Group Maker Requirements Document

Table of Contents

1. Introduction

1.1. Purpose

1.1.1. This document is an overview of the Group Maker application. Here you can find more detailed information about the program's requirements, objective, and usability.

1.2. Scope

1.2.1. Our main client is Karen Anewalt, a Computer Science Professor at the University of Mary Washington. This program will mainly be used by the professors in the Computer Science Department. This will not be shared with other universities or departments servers.

1.3. Reference

1.3.1. The client will provide an excel sheet of what the student data will look like.

1.4. Overview of the Remainder of the Document

1.4.1. This document is organized into: project description, describing the overview and functionality of the program; requirements, listing the main services provided; non-requirements, services not provided; and a list of feature.

2. Project Description

2.1. System Overview

2.1.1. This project is being made so that computer science professors will have less trouble when trying to assign groups at random and can use the project to ensure true randomness while being able to place limitations and make edits to groups.

2.2. Client Characteristics

2.2.1. Our client is Karen Anewalt, she decided she wanted this system as she was assigning group members to ensure that she could have a random system so that she would not have to assign group members by hand. She wanted to be able to have truly random groups made partially so that she did not have to do it by hand but also so that she does not subconsciously bias her groups.

2.3. User Characteristics

2.3.1. The users for the system is Karen Anewalt and other Computer science professors. The professors will use it exclusively for assigning groups at random so all the professors may not use

the system. The system can be pushed out to other professors at umw in the future.

2.4. Product Functions

2.4.1. The product shall be able to assign group members randomly with a maximum group size, it shall be able to have a weights system to assign different group members with a blacklist/whitelist that shall be considered when making groups. The professors shall receive a notification and and can edit group members by moving them or reorganizing them. The program shall also be able to keep track of previous groups so that professors may have different group orientations when assigning different projects throughout the semester.

3. Requirements

3.1. Group Randomization

Description: The user enables randomization of groups.
Main Flow:

1. The user selects the setting to enable group randomization.
2. The system adds randomization to the list of parameters.

Alternate Flow:

1. The user deselects the setting to enable group randomization.
2. The system removes the randomization parameter from the settings.

3.2. Group sizes

Description: The user will provide a maximum group size for the program to make groups.
Main Flow:

1. The user specifies the maximum group size to the program.
2. The program will output groups of the appropriate size.

3.3. Read csv

Description: The user will provide a CSV in which the program will read and use the data to make the groups.

3.3.1. Formatting original upload

Description: The format for the CSV will be ID number(1 to class length), last name, first name, and then 3 numeric value columns. The numeric values are up to the professor's choice, but are usually numbered between 1 and 10. An example of a numeric value is gender, personality, or grade. The prefered and black list is also a numeric value where the values are a list of ID numbers.

Main Flow:

1. The user will create a CSV file from google forms.
2. The user opens the program and inputs the CSV file.
3. The program shall parse the CSV and output the data on the screen.

Alternate Flow:

1. The user will create a CSV file from a paper questionnaire.
2. The user opens the program and inputs the CSV file.
3. The program shall parse the CSV.
4. If there is a missing main value, names or ID number then the program will not read the CSV.

3.3.2. Formatting repeating group

Description: This CSV will contain ID number(1 to class length), last name, first name, 3 numeric value columns, and a repeat column. The repeat column shall be a list of ID number of previous group members

Main Flow:

1. The user creates groups.
2. The user exports the created groups CSV.

Alternate Flow A:

1. The user imports a previous CSV with the repeat column.
2. The program shall parse the CSV and output the data on the screen.

3.4. Output

Description: The format for the output will be a csv file with the groups allocated properly. Before the output is given it will be displayed on a screen with a warning if there a group size that is too small. The program will also allow the user to edit group sizes.

3.4.1. On screen - export csv

Description: The output will be displayed on the screen and there will be a button for exporting as a CSV

Main Flow:

1. The user takes steps to make groups with the program.
2. Once the program is done it displays groups on the screen.

Alternate Flow:

1. The user may press a button that will export groups in a csv.

3.4.2. Warning on screen for groups too small

Description: The program will display a warning of some sort should there be a group that is not the specified group size.

Main Flow:

1. The program will create groups based on given criteria
2. It will then display the output in the form of groups
3. If the groups are all the correct sizes there will be no warning on screen

Alternate Flow:

5. The program will create groups based on given criteria
6. It will then display the output in the form of groups
7. If there is a group that is not the specified group size there will also be a warning on screen so the user knows.

3.4.3. Edit groups

Description: the program must be able to allow users to edit the groups so that if there is any problem in group size or blacklist/whitelist that the program overlooked the user can edit the groups themselves.

Main Flow:

1. The program will make groups based on the given criteria to the best of its abilities.
2. The program will then output the groups to the screen.
3. If the user is satisfied no further action should be done.

Alternate Flow:

1. The program will make groups based on the given criteria to the best of its abilities.
2. The program will then output the groups to the screen.

3. The user will edit the output if need be so that the groups are satisfactory, when editing neither group limit not whitelist/blacklist will be considered.

3.5. Adding Custom Criteria

Description: The user adds a custom criterion to be factored into group creation.
Main Flow:

1. The user indicates to the system that they wish to add a custom criterion to the parameters.
2. The user inputs the number of different categories for the criterion.
3. The system adds the custom criterion to the list of parameters to account for.

Alternate Flow:

1. The user indicates to the system that they wish to add a custom criterion to the parameters.
2. The user does not input the number of different categories for the criterion.
3. The system presents an error to the user until the user completes the required field.

3.6. Removing Custom Criteria

Description: The user removes a custom criterion previously added.
Main Flow:

1. The user selects one of the custom criteria from the list of settings.
2. The user indicates to the system to remove the custom criterion.
3. The system removes the custom criterion from the list of parameters.

Alternate Flow:

1. The user selects one of the custom criteria from the list of settings.
2. The user decides not to remove the criterion and cancels the operation.
3. The criterion is left unchanged.

3.7. Defining Custom Criteria

Description: The user indicates that a custom criterion should be weighted to either group based on similarity or differences.

Main Flow:

1. The user selects a custom criterion from the list of settings.
2. The user indicates that students should be grouped with other students of similar scores.
3. The user inputs a weight to indicate the relative importance of the criterion.
4. The system updates the criterion in the list of parameters with the weight and closeness setting.

Alternate Flow A:

1. The user selects a custom criterion from the list of settings.
2. The user indicates that students should be grouped with other students having different scores.
3. The user inputs a weight to indicate the relative importance of this criterion.
4. The system updates the criterion in the list of parameters with the weight and difference setting.

Alternate Flow B:

1. The user selects a custom criterion from the list of settings.
2. The user makes an indication on whether students should be grouped with similar or differently scoring students.
3. The user does not input a weight to indicate importance.
4. The system presents an error to the user.
5. Flow is directed back to step 1.

Alternate Flow C:

1. The user selects a custom criterion from the list of settings.
2. The user does not indicate whether students should be grouped with similar or differently scoring students.
3. The user inputs a weight to indicate the importance of the criterion.
4. The system presents an error to the user.
5. Flow is directed back to step 1.

3.8. Changing Custom Criteria

Description: The user changes a custom criterion that was previously defined.

Main Flow:

1. The user selects one of the custom criteria from the list of settings.
2. The user changes the similar/different score grouping option.

3. The system updates the criterion with the new grouping setting.

Alternate Flow A:

1. The user selects one of the custom criteria from the list of settings.
2. The user changes the weight of importance for the selected criterion.
3. The system updates the criterion with the new weight.

Alternate Flow B:

1. The user selects one of the custom criteria from the list of settings.
2. The user decides not to change any of the settings for the selected criterion.
3. The system does not make any changes to the criterion that was selected.

3.9. Blacklist

Description: This is a list of student(s) who should not be in the same groups.

Main Flow:

1. The user will input the CSV, with “blacklist” being the name for one of the columns.
2. The user will select the blacklist option before creating groups.

Alternate Flow A:

1. The user will input the CSV without a blacklist column.
2. The user will select blacklist and manually input the blacklist.
    a. The input will be in pairs
3. The user will then create groups.

Alternate Flow B:

1. The user will input the CSV without a blacklist column.
2. The user will create groups.

3.10. Preferred

Description: This is a list of student(s) would would like to be in the same group.

Main Flow:

1. The user will input the CSV, with “preferred” being the name for one of the columns.
2. The user will select the preferred option before creating groups.

Alternate Flow A:

1. The user will input the CSV without a preferred column.
2. The user will create groups.

Alternate Flow B:

1. The user will input the CSV without a preferred column.
2. The user will select preferred and manually input the data.
    a. The input will a list.
3. The user will then create groups.

3.11. No repeating groups

Description: The program will keep track of which students have been together in a group before. This information can be saved in a CSV

3.11.1. Save feature csv

Description: The output CSV will have a repeated section. This will be a list of ID numbers.

Main Flow:

1. After the groups have been created, there shall be an export CSV option.
2. Once exported, the CSV will have a repeated column

Alternate Flow:

1. After the groups have been created, there will be an option to create another group.
2. This new group will not have any repeated members.

4. Non-requirements

There are a few services that this project will not provide. Below is an explicit list of services that are not included in this project.

4.1. The system shall not be accessible by students.

4.2. The system shall not maintain any sort of database or any form of persistence.

5. Assumptions

The included assumptions are the conditions that are assumed to remain true during all user interactions with the system. These assumptions are based on the requirements of our system. Listed below are the assumptions:

5.1. Users are Computer Science professors
5.2. All users have a CSV file containing a class roster as input to the program.
5.3. The system will not modify the input file
5.4. The user may specify additional criteria for the system to take into account
5.5. The system will only keep data persistent during execution

6. Appendices

6.1. Glossary of Terms Related to your project

6.1.1. This document will use the terms shall and shall not
6.1.2. This document will use “use cases” and not “user stories”

6.2. Author Information

6.2.1. This Document was prepared by Anum Qureshi, Evan Shipman, Jacqueline Coates.

6.3. Additional Documents

6.3.1. CSV roster of students

6.3.1.1. Client provided what they think the template should look like, but is open to suggestions

unique number ID (probably just numbers 1-40), last name, first name, numeric value1, numeric value2, numeric value3,

1, smith, joann, 1, 4, 10
2, adams, sally, 1, 2, 10
...