**CIS 422 Project 1:  
Ducks on Deck  
Project Plan**

Luying Cai (lc), Ellie Kobak (erk), Kalyn Koyanagi (kek), Liza Richards (ljr), and Kelly Schombert (ks)

January 8,2022 – v1.0

**Table of Contents**

[**1. Project Plan Revision History**](#_heading=h.30j0zll) **1**

[**2. Management Plan**](#_heading=h.1fob9te) **1**

[2.1. Organization and Roles](#_heading=h.3znysh7) 1

[2.2. Meetings and Communication](#_heading=h.2et92p0) 2

[**3. Work Breakdown Schedule**](#_heading=h.tyjcwt) **3**

[**4. Monitoring and Reporting**](#_heading=h.1t3h5sf) **3**

[**5. Build Plan**](#_heading=h.4d34og8) **4**

[5.1. Plan Details](#_heading=h.2s8eyo1) 4

[5.2. Rationale](#_heading=h.17dp8vu) 4

[**6. Acknowledgements**](#_heading=h.3rdcrjn) **5**

# 1. Project Plan Revision History

**Date Author Description**

**————— ——— ————————————————————————————**

1-8-2022 ks Created the initial document and wrote first draft to be proposed at second group meeting.

1-10-2022 ks Revised Organization and Roles section and Build plan section

# 2. Management Plan

## 2.1. Organization and Roles

Each group member is assigned the following role which includes the outlined responsibilities:

* Writing lead: Liza Richards
  + The writing lead will verify the quality and completion of the writing in project documents such as the SRS and SDS.
  + The writing lead will assign writing related work to other members while monitoring progress.
* Record keeping: Ellie Kobak
  + The record keeper will ensure that task assignment and completion is being recorded in a complete and timely fashion.
  + The record keeper will summarize any decisions made or issues brought up during group meetings.
  + The record keeper will make sure that any revisions to records or documents are properly noted and archived.
* GUI Lead: Kalyn Koyanagi
  + The GUI lead will monitor progress and verify completion of assigned implementation tasks pertaining to visual and user interface components
  + The GUI lead will routinely check the quality of the GUI code such as performance, style, maintainability etc.
  + The GUI lead will be referred to when any minor design decisions pertaining to visual and user interface components need to be made.
* Backend lead: Kelly Schombert
  + The backend lead will monitor progress and verify completion of assigned implementation tasks.
  + The backend lead will routinely check the quality of the backend code such as performance, style, maintainability etc.
  + The backend lead will be referred to when any minor design decisions pertaining to visual and user interface components need to be made.
* Test lead: Luying Cai
  + The test lead will develop a thorough and robust set of program test cases for the debugging stage. The test lead will assist in developing component test cases where needed.
  + The test lead will check throughout the development process that the currently implemented components are being robustly tested by group members.
  + The test lead will make sure that any issues or bugs that have been raised are corrected in a timely fashion and assign debugging tasks.

The type and number of responsibilities for each member will be flexible to accommodate any issues or imbalances in workload or unforeseen circumstances that may arise. Every member will participate in routine meetings and communicate with each other outside of meetings to make decisions, be assigned tasks, and track the progress as the project goes on.

When a large decision (any decision that may impact other parts of the development process) must be made, the relevant lead will decide on a course of action individually and then propose their verdict to the group. The group will discuss if there are any alternatives that may be preferred or if there are concerns with the decision. If there are no concerns or alternatives, the lead’s original decision is carried out. Otherwise, a group decision is made and carried out instead. Smaller decisions that will not impact other pieces to the development process will be left up to the appropriate lead.

## 2.2. Meetings and Communication

Group members will attend regular formal meetings at the following times:

* Sundays @ 4:30 pm
* Fridays @ 12:00 pm
* *Additional meetings will be scheduled as needed*

Meetings will take place either over Zoom or in the Knight Library depending on the health status of each individual. Each meeting will last 60-80 minutes depending on agenda items and as time permits for individual members.

Group members will attend regular informal check-ins after each CIS 422 class period.

Check-ins will take place in the Price Science Commons with whichever members are available and present for a check-in.

Group members will have discussions and report their progress outside of meetings via the following communication methods:

* Group SMS messaging
* Discord

Members are expected to read such messages to stay up to date on any updates or issues that might arise during development.

# 3. Work Breakdown Schedule

* Week 1 (1/5 – 1/11)
  + Create build plan
  + Create project plan
  + Create working drafts of the SRS and SDS
  + (Initial project documents due on Tuesday, Jan. 11)
* Week 2 (1/12 – 1/22)
  + Gather sample data
  + Build non-GUI components such as structures for holding student data, randomizing algorithms, and systems of reading to log files or from saved files
  + Build GUI components dealing with visual output and user interface
  + Combine all components to have an executable system ready for testing
* Week 3 (1/23 – 1/29)
  + Begin testing and debugging phase
  + If on or ahead of schedule, consider implementation of ‘nice to have’ requirements or improvement to required implementation
  + Review and update documentation
* Week 4
  + (Project due on Sunday, Jan. 30)

The responsibility for monitoring progress of each milestone will be in accordance with the roles outlined under the **Management Plan** section. Members will likely have roughly equal distribution of implementation tasks.

# 4. Monitoring and Reporting

Tasks that have a completion that can be objectively verified will be recorded to a shared spreadsheet. This spreadsheet will also note the group member in charge of completing this task and by what date the task should be completed by. The spreadsheet will be frequently updated to show every self-reported task completion and verification by another group member. Ideally, verification will be done by the relevant lead member or the record keeper.

# 5. Build Plan

## 5.1. Plan Details

As further outlined in the SDS, the program will consist of 5 components implemented through 5 files. These files are: main.py, buildQueue.py, fileReader.py, fileWriter.py, and visual.py.

The first files to write are fileReader.py and buildQueue.py. fileReader.py reads an imported tab-delimited file containing the class roster from the instructor and stores any student data necessary for the remaining program execution. buildQueue.py uses the class data obtained by fileReader.py to create a randomized queue of students that determines which students will be “on deck”.

After the components concerned with the organization and storing of data are implemented, then work begins on the visual, user interface, and data writing components. These components are written through the visual.py and fileWriter.py files. visual.py handles the display of “on-deck” names as well as the key controls used by the instructor. fileWriter.py responds to key controls from visual.py and writes the relevant student data to log files on the system.

Once each component can execute its individual functionalities and have been tested for appropriate requirements, the final component to link all other components together will be written. This is implemented through the main.py file. This file links the components concerned with receiving and organizing data to the visual and data writing components that need information stored in the backend.

Afterward reaching this buildable state, any remaining ‘must-have’ requirements will be implemented and the possibility of implementing remaining ‘should-have’ or ‘nice-to-have’ requirements will be evaluated.

All resources should be focusing on testing and debugging to reach a stable release candidate by the project deadline unless optional requirements and functionality are being added.

## 5.2. Rationale

The rationale behind breaking down the system into five components is to separate the most prominent “must-have” requirements into manageable pieces. When the components are separated, the group is also able to ensure that the pieces work individually before attempting to combine the system. This modularity will also give the best environment to attempt incorporating “nice-to-have” requirements.

Development of the backend (non-GUI) and GUI components have been separated to allow group members to work in areas that within their expertise or that they find interesting.

We do expect team members to have more difficulty in creating and integrating the GUI elements as no members have had experience with GUIs. We will have to assess the difficulties that arise with implementing the visual and user interface components early on and reallocate team members accordingly. In addition, even members who are not part of the initial development of GUI components should be required to make themselves familiar with the relevant libraries.

# 6. Acknowledgements

The formatting of this document was based on a Software Requirement Specification template provided by Professor Anthony Hornof.