Project Plan Document

for

Group Maker

Prepared by:

Anthony Phimmasone

Denzel Saraka

James Donahue

Igor Asipenka

CPSC 430

02/19/19

Table of Contents

1. Introduction 3
1.1 Purpose 3
1.2 Scope 3
1.3 References 3
1.4 Overview of the remainder of the document 3
2. Project Description 4
2.1 System overview 4
2.2 Client characteristics 4
2.3 User characteristics 4
2.4 Functional requirements 4
2.5 General constraints 7
3. Project Schedule 8
3.1 Approach 8
3.2 Milestones and Deliverables 10
3.3 Work Breakdown Structure 12
3.4 Gantt chart 14
3.5 Task Dependency Diagram 15
4. Appendix 16
4.1 Glossary of terms related to the project 16
4.2 Author Information 17

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed schedule and plan of how the implementation team will deliver a completed product to the client. This project plan contains several factors that are integral to delivering a successful product on time: priorities, scheduling, milestones, and personal approaches. This document is intended to serve as a plan for the implementation team and the client as well.

1.2 Scope

The purpose of the group making software is to form individual groups from an input CSV file that contains various metrics that can be used to distribute students into certain groups. Additionally, the client will be able to select options from a menu to specify how the groups should be created and what factors to include in the randomization process.

1.3 References

Software Requirements Specification for Group Maker
Group Maker Requirements Document

1.4 Overview of the remainder of the document

The remainder of the document is organized in the following way:

- Section 2 [Project Description]: This section presents a general overview of the product, introduces the client and users, explains the project constraints, and provides priorities for the functional requirements.
- Section 3 [Project Schedule]: This section provides the breakdown of the proposed 5 stage schedule, defines milestones and deliverables, includes a work breakdown structure, and presents a Gantt diagram of the planned implementation schedule.

2. Project Description

This section gives an overview of the product, introduces the client, explains who the users are, explains what the constraints of the project are, and assigns priorities to the functional requirements.

2.1 System overview

The product will be a Python-based software that will create groups based on a CSV file. The product will import the CSV file, parse through it, and ultimately group students together based on the condition that the user chooses. The user will be able to sort groups based on preferred lists, black lists, personality, and gender. The user will also be able to set the number of groups once they start running the program. The program will display the groups to the user after they are created. It will also create an output CSV that will contain all the groups that have been run by the program.

2.2 Client characteristics

The client is Dr. Anewalt. She is a member of the University of Mary Washington's Computer Science department. She requested this project because she wanted an easier way to create groups for her computer science classes. She wanted a system that could make these groups based off certain conditions, such as a preferred list.

2.3 User characteristics

The intended users of this project are all the professors in the UMW Computer Science department. All the professors have similar needs and wants as the client when it comes to creating groups for their classes. This will allow the product to be useful for the entire Computer Science department. Students and other departments of Mary Washington will not have access.

2.4 Functional requirements

Requirement 1: As a user, I want to be able to pass a CSV file created from a Google Forms survey, so that the software will have all students that I want to put into groups. (Priority: 1)

The software must be able to read an input CSV file created from data gathered through a Google Forms survey, passed in as an argument.
Source: Denzel Saraka

Requirement 2: As a user, I want the CSV file in the same directory as the software so that the software will have access to the CSV file. (Priority: 1)

The user must run software within the same directory as the CSV file.

Source: Denzel Saraka

Requirement 3.1: As a user, I want the software to write a CSV file that contains all groups generated by a single run of the software, with each student's name and each group sorted by numbers. (Priority: 1)

The software must be able to write an output CSV file with sorted groups numbered and containing the names of the students in each group.

Source: Denzel Saraka

Requirement 3.2: As a user, I want the software to write a CSV file that contains multiple possible sets of groups instead of just one group set. (Priority: 1)

The software must be able to write an output CSV file with multiple sets of groups.

Source: Denzel Saraka

Requirement 3.3: As a user, I want the software to export the CSV file into the same directory as where the original file was, or the directory of my choosing for my convenience. (Priority: 2)

The software must be able to write the output CSV file into the local storage of the user's machine, into a directory of their choosing.

Source: Denzel Saraka

Requirement 4: As a user, I want to be able to give the software the necessary information when prompted in order to get the best results possible. (Priority: 1)

The software must prompt the user for specific parameters for group making use including but not limited to: group size, number of group sets, randomization, student

"blacklists", group generation by similarities or differences, and whether students should be included in more than one group.

Source: Denzel Saraka

Requirement 5.1: As a user, I want to be able to use the software through the command line as it is straightforward and easy to use. (Priority: 2)

The software must be utilized through the command line.

Source: Denzel Saraka

Requirement 5.2: As a user, I want the software to display the prompts for my input through the command line. (Priority: 1)

The software must display menu options through the command line.

Source: Denzel Saraka

Requirement 5.3: As a user, I want to specify the size and/or number of groups to create, as well as whether a single individual can be part of more than one group. (Priority: 4)

The software's menu will display the following questions in this order through the command line: "How many groups will be made?", "How many sets of groups will be made?". If there will be more than one set of groups, software will ask "Will there be repeated members within groups?".

Source: Denzel Saraka

Requirement 6.1: As a user, I want to be able to look at the generated sets of groups before they are written to make sure that everything looks correct. (Priority: 1)

The software will allow the generated sets of groups to be shown to user before being written to an output CSV file.

Source: Denzel Saraka

Requirement 6.2: As a user, I want to be able to change the groups if I spot a problem with them before the groups are finalized. (Priority: 4)

The software will allow the user to make changes to the groups after being shown to the user, but before being written to an output CSV file.

Source: Denzel Saraka

Requirement 7: As a user, I want to receive a clear, informative error message when I enter invalid input into the software, or when something goes wrong with the software. (Priority: 1)

The software must display an error message when not given the correct information.

Source: Denzel Saraka

Requirement 8: As a user, I want to be able to create groups formed from a metric with similar or different values. (Priority: 1)

For each metric the user uses to create groups, the software will ask if the groups should be formed by using similar or different values of that metric.

Source: Denzel Saraka

2.5 General constraints

Non-functional Requirement 1: As a stakeholder, the budget set for this software project is $0.00 because that is what is left in the budget for the semester. (Priority: 1)

The budget for the product is $0.00.

Source: Denzel Saraka

Non-functional Requirement 2: As a user, I want the software to have a runtime of less than 5 seconds, as I want groups to be made almost immediately. (Priority: 1)

The software must have a runtime of less than five seconds 95% of the time after all information has been collected from the user.

Source: Denzel Saraka

3. Project Schedule

Our group has broken down the schedule into five stages: reading a CSV, writing to a CSV, menu options, a user interface, and basic functions of the software. We will complete these stages based upon priority and dependencies. Each of these stages are explained in detail below.

3.1 Approach

Stage 1

The first stage in our schedule is reading from a CSV file. This is the first stage that needs to be completed because most of the functional requirements depend on this stage. We estimate that the whole process of reading (design, implementation, testing) will take two hours.

We estimate that most of the time spent in this stage will be testing the reading functionality. To test the CSV file, we will start with sample data from a google form survey and read in the CSV file. It is necessary to ensure that the CSV file is read in correctly.

Possible problems that may arise during this stage are converting the file into a format that is easily accessible within Python. To solve this issue, we will heavily research Pandas and DataFrames.

Stage 2

The second stage in our schedule is writing to a CSV file. This stage is heavily dependent on reading from a CSV file and holds many dependencies such as menu options and user interface. We estimate that the writing process will take about four hours.

This stage in the implementation process is also high in priority compared to the four other stages listed in this section. The four-hour estimate is necessary in order to make sure that the software exports a properly formatted CSV file. The software will sort the students into groups and assign each group an identification number. It will also write a single output CSV file with the randomized group information and store the file into a directory of the user’s choosing.

Possible problems that may arise in this stage include exporting the results as a CSV file, creating only one file, and storing the file in a specified directory. The prior research in Stage 1 will be crucial, as well as reading the Python Man pages.

Stage 3

The third major stage is our implementation schedule are the menu options. We estimate that the options will take twelve hours.

We plan to split these hours across four days in order to properly implement the menu as a whole. The menu will prompt the user to select parameters: group size, number of group sets, randomization, student "blacklists", group generation by similarities or differences, and whether students should be included in more than one group. The menu options are heavily dependent on Stages 1 and 2 and rank high in prioritization.

Possible problems that may arise in this stage include randomization and properly inputting chosen parameters. We will create a randomization function based on a random number generator will initially just randomly place students into groups. After verifying this function is working as intended, we will implement an algorithm to properly handle the inputted parameters.

Stage 4

The fourth stage in our schedule is creating the user interface. This stage in our implementation process is heavily dependent on the previous three stages. We estimate that the user interface will take twelve hours.

Similarly to Stage 3, we plan to break this stage down and work on it throughout several days due to the importance of providing a functional user interface. Many of the requirements within the user interface stage are high and dependent on the previous stages, therefore the user interface needs to be completed next in the process. The software will utilize a command line interface display menu options. The software will also allow for the generated sets of groups to be shown to the user before being written to an output CSV file, which will allow for the user to make changes to the groups before final output.

Possible problems that may arise in this stage include input verification and altering the groups after creation. We will properly check the input from the menu options and ignore improper input error messages until Stage 5.

Stage 5

The final stage of our implementation scheduling process is working on error checking and supplementary functions of the software. Error detection is the only requirement in this stage, and it is dependent on all the previous stages. Therefore, we will complete this stage last in the scheduling process. We estimate that the error checking functions of the software will take five hours.

Within this stage we will focus on displaying an error message when incorrect information is given. This requirement is a lower priority than all the other requirements; as a result, this stage will be completed last. Additionally, if the schedule is completed on track, extra time can be used to create supplementary features that would facilitate use of the software- such as a graphical user interface.

Possible problems that may arise in this stage include error detection and time management. Errors are sometimes difficult to catch, so we will create test cases and test the program against incorrect input from the user, incorrectly formatted CSV files, and bug prone functions (randomization algorithm).

3.2 Milestones and Deliverables

The following section includes information about the milestones and deliverables that we will follow when creating this software. There are three main milestones each containing mini milestones or subtasks that software needs to have.

Milestone 1

Milestone 1 will include all the high priority tasks, such as reading to a CSV file, writing to a CSV file, creating the menu options, handling the group creation parameters, and creating part of

the user interface. It will initially include a very simple menu and be able to read and write to a CSV file. This milestone will take approximately five days to a week to complete. This milestone contains five subtasks and deliverables as stated below.

[Milestone 1.1]- Read from a CSV file created from data gathered through a Google Forms survey.

[Milestone 1.2]- Run the software within the same directory as the CSV file.

[Milestone 1.3]- Write to the output CSV file with multiple sets of groups and sorted groups containing names of the students in each group, as well as group number.

[Milestone 1.4]- The software will display menu options through a command line.

[Milestone 1.5]- The software will display an error message when not given the correct information.

Milestone 2

The second milestone is Milestone 2. This milestone will include high and mid priority tasks such as, storing output into a local storage on the user's machine, creating a command line user interface, prompting the user for specific parameters to make groups, and allowing for the user to view and edit the CSV before being finalized. This milestone contains five subtasks and will ultimately contain a working user interface, menu options, and functions of the software.

[Milestone 2.1]- Storing output into a directory of the user's choosing.

[Milestone 2.2]- Creating a simple command line user interface.

[Milestone 2.3]- Create and utilize random generation for making the groups.

[Milestone 2.4]- Generate groups using selected parameters prompted to the user by the software.

[Milestone 2.5]- Allow the user to edit the CSV file before being finalized.

Milestone 3

Milestone 3 is the final milestone of our software. It will be the final working version of the software and contain all the features and deliverable to the client. In this milestone the lowest level priorities will be completed, and the software will be ready to be tested by the testing group.

Deliverables:

Deliverable 1- A preliminary version with the importing and reading features integrated. It will also be able to run from the same directory as the input CSV.

Deliverable 2- A basic prototype that will be able to create an output CSV. The prototype will also have a working menu and display error messages when it is not given the correct input.

Deliverable 3- A more advanced prototype that will be able to store output CSV into a directory of the user's choosing. It will also have all the different group making features implemented. This version will also allow the user to edit the CSV file before it is finalized.

Deliverable 4- A fully integrated version of the software.

3.3 Work Breakdown Structure

Our group will start the implementation for the project the week before Spring Break (February 25th). We plan to finish the project before the testing plan is due on March 28th. We will begin implementation phase by implementing Milestone 1.1. This milestone involves importing and

reading from a CSV file. The start date for this milestone will be February 25th. We plan on finishing this milestone in one day.

We plan on implementing both Milestone 1.2 and Milestone 1.3 on the same day. Milestone 1.2 involves having the software run from the same directory or folder as the input CSV file. Milestone 1.3 involves creating and writing to the output CSV. The completion date for these two milestones is February 26th.

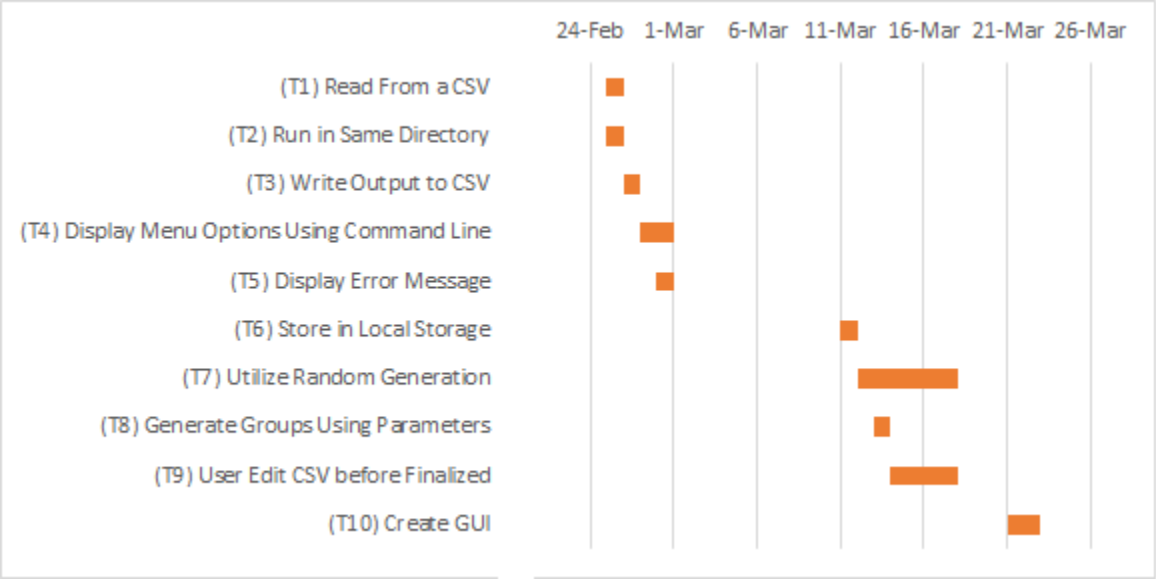
The implementation team will begin integrating the software with the command line, milestone 1.4, on February 27th. The expected completion date is February 28th. We will also be working on Milestone 1.5, which is creating and testing error messages for the software, during this timeframe.

After Spring Break (March 11th), we will begin implementing Milestone 2.1. This milestone will involve storing output into a directory of the user's choice. It will take approximately one day to complete this milestone.

Between March 12th and March 16th, we will be working on Milestones 2.2 and 2.3 in parallel. This is because these milestones can be done in tandem. Milestone 2.2 involves creating the menu that will be displayed to the user. To elaborate, this is the milestone that is responsible for prompting the user to enter their input and choosing their group preferences. Milestone 2.3 encompasses all the group making features that involve randomization.

From March 14th to March 28th, we plan on finishing the remaining milestones. The final milestones are 2.4, 2.5, and 3. These milestones are responsible for creating groups based on specific preferences, allowing the user to edit groups, and fully integrating all the milestones to the software.

3.4 Gantt chart



According to our Gantt chart, we will be running a few tasks in parallel or will be starting them within the same day, this includes reading from a CSV (T1) and running it in the same directory (T2). This is because T2 is dependent on T1. T1 will also not take long to implement. T3 (write to a CSV file) will be implemented the following day. T4 (use a command line) and T5 (display error messages) will also be implemented in tandem. This is because although they are somewhat dependent on each other, they can both be implemented at the same time or day.

While two people work on the command line, two other people can start working on displaying error messages. After these tasks are done, our group will take a break during Spring Break and begin working on T6 (store in local storage) the following Monday. The following three tasks can also be performed in parallel, this includes T7 (utilize random generation), T8 (generate groups using selected parameters), and T9 (allowing the user to edit a CSV before being finalized). This is because these three tasks are independent of each other. Finally, T10 will be completed the following day up until our goal date of March 28.

3.5 Task Dependency Diagram

Task	Effort (Days)	Duration (Days)	Dependencies
(T1) Read From a CSV	1	1	
(T2) Run in Same Directory	1	1	T1
(T3) Write Output to CSV	1	1	T1, T2
(T4) Display Menu Options Using Command Line	2	2	T1, T2, T3
(T5) Display Error Message	1	1	T1, T2, T3
(T6) Store in Local Storage	1	1	T1, T2, T3
(T7) Utilize Random Generation	1	1	T1, T3
(T8) Generate Groups Using Parameters	6	4	T1, T3
(T9) User Edit CSV before Finalized	2	2	T3, T8
(T10) Create GUI	6	4	T1, T4, T5

By examining the task dependency diagram, those can view that almost all the ten tasks are dependent on T1 (read from a CSV file). This is because without reading from a CSV file, the user will not be able to perform any other task. Therefore, reading from a CSV file is incredibly important. The following tasks follow in almost a stair like dependency, meaning that each task is dependent on at least one other task before.

While examining the diagram, one can understand that T4, T5, and T6 are all dependent on T1, T2, and T3. This is because tasks such as reading a file, writing a file, and saving the file to storage are integral to the project. Furthermore, the task for the graphical user interface (T10) is only dependent on the tasks that display items and tasks such as generating groups using parameters is dependent on the randomization through the random generator. Lastly, the final task is dependent on the completed group generation feature, because the groups need to be created before the user can edit them.

4. Appendix

4.1 Glossary of terms related to the project

CSV - CSV stands for “Comma Separated Values”. CSV files are a file format that stores data, such as a spreadsheet or database. CSV files can be imported and exported to and from a program, such as Microsoft Excel.

Blacklist - A list of names of people that someone does not want to work with. This could be for many reasons, such as people who do not get along with one another. People are put on the blacklist are those that one may think of as unacceptable and wants to avoid for any given reason.

Preferred List - A list of names of people that one wants to work with. The people that someone puts on this list are those that they get along well with and may regard to them as colleagues or friends. Those put on the preferred list are people that are trustworthy.

Random Generation - A function that produces a random number. The number will be truly random meaning that it is chosen without method or conscious decisions. The random generation will pick a number finite number starting from one through a specific given number needed, for example, twenty. This random generation will be used to randomize groups.

Pandas - A Python library for working with data manipulation and analysis. This library includes many functionalities but is mainly used for data structures and operations used for manipulating numerical tables. In our software, Pandas will be used for working with our CSV file.

DataFrames - A Python data structure that stores data in tables separated by rows (records) and columns (fields). A DataFrame can be easily viewed and manipulated by a

user. In our software, DataFrames will be used to work with the parameters given by the user.

Command Prompt - A command prompt is a command line interpreter that is used to execute entered commands. For example, reading and writing to the file.

GUI - A GUI is a Graphical User Interface. A GUI is a type of interface that allows the user to interact with visual buttons, words, pictures, and images. A graphical user interface is used in most software rather than a text-based interface, this is because it is easier to read and comprehend.

4.2 Author Information

Igor Asipenka - iasipenk@mail.umw.edu

- Section 1
- Section 3

Denzel Saraka - dsaraka@mail.umw.edu

- Section 2
- Section 3

Anthony Phimmasone - aphimmas@mail.umw.edu

- Section 3, 3.4, & 3.5
- Section 4.1

James Donahue - cdonahue@mail.umw.edu

- Section 1.1
- Section 1.2
- Section 4.2