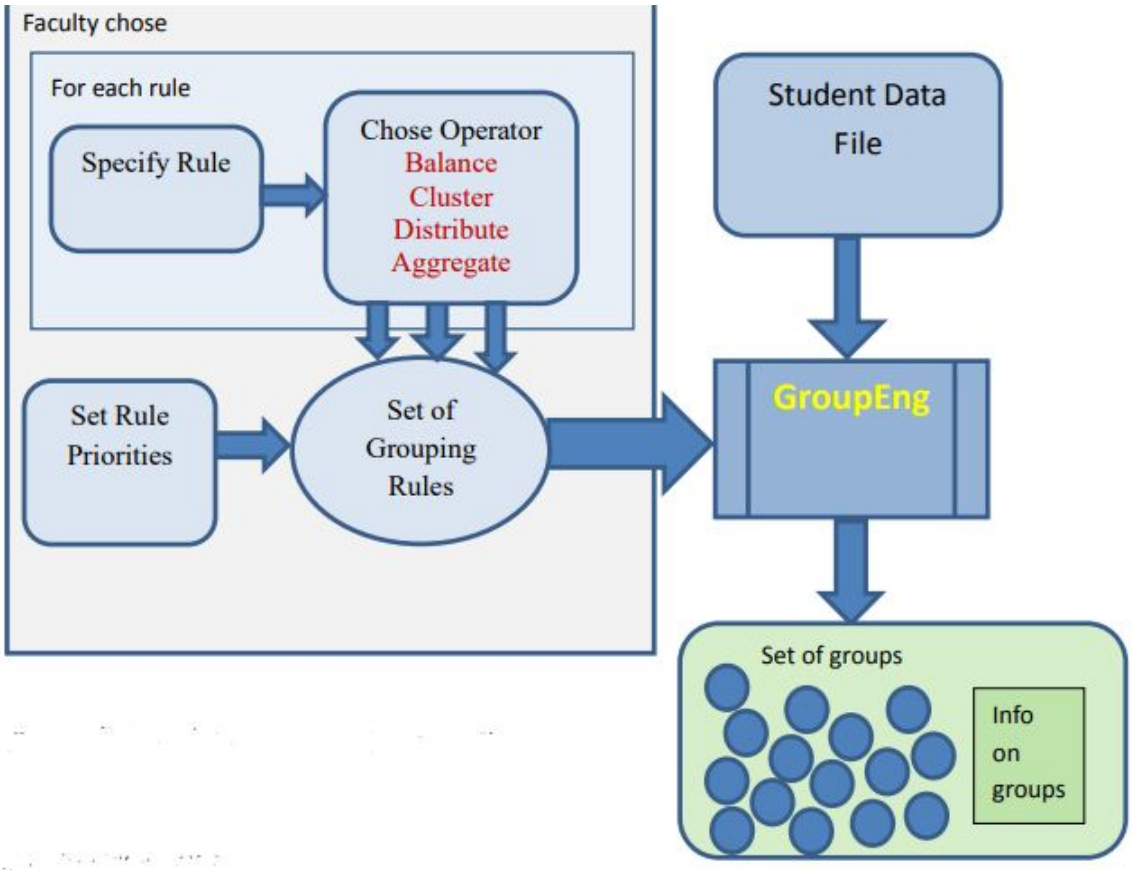


Figure 4: Schematic of GroupEng operation [1]

GroupEng is implemented using a *heuristic-guided stochastic greedy algorithm*: this means that the algorithm has random elements, and that it decides whether or not a group is of high quality based on the following heuristics (definitions):

- **Cluster:** Attempts to not isolate certain people in groups by clustering students with like traits. Not met by groups which contain exactly 1 student with the given value. Recommended for traits like gender and ethnicity to ensure certain groups of people aren't isolated.
- **Aggregate:** Form groups whose members are alike in a particular property. This forms homogeneous groups. It is not met by groups which mix values for the given attribute. Often 1 group will break this rule for each value under consideration because the number of students is not evenly divisible by the group size. It is recommended for traits that you want to be shared across all members like project choice, major, grade year, and availability. [1]

- **Distribute:** Spread students with a particular attribute out across groups. It is not met by groups which have more or less than their “share” of members with any attribute value under consideration. A group’s “share” of members with a certain attribute is determined by the formula: (# students with attribute value)/(# groups), or the two integers on either side of this value (floor/ceiling) in the event that the formula produces fractions. It is recommended for traits that you want to be in each group like major, skills, or english proficiency. [1]
- **Balance:** Strives to create “fair” groups by keeping each group’s average attribute values within a tolerance of the class’s average. It is not met by groups for which the standard deviation of group members’ values for the attribute is larger than a tolerance times the class’ standard deviation for that attribute. This operation only works with numeric values such as GPA, test scores, or class grades.

Quality Assurance:

To ensure the best quality while designing our product, here are some things that we decided to do during the development process.

- ***Unit Testing****:*
  - Before updates are pushed to GitHub, a developer will assert that all tests listed (plausibly in a Google Doc) are still passing.
  - Additionally, many of the python files used in the project run tests of their own in *main*, emulating the behaviors one would expect of a JUnit test in a Java context.
- ***User Interface Testing (automated or otherwise)***
  - Our project relies mainly on the feedback from beta testing, as well as frequently repeated tests done by the developers, in order to assert that the user interface works as intended. Relying on beta testers enables us to test the buttons and other UI elements across diverse sets of hardware/browsers.
- ***Integration Testing***
  - Some of our individual parts include the source code for GroupENG and the GUI through HTML. When combining these, we will need to test if everything still functions the way we intended it to, as well as still making the groups.
- ***Code Reviews***
  - Code implemented by one team member was always read over by at least one other team member to ensure that it was readable and that it accounts for edge cases.
- ***User Acceptance Testing***
  - We contacted Mines Professors through Dr Walton and asked them to use our website and see if it meets their specifications. This is our target demographic, so we made sure that the users know how to use the program, with or without technical knowledge.
  - We showed this program to close friends/family with limited technical knowledge to get more feedback in case anything is unclear. Our intention was to make this app as user friendly as possible.
- ***Code Metrics***

    - We will assert that the runtime of the algorithm scales reasonably to a large number of students and student characteristics. We did this by sending edge cases and extremely large datasets averaging 1000 students to ensure that our code executes in a reasonable time.
- ***Static/Dynamic Program Analysis***
    - Web app development will have dynamic analysis because the website will be updated in real time. This allows for output checks while coding.

Results:

- ***List of Features We Did Not Have Time to Implement***
    - A bug reports submission button
- ***Performance Testing Results***
    - We had a checklist and a timer while testing our prototype to test the two biggest features that we need to implement, ease of use and short execution time.
- ***Summary of Testing (e.g., browsers)***
    - At the bottom of many of the python files, main() tests the functionality of that particular file's functions/scripts
    - Beta testing allowed us to test the web app across diverse browsers/connections/hardware
- ***Results of Usability tests (e.g., testing educational software with real students)***
    - Dr Walton spread our project out to professors willing to participate in beta testing, and here are some of the feedback we received:
        - Pro:
            - Visually Appealing UI
            - Easy to read texts
            - Provide easy to use navigation between pages
        - Con:
            - More specific instructions are needed for each page
            - The "One extra member" or "One less member" buttons should show up for both group sizing options. Also, more explanation is needed for what these buttons mean.
            - The suggested/default GPA tolerance, as presented to the user when they are on the Attribute Menu, is still somewhat high and needs fine-tuning.
- ***Future Work (e.g., any ideas you may have for extensions)***
    - Our work could be expanded on in such ways as:
        - Experimentation with other algorithms or making improvements on the current algorithm
        - Groups could be saved to a teachers' account for later accessing; stored in a secure database, to facilitate such services as those listed below (likely requiring students to create groups as well)
        - Peer Evaluation (groups made can then evaluate one another throughout the semester)

        - Interteam Reviews (reviewing one anothers' presentations, or giving feedback on rough drafts, etc)
        - Extra security measures such as file access only to specific users with session keys.
- ***Lessons Learned***
    - Time flies. It was a little bit tight to get our project done in the allotted amount of time, and we were fortunate enough to discover an open-source algorithm, eliminating the need to write our own
    - Version control is important for delegating programming tasks. Use of GitHub made it much more attainable to assign individuals their own programming tasks, because all changes were tracked and revertible.

Appendices:

Appendix A

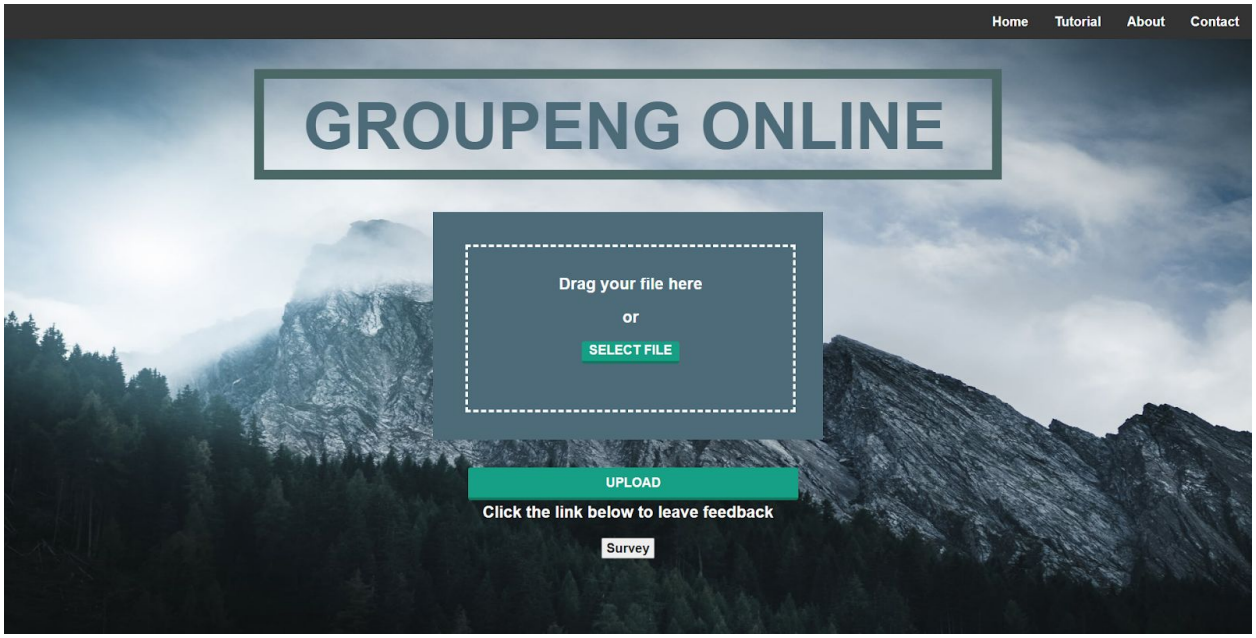


Figure A.1:Home Page

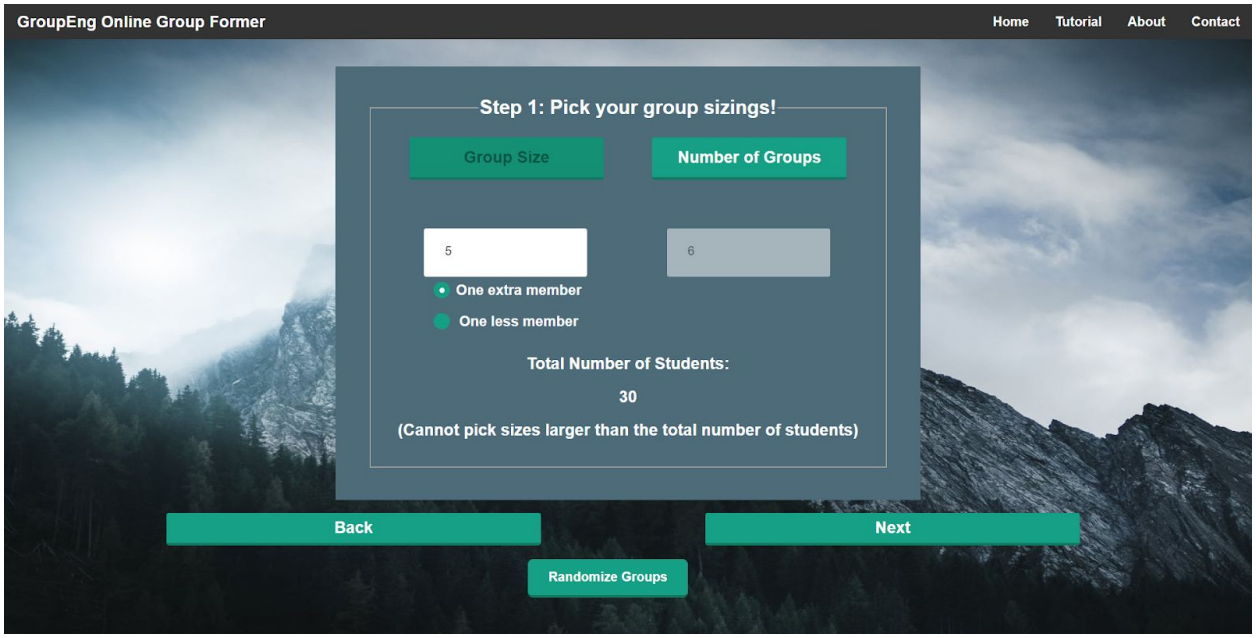


Figure A.2:Step 1, Pick group size page

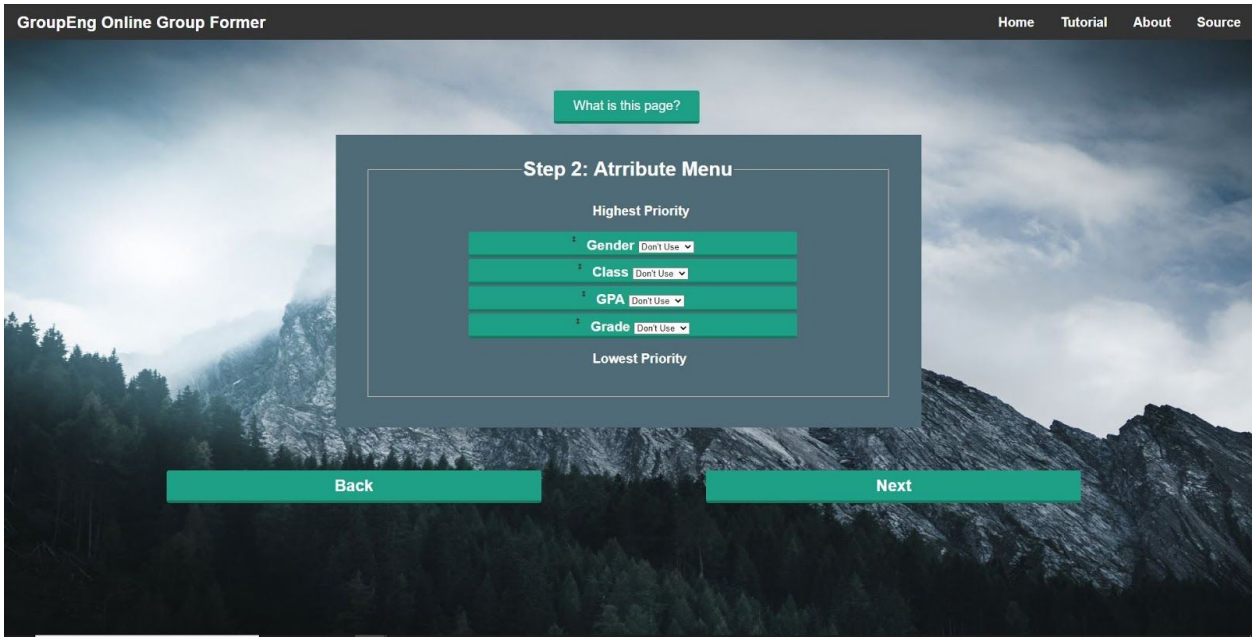


Figure A.3: Step 2, Attribute Menu Page

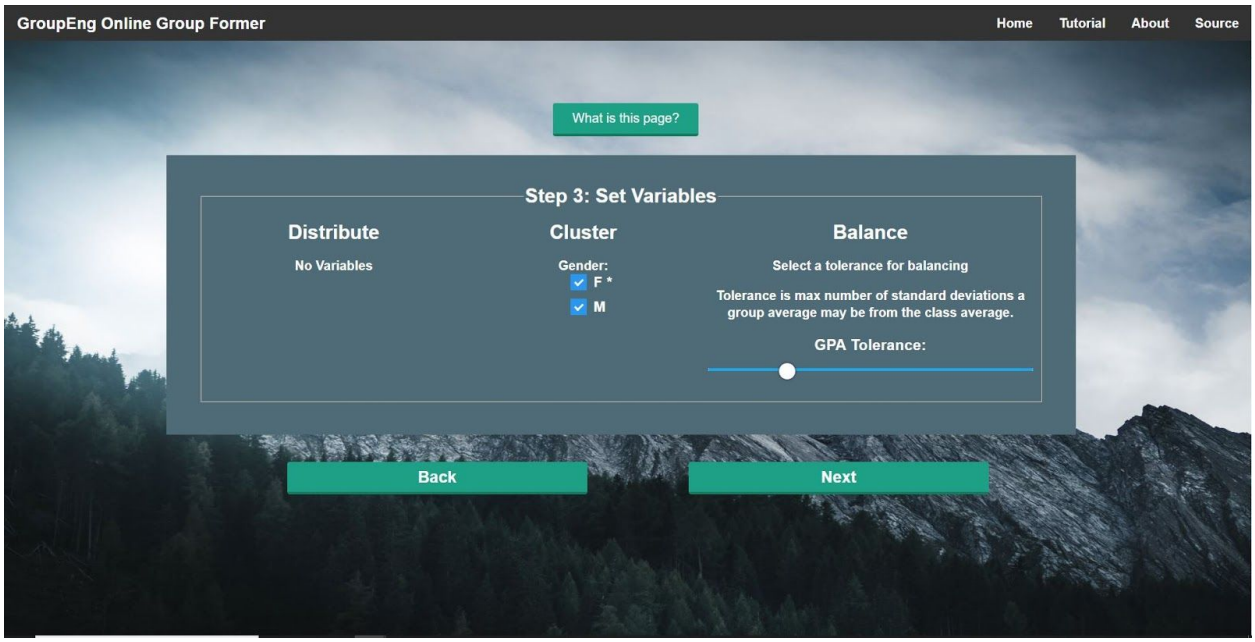


Figure A.4: Step 3, Setting Variables Page

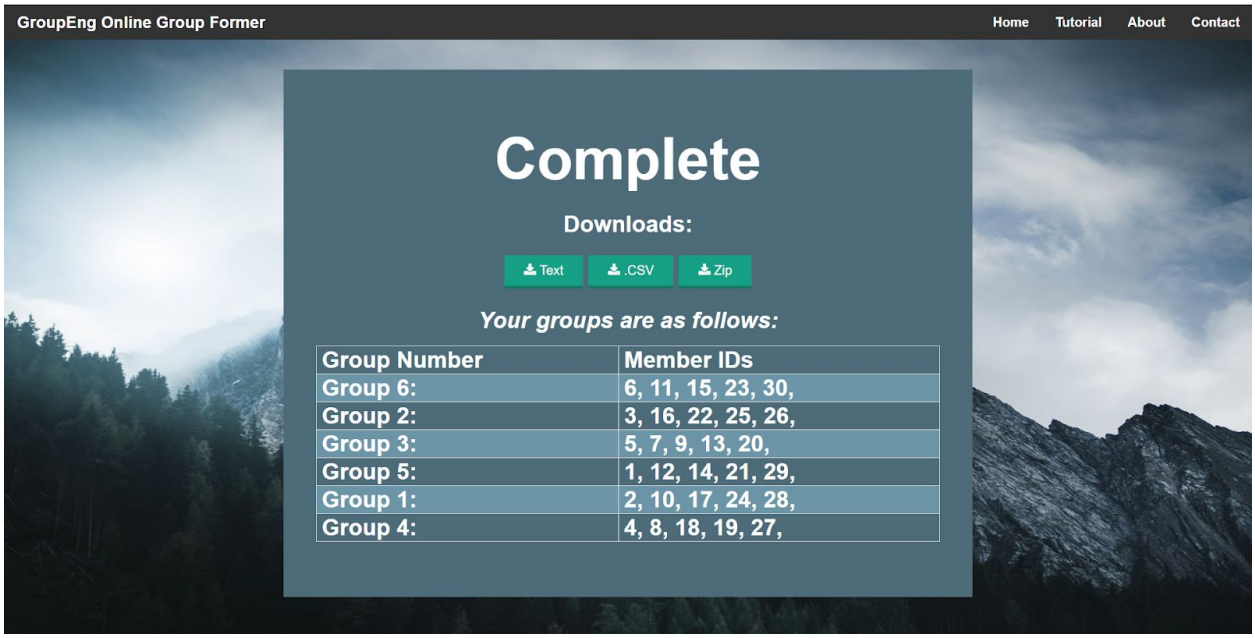


Figure A.5: Final Download Page

References:

[1] Thomas G. Dimiduk, Kathryn C. Dimiduk. (2011). Schematic of GroupEng operation. Effectively Assign Student Groups by Applying Multiple User-prioritized Academic and Demographic Factors Using a New Open Source Program, GroupEng. Harvard University/Cornell University