Software Requirement Specification (SRS)
Classroom Cold-Call Assist Software Program

Z. Carroll (zc) - 01-28-2022

K. Nguyen (kn) - 01-28-2022

N. Onofrei (nof) - 01-28-2022

L. Vandecasteele (lv) - 01-28-2022

H. Zhang (hz) - 01-28-2022

Table of Contents

1.System Summary

The purpose of this system is to give teachers a platform to help them organize the next set of students to be called on during lecture aka "Cold Calling". The system will be able to take a roster of all the students in the class and throughout the course of the system's use, each student will be selected randomly and evenly, meaning no student will be picked on more than the next student. The system is designed to work for as long as the teacher needs it. All data of each student will be tracked and kept in a file for future use.

The system will be a small window on the foreground of the desktop at all times. It is small in order to not distract from the course materials that the teacher will be giving. The system displays a list of 4 students currently "on deck", meaning the next 4 students that will be cold called. The intention of this is to let the students know beforehand so that they may warm up for the oncoming conversation.

The system includes a "flagging" system which allows the professor to mark a specific student on deck in the case in which they would like to have a personal conversation with the student.

As for the system's input, it uses the keyboard's number keys 1-4 to remove or flag students that are currently on deck.

2.Basis for the system's purpose

Sometimes it is hard for student's to stay focused in class due to a lack of concentration, then they tend to doze off and not actually learn from the presented material. This system was intended to keep the students more alert and focused during class. Having an equally distributed cold call system in class will force the students to be more alert and provide more attention to learning. By the end of class, students will have participated in an engaging environment.

3.Use Cases

Usage for this program will proceed as follows:

Case 1:

Preconditions: A teacher wants to give an extremely important course lecture through a powerpoint and wants the students to be more engaged. The teacher is going to "cold call" each of the students equally for discussion.

1. The teacher loads up the powerpoint slide as well as the cold call assistant program which has a minimal window size.
2. The teacher then imports a "class roster" text file of their current class.
3. The program is dragged to the corner of the screen so it is out of the powerpoints way but still visible
4. The program currently has 4 students (Students A-D in order) "on deck" to be chosen for discussion. The 4 students were randomly chosen to be on deck.
5. During lecture, the teacher refers to the list on display and cold calls student C. Student C answers the teacher's question.
6. The teacher clicks on the System's window in order to use the specific keyboard inputs for it.
7. The teacher presses the keyboard input "3" which removes student C from the deck.
    a. We have decided to use numeric keyboard keys 1-4 to make changes to the students on deck because it is very straight forward.
    b. Each of the number keys refer to the positions of each student on deck
        i. Example: 1 - Student A, 2 - Student B, 3 - Student C, 4 - Student D
8. A new student is added to the end of the deck which moves Student D's position to Student C's old position. The new student then takes the place of student D's old position.
    a. Before: Student A, Student B, Student C, Student D
    b. After: Student A, Student B, Student D, New Student
9. The teacher clicks back on the powerpoint presentation and resumes her lecture.
10. After class is over, the teacher generates the summary file by clicking the "Cold Call Assist" button again. The teacher reviews who they had cold called and makes personal notes on each student.

Postconditions: The system can be shut down by closing the window. The system will be available for use again on the next start up.

Case 2:

Preconditions: The teacher is preparing in the morning before class starts at noon. She realizes that her keyboard buttons 1,2,3,4 are all sticky and unusable. These were the buttons that are

used to remove a student from on deck after the teacher calls on the student. She needs to change the mapping of the keys so that she can use the remove function.

1. The teacher locates the directory where the Cold Call system is.
2. She opens up the system files and opens up the source code “ColdCallOperation.py” file.
3. Near the top of the source code there exists 4 variables named “remove1”, “remove2”, “remove3”, “remove4”. Which are mapped to the keys 1,2,3,4 respectively.
4. She changes the source code by changing the variable to her desired keyboard button.
    a. Remove1 = ‘5’
    b. Remove2 = ’6’
    c. Remove3 = ’7’
    d. Remove4 = ’8’
    e. This effectively changes the remove buttons to the keys 5,6,7,8 on her keyboard
    f. The teacher saves the source code and closes it
5. The teacher starts the program with her class roster and tests the new changes
    a. On deck: Student A, Student B, Student C, Student D
6. She presses ‘5’ which removes Student A from the list and shifts everyone to the left and adding a new student to the deck
    a. Student B, Student C, Student D, New Student A
7. She presses ‘5’ which removes Student B from the list and shifts everyone to the left and adding a new student to the deck
    a. Student C, Student D, New Student A, New Student B
8. She presses ‘5’ which removes Student C from the list and shifts everyone to the left and adding a new student to the deck
    a. Student D, New Student A, New Student B, New Student C
9. She presses ‘5’ which removes Student D from the list and shifts everyone to the left and adding a new student to the deck
    a. New Student A, New Student B, New Student C, New Student D
10. She observes that the changes are working properly and close the program

Postconditions: The key mappings for the system have been changed until she can fix her keyboard. The new keys will stay like this until the next time when someone changes the key mappings again.

Case 3:

Preconditions: A new exchange student has joined Professor H’s class and the professor now needs to add the student into the class roster for the cold call system.

1. The professor pulls up information about the student which includes, their full name, Duck ID, reveal code, email address, and a phonetic spelling of their name.
2. The professor then accesses his student roster text file and manually enters each piece of the students information into the general format as a new line.
    a. Format: <First Name> <tab> <Last Name> <tab> <UO ID> <tab> <email_address> <tab> <Phonetic Spelling of first name> <tab> <Reveal Code>
3. The professor saves the text file, and opens up the cold call system via terminal.

4. The professor clicks on the “Import New File” button and imports his student roster with the new student added.
    a. This new import will overwrite the CurrentRoster.txt once the professor clicks the Cold Call Assist button.
5. The professor then clicks on the “Cold Call Assist” button and sees the student’s on deck
    a. The professor removes the students from on deck 1 by 1 until they see the new exchange student's name.
6. After seeing the new student’s name on the deck, the professor closes the program and knows that the student has been successfully added.

Postconditions: The new exchange student has been added to the roster and will stay there until the next roster change. The next time the professor uses the cold call system he will be able to call on the new student if they appear on the on deck list.

4.Function and Non-Functional Requirements

A. Instructor-System Interactions (Key Differences)

- ***Call on Students and remove them from the “on deck” list (Functional)***
    - We used the numeric key input because it is very straightforward. Each keyboard input “1-4” refers to the student's current position from the “on deck” list.
    - The display shows the current 4 students that are on deck
        - [Kenny Park, Luke Park, Nick Park, Zacree Park], each student refers to the following positions [1, 2 ,3, 4]
    - Assume the professor cold called Kenny and now he is free. The professor would then press 1 to remove Kenny Park from the on deck list and the rest of the students would shift their positions to to the left. The next student to be added on deck is randomly chosen.
- ***PowerPoint is the active application but the Cold Call window is the foreground window (Functional)***
    - Our cold call will be in the foreground at all times but in order to not override the button configurations of other applications, the cold call inputs will only work if you click on the Cold Call window to activate it.
- ***Display (Non-Functional)***
    - Our display’s main window is the “Model Select” window. On this page, there are 3 options.
        - Cold Call Assist - Shows the expected window display of the list of current students on deck and a back button to traverse to the Model Select page
        - Import New File -
        - ●.○.■.1. If there is no student roster file imported into the program, the teacher will have to import a previously made student roster file in order to start the Cold Call Assist.
        - ●.○.■.2. If there already is a student roster file linked to the program a warning is given to let the user know that they are going to change the linked student roster file to the new one given.
        - Select Location to Export Student File - This button allows the user to export their current student file to a specified directory of their choosing.

- ***Roster Input and Output (Functional)***
    - Our system uses buttons to export and import student roster files.
        - **Exporting (Functional)** - This allows the user to save their current student roster file to any directory that the user desires.
        - **Importing (Functional)** - If the user imports an improper student roster file, an error window will pop up. The window explains the reason why the imported file isn't in the proper format. As a side effect the imported file will not be read into the system.
- ***Change-related requirements (Non-Functional)***
    - We decided not to go with the option to switch the student roster file delimiter from spaces to commas or vice versa. This program is meant to be a stand alone program and only works with tab delimited files since that is the convention that we chose for sake of simplicity.
    - **Key Mapping Changes** - In order to change the mapping of keys, the user would need to directly access the ColdCallOpertation.py source file and change it manually. The current key inputs are located at the top of the source file. However the user can change the keys to whatever key they desire.

B. Data Constraints

- ***The ordering of Students in the Queue (Functional)***
    - Our system randomizes the ordering of students along with a priority queue as the main data structure.
    - So the way it chooses a student: We have a priority queue where students with the fewest number of cold calls are given the highest priority, thus ensuring that cold calls are distributed equally amongst all the students. Furthermore, when students are inserted into the priority queue they are chosen at random using the python builtin random.choice() function. Therefore, even when students have been cold-called the same number of times, (0 times to start) they are randomly arranged such that each time the system is run there will be a different list of students displayed
    - Priority queue works -
        - The system's priority queue places a higher priority on students with the fewest number of cold calls. This allows the system to distribute cold calls evenly throughout all students.
        - Students are chosen randomly to be placed onto the on deck list but higher priority students will get chosen first over low priority students.
        - Assuming that every student has been cold called the same number of times then they all have equal chance to be chosen randomly.
            - Once the system starts with a fresh student roster file, the queue will have a randomly decided arrangement of students.
    - Only the full name of each student can be seen through the display.
- ***Realistic Sample Data (Functional)***
    - Our system has a testing file named "Testing.py" which is able to generate a list of up to 100 unique students.

        - The student generation method will generate n students and write to a file named “StudentFile.txt”.
            - Each student inside the file will have the format: <First Name> <tab> <Last Name> <tab> <UO ID> <tab> <email_address> <tab> <Phonetic Spelling of first name> <tab> <Reveal Code>
- ***The user names will reside in a roster file with the following format: (Functional)***
    - <First Name> <tab> <Last Name> <tab> <UO ID> <tab> <email_address> <tab> <Phonetic Spelling of first name> <tab> <Reveal Code>
        - The UO ID is 9 digits long and the Reveal Code is 7 digits long. Both the UO ID and Reveal Code are unique, meaning that no student will share the same id or code as another student.
        - The format of each email address is the first 3 letters of a students first name combined with their full last name attached to a “uoregon.edu”.
        - Our file to be written and read does not have the “first line of the file” which in Hornof’s version of the Cold Call System is “a comment that is not modified by the system, both when reading from and writing to the file” as this comment is not necessary to our system.
- ***Random Distribution Verification Mode (Functional)***
    - The testing system generates a list of 100 unique students using the student generator method. After that it randomly places 4 students into a queue. Students are chosen randomly and removed from the queue. They are then written to a file named “testoutputfile.txt”. As each student is being read they have a 25% chance of being flagged and if they are flagged their line will be labeled with an “X” within the testoutputfile.txt.
        - This way of testing will allow the user to see that the students are randomly chosen for the queue and shows how the flagging will look inside the summary output file.

C. System Startup and Shutdown

- ***System Startup (Functional)***
    - To start up the system, the user must open up the terminal and go to the system’s location inside the directory. To turn on the program, enter the command “python3 ColdCallOperation.py” into the terminal.
    - During Startup
        - The system should start up without any faults and get to the “Model Select” page within a few seconds.
        - The system could be linked to a current roster file or be imported to a new roster file. If “CurrentRoster.txt” exists in the same directory as the system then this would be the linked roster file.
        - If the system is currently linked to a proper roster file then the “Cold Call Assist” option should work without issue. The program could be closed and started up at different times but still use the same data.
        - If the system is not currently linked to a roster file, it will display the error warning “Error: No Existing Student list” and the user will be prompted

with the message “Please Import Student List” which will return the user back to the “Model Select” Page.
        - **Summary File (Functional)** - A summary file of the previous Cold Call session will be written to a summary file with the current date as the file name. The summary file will contain all of the students that were cold called including flagged students with the format: <X> (if the student was flagged) <First Name> <Last Name> <email_address>
- ***System Shutdown (Functional)***
    - Closing the system window will shut down the program. The data will be saved into a “CurrentRoster.txt” which is also the currently linked student roster file.

D. Optional Secondary System: A Photo-Based Name-Learning System (Functional)

We chose to not include this secondary system because we felt the photos will make the display window and roster files more complex and hard to read. Keeping the secondary system away makes the original system feel more slick and simple like a single use tool.

5. Build Related Constraints

- ***Target Platform***
    - The system will comfortably run in Macintosh OSX 10.13 (High Sierra) or 10.14 (Mojave).
- ***System Document File Formats***
    - All system-related and system-development-related documents that are intended for human reading must always be in plain text.
- ***Programming Constraints***
    - The system was built using python 3 and tkinter along with python standard libraries.
- ***Installation***
    - Download the file through the github repository
    - Place the file in the desired directory
    - Run the program through the terminal
    - (Check “Installation Guide.pdf” in Documentation folder for more in depth detail)

6. References

Hornof, Anthony, (2022, Jan). *Project 1 - Develop a Classroom Cold-Call Assist Software Program*
https://classes.cs.uoregon.edu/22W/cis422/Handouts/Cold_Call_System_SRS.pdf

Group 7 CIS422, (2022, Jan). *InstallationGuide.pdf*
Located in Documentation folder