Project Plan

Nolan Cassidy, Andrew Evelhoch, Jake Gianola, Kevin Kincaid, Kylie Quan
Last edited: 4/29/2019
We have consistent times meeting in the science library already started friday & tuesday directed through a group chat. We will be breaking the project into equally sized parts and distributed among the group. After each member finishes their part, they will continue to help overlook and debug others' code.

Work Breakdown Schedule

1. Determine meeting schedule
2. Project Plan
3. SRS/SDS
4. Data Repo
    a. Import/Export student roster
    b. Maintain info between sessions
    c. Export reports
        i. Daily participation
        ii. Term participation
5. Interface
    a. Small gui
        i. Display 3 names
    b. Large gui
        i. Display 3 names
        ii. Phonetic Spelling
        iii. Nickname
    c. Help menu
6. Queue
    a. Populate from data repo
    b. Extract student from top 3
    c. Re-add extracted student to queue
    d. Randomize queue equally
7. User input/output
    a. 1 & Space, 2, 3 to remove students from front of queue
    b. Menu for additional options
8. Integrate subsystems
9. Testing and debug
10. Final Maintenance

Project Schedule:

One individual takes lead of each milestone.
Interface for app/help -Nolan Cassidy
Queue - Andy Evelhoch
Export/Keyboard Input - Kylie Quan
Data Repo - Kevin Kincaid
The individual will take charge but everyone will contribute and overlook the progress.

Organization

We have a google sheet excel document to keep track of each person's assignments and the current due dates. This is shared between everyone and is color coded with the status of completion. This creates a dynamic workflow to keep everyone on the same page and maintain a transparent environment. The spreadsheet will list who completed each section of code, and who tested and confirmed it.

Build Plan

To build the system we will divide the milestones between the group. Because our system is separated into subsystems that can function by themselves with dummy inputs, and their interactions are very well-defined, we can build the subsystems individually and then put them together afterwards.

The subsystems are: Interface, the queue, the data repo, and export functionality.

1. The first build will therefore be multiple builds, one for each subsystem.
2. Once the subsystems are working properly, the next build will be to connect one subsystem to another.
3. Builds beyond that will be to debug and refine the program until it is working perfectly.

Ultimately, everyone will look over the entire project, but each person will take charge of a single part initially.

The system has been broken down into these parts because they are the smallest parts of the project that can function separately with minimal dummied inputs from other parts of the program before integration. We decided to build these subsystems separately so that each team member can begin working on their subsystem simultaneously, in order to increase the initial rate at which the subsystems get developed. Integrating the subsystems together will be facilitated by the well-defined way that the subsystems interact.

One of the main risks of any program is making early structural decisions that can be hard to undo or change once the program has been further developed. We have attempted to mitigate this by breaking it up into modules that can work independently and can be entirely rewritten more easily than the whole program, if need be.