Project Documentation

**Goal**

The goal of our project is to create a program which will prompt the user to enter information about their assignments to be completed over the course of the semester. The program will prompt the user for information about these assignments such as the day in which they are due, the class that the assignment is for, and how long the user thinks they will spend working on the assignment. From here, the program will gather all of the assignments and save them to a file for the user to print or be able to save to their device and remain organized for the remainder of the semester. The user may also return to the program at any time to add additional assignments and to mark assignments are completed or delete an assignment which may no longer apply.

**Decisions Made**

We had to make several decisions about our code as we needed to decide what to do with the assignments once they had been inputted by the user. We made the decision that the best next step would be to save this information into a file which would be saved to the same directory on the user’s computer as the code is being run. This decision was made as the user can take several different routes once the information has been saved onto a file. They can either print or save the file onto their desktop, allowing them to access it frequently throughout the semester without having to run the program and enter in all of the assignments once again. In addition, we also had to decide if we wanted to sort the assignments by priority. Part of our code asks the user to determine whether each assignment is a homework, a project, studying, or a separate category. While we had certain priorities in mind when writing the code, we realized these priorities may not align with everyone and that the order in which the assignments would be sorted, further from the due date, would involve too much personal preference and would require a lot of unknown variables to be determined. Due to the large number of discrepancies in the sorting process, it was decided that it was best not to place priority over a specific category of assignments beyond the day that they are due.

**Challenges**

During the development process of the code, we faced challenges when it came time to storing the information into a database. Being able to complete this step would mean that the user would have access to the data they inputted when they first ran the program and be able to add additional assignments as they were assigned throughout the semester. After researching how to implement a database into a program involving C++ code we realized the process was a lot more involved than anticipated and required knowledge using SQL, which none of us previously had. Since the code we already had planned out was ambitious for two people given our time constraint, we decided to have the program save the input provided by the user into a file. This would allow the user to still have access to their information without having to repeat the initial process.

We also ran into an additional process when implementing the user interface, or UI. Ideally, we wanted to be able to read the file that was generated by our C++ code into the UI so that the user would not have to run the program through a terminal and perhaps the UI could be developed into an application or website in the future. This development would mean that students would have access to a list of assignments and due dates on their mobile phones or computers, where the majority of their schoolwork is stored. We also faced issues with the UI for similar reasons as the database as it required previously knowledge using C++ builder, as well as a functioning data base which stored the inputs from our program. Since we decided to not pursue the route which involved the database, we opted to create a simple user interface with hopes of developing it further in the future.

**Core Algorithms & Process**

The main algorithms we used within our C++ code were the main.cpp and the OrganizeLife.cpp file. Looking closely at the main.cpp file, it begins by getting today’s date and displaying it. This is necessary for our code as an error will be displayed if the user inputs the due date of an assignment as being before the present day. Moving forward with the code, a list of assignments is created, and the user is asked to input the number of assignments for that semester and that they would like to include in their schedule. The function OrganizeLife is later called which collects the assignment data and organizes this information into a list. The user is then prompted once again on what they would like to do with the information they have just provided regarding their assignments. As can be seen by out file, the user is given the option of deleting the assignment, marking it as complete, searching for the current assignment, which is show the user a series of possible criteria to choose from and will then display a list of all possible assignments which meet these criteria, or to quit the program. If the user chooses to quit the program before all assignments have been marked as completed, a message will be printed which asks the user if they are certain that they want to quit the program although there are still assignments left to be marked off as completed. Based on the input that the user gave to the previous series of questions, the code enters a switch statement which will start the process of preforming the task requested by the user. An example of this can be observed in case d as shown below.

Graphical user interface, text, application, email

Description automatically generated

This case will ask the user which assignment they would like to delete and then ask them to make this selection based off of the assignment number that has been assigned. It will then call the function “deleteAssignment” which is shown below and remove the assignment from the assignment list.

Text

Description automatically generated with medium confidence

A similar process is repeated for the other possible commands the user could have chosen at the beginning of the switch statement.