CoolCall: A Cold-Call Assist Program Software Requirements Specification

Arden Butterfield, Quinn Fetrow, Derek Martin, Amy Reichhold, Madison Werries

January 30, 2022

Table of Contents

1. SRS Revision History **2**

2. The Concept of Operations (ConOps) **2**

2.1. Current System or Situation 2

2.2. Justification for a New System 2

2.3. Operational Features of the Proposed System 3

2.4. User Classes 3

2.5. Modes of Operation 3

2.6. Operational Scenarios 4

3. Specific Requirements **7**

3.1 Functional Requirements 7

[FR] Absolutely required: 7

[FR] Not absolutely required: 8

[FR] Future considerations: 10

[FR] Specifications omitted: 10

3.2 Non-Functional Requirements 10

[NFR] Absolutely required: 10

[NFR] Not absolutely required: 10

[NFR] Future considerations: 10

[NFR] Specifications omitted: 10

4. References **11**

5. Appendix A: Alternatives Considered **11**

6. Appendix B: Requirements Omitted **11**

The structure and format of this document was derived from and inspired by the Software Requirements Specification Template noted in the References section of this document, which was created by A. Hornof, 2019

1. SRS Revision History

Date	Author	Description
1-9-2022	Madison	Created the initial document from A. Hornof’s template.
1-28-2022	Madison	Edited section 2.6, *Operational Scenarios*
1-28-2022	Arden	Edited section 2, *ConOps*
1-29-2022	Derek	Edited Appendix B, *Requirements Omitted*
1-29-2022	Derek	Edited section 3.2, *Non-Functional Requirements*

2. The Concept of Operations (ConOps)

2.1. Current System or Situation

Currently, an instructor wishing to cold call students in their classroom has to do it manually. For instance, an instructor may print out the class roster on a sheet of paper, and put a tick mark next to students when they are placed on deck. When a student is called on, the instructor can record that in some way on the piece of paper. The instructor can verbally let the students know who is on deck, and give an update to the class when changes are made to the on-deck list.

In the current situation, it is relatively easy to handle students who are absent. The instructor can just skip over the absent student’s name on the class roster.

To ensure that the current system is randomized, the instructor can start each class with a different group of students on deck. This can be verified by checking previous on deck lists.

To flag students, the instructor may write notes on the paper next to the names of students who they would like to follow up with later.

2.2. Justification for a New System

The existing cold-calling system has some shortcomings. Manual cold-calling requires the instructor to print out rosters, decide who to place on deck each time, and remember who has been on deck recently if they want to ensure that they are calling on students equitably. Under the current system, creating a term-summary would require the instructor to sort through all of the printed rosters they have used throughout the term, compiling the data by hand.

Another shortcoming is that students must remember that they are on deck, since there is no visual feedback of who is currently on deck. A student might easily stay on deck for the entirety of a class period, and must remember that they are on deck while simultaneously keeping track of the lecture.

The CoolCall system will provide a platform to achieve the same functionality while making choices like who to place on deck for the instructor automatically, improving both ease-of-use for the instructor and fairness for the students. It will also enable both the students and the instructor to easily see who is on-deck students, reducing the risk that a student is caught off guard by being cold-called since the names will be visible to everyone.

2.3. Operational Features of the Proposed System

The CoolCall system is a proposed solution to this problem. It lets the instructor import a roster file which is stored internally. Before each class, the instructor can open up CoolCall and a window will show the students who are currently on deck. The instructor can call on students from the window, which will cause it to add another student to on deck replacing the student who was called on. The instructor can "flag" a student when calling on them if the student asked a particularly good question and the instructor wants to follow up with that student after class. The instructor can also review the students who have been called throughout the term.

The system should call on all students at an approximately even rate. For instance, it should not be biased towards students at the top of the roster.

Since this technology will be used in front of an audience, it is of utmost importance that it does not crash, lag, or waste valuable lecture time. The system should start up easily at the beginning of each class. It is imperative that the system display is visible without distracting the students in the class. It should sit atop lecture materials on the screen without consuming too much screen real estate.

2.4. User Classes

A. Instructors

The primary users of the CoolCall software are professors who present PowerPoint slides during their lectures and who would like to implement fair, easy-to-use cold-calling in their classroom. These instructors already have basic computer skills such as opening an application or reading a file, and their screen is already shown during class time. However, these users may not know any programming languages or how to use the command line. Therefore, they expect the application to run after double-clicking an icon.

B. Students

Students want to know when they are on deck so that they have time to prepare. Students also want to ensure that the instructor is calling on students in a fair manner, especially if the instructor is using a new and unfamiliar piece of software to call on students.

2.5. Modes of Operation

This system has three modes of operation.

1. In **normal usage**, the system displays student names to the class, and the instructor calls on students from the display.
2. In the **file import/export mode**, the instructor either imports a roster into the system, or exports the roster stored within the system. Students are likely not present when the instructor uses this mode.
3. In **Random Distribution Verification Mode**, the instructor wishes to test if the system is calling on students equitably. Students will likely not be present when the instructor uses this mode.

2.6. Operational Scenarios

Use Case: First-time classroom use

Brief Description: This use case describes how an instructor might set up and use the CoolCall application for the first time in a classroom setting.

Actors: An instructor.

Preconditions: An instructor has already installed the CoolCall application onto their Mac computer. The instructor has not loaded their tab-delimited roster into the application before, but the formatted roster is ready for use. They plan to load the roster and then immediately begin using the system to cold-call students during the day's lecture.

Steps to Complete the Task:

1. The instructor arrives a few minutes early to set up before the lecture. They open their laptop, and double-check the location on their computer where they have saved the formatted roster on their computer.
2. The instructor opens a Terminal window on their Mac, and runs the command to start the CoolCall software.
3. The instructor sees a popup window indicating the system did not find a previously loaded roster file, and the instructor is then prompted by the system to select a new roster file to load.
4. After the instructor navigates to the folder where their tab-delimited roster txt file is located, they load the roster into the system by either selecting the desired file and hitting "OK", or double-clicking on the filename.
5. The system notifies the instructor with a list of names that will be loaded into the system based on the roster file. The instructor reviews the list, confirms it looks correct, and hits "OK".
6. The CoolCall Application starts up ready to be used by the instructor, showing a window with the names of the 4 students who are currently on deck.
7. The instructor opens their PowerPoint lecture presentation slides and hits "Play from start" in the PowerPoint application. They notice that as the presentation opens, the CoolCall application stays on top of the slides as desired.
8. They then drag the CoolCall application window to the location on the screen where they would like it to remain for the duration of the lecture.
9. The instructor connects their computer to the projector, greeting students as they arrive.
10. The instructor begins the lecture, making sure that they have clicked on the PowerPoint presentation window to select it as the active application so the PowerPoint slides may be navigated using arrow key input.
11. After lecturing for a few minutes, the instructor pauses to ask the class a question about the material. An on-deck student raises their hand to answer the question.
12. After the instructor clicks on the CoolCall application window to switch arrow key input over to the CoolCall application, they call on the student who has their hand raised.
13. The instructor removes the student from the CoolCall on-deck list using the following operations:

   a. They select the students name on the list by pressing the left and right arrow keys on their keyboard.
   b. They press the down arrow key on their keyboard to remove the student from the on-deck list without flagging them in the system.
   c. Or, they press the up arrow key on their keyboard to remove the student from the on-deck list, “raising” a flag for the student’s call instance.
14. The instructor continues in this manner, using the CoolCall application to call on students until the end of class.
15. After class ends, the instructor closes the CoolCall application by hitting the “x” button on the window.

Postconditions: The CoolCall application is no longer running, but the roster file that the instructor loaded into the system has been saved as a persistent internal data file in the CoolCall system. The previous ordering of the on-deck queue has been saved internally to be re-loaded the next time the CoolCall application is run. A daily log file has been saved, and the semester summary file has been created and saved.

Use Case: Verifying the randomness of the CoolCall queue system

Brief Description: To ensure that cold-calling is being done fairly, the instructor wants to verify that the application places students on-deck in a way that treats students equally. Meaning, the system does not repeatedly place students who were on deck back on deck right away.

Actors: An instructor.

Preconditions: The instructor is familiar with how to run and use the CoolCall application, and they have loaded a student roster file into the system already. They would like to use the Random Distribution Verification Mode to examine whether or not the CoolCall system’s method of queue-insertion after removing students from on-deck seems fair.

Steps to Complete the Task:

1. The instructor opens a terminal, issuing the command to start the CoolCall application. They note that the application starts immediately and does not prompt them to load a roster file, as the system defaults to loading the internally stored roster and queue data when such data is found.
2. After verifying that the CoolCall application window is selected, the instructor hits the left arrow key 10 times in a row to prompt the system to enter Random Distribution Verification (RDV) Mode.
3. The instructor sees a window from the CoolCall application verifying that the instructor would like to enter RDV Mode, noting that any previously created RDV test data will be overwritten by the system.
4. The instructor verifies they no longer need any old RDV test data that might exist, and hits “OK” beginning CoolCall Random Distribution Verification Mode.
5. The program enters RDV Mode, which starts the program 100 times, cold-calling 100 students randomly each time.
6. The instructor sees a popup box indicating that RDV has terminated successfully, and hits “OK,” wherein the program returns to normal CoolCall mode.
7. The instructor closes the CoolCall program.

8. The instructor would like to see the data generated by RDV mode, and so they navigate to the program folder called "logs".
9. The instructor opens the file "random_distribution_verification.txt" and sees a file header indicating that this is a test of the Random Distribution Verification Mode. The date it was run appears underneath this on the next line. On each following line, the names of the 10,000 students called during the 100 runs is listed.
10. The instructor scans through the data, seeing that students were never called on twice in a row.
11. The instructor would like to see how many times each student was called on, and so they open the file "RDV_summary.txt." They see that each line in the file is of the form **<student name> <number of times called>**.
12. They notice that some of the students have a "0" next to their name, and recognize this data aligns with the reveal codes in their roster file for students who should not be placed on deck.

Postconditions: The RDV summary data is stored in the logs folder in the CoolCall program directory to be viewed as necessary in the future.

Use Case: Changing roster data

Brief Description: The instructor would like to change the contents of the student roster that is currently loaded into the system.

Actors: An instructor.

Preconditions: The instructor has used the CoolCall system before, and there is already a student roster loaded into the system. A few days into the term, several students have dropped the course. The instructor would like to update the roster in the CoolCall system to accurately reflect the roster changes.

Steps to Complete the Task:

1. The instructor opens a terminal, issuing the command to start the CoolCall application.
2. The instructor sees two buttons at the right side of the application window labeled "Import roster" and "Export roster."
3. First, the instructor would like to export the current roster which is saved in the CoolCall system. They hit "Export roster," seeing that a window pops up which enables them to choose where to save the roster.txt file. They choose a location on their computer, and hit "OK".
4. The instructor sees a notification popup window confirming that the file was successfully saved to the selected location.
5. After opening the saved roster file, the instructor simply deletes the lines containing the data of the students who are no longer in the course. They save the txt file with a new updated name so they know which file is which.
6. After returning to the CoolCall application window which is still running, the instructor clicks the button labeled "Import roster," selecting the newly saved and updated roster file.
7. The instructor sees a popup window indicating the names of the students whose data will be modified after loading in the new roster. The instructor reads the list, realizing they made a mistake and accidentally deleted a student's name who is still in the class.

8. The instructor repeats steps 3-6, ensuring that they delete only the correct names this time. After proceeding with step 7 again, the instructor verifies that the list of names to be modified is now correct.
9. The instructor hits "OK."

Postconditions: The internal student roster and queue data in the CoolCall system have been updated to reflect the current roster. The system is still running and awaiting further input.

3. Specific Requirements

The Software Requirements for the CoolCall Program are described here. Section 3.1 describes the functional requirements of the software, and Section 3.2 describes the non-functional software requirements. *Appendix A includes any alternatives which were considered in the development of these specifications.*

The functional and non-functional requirements have been further divided into four distinct categories based on their hierarchical importance:

1. ***Absolutely required:*** These are the core software requirements which must be met in order for the CoolCall system to function as expected.
2. ***Not absolutely required:*** These are the requirements which are essential but not necessarily vital for the CoolCall system to function.
3. ***Future considerations:*** These are the software requirements which have not yet been included as part of the CoolCall system SRS, but could be considered in future updates.
4. ***Requirements omitted:*** These are the requirements which were described in the original SRS (Hornof, 2022), but not included in the current SRS. Appendix B includes the rationale behind these omissions.

3.1 Functional Requirements

[FR] ***Absolutely required:***

1. Display Requirements
    - 1.1 The names of the students who are currently on-deck in the queue must be displayed at all times, and this list must be updated live as students are added and dropped from the queue.
    - 1.2 The name of the currently selected student must be indicated through a simple highlight effect.
    - 1.3 The system must provide visible buttons which allow the instructor to import and export student roster data files.
2. User input
    - 2.1 Pressing the left and right arrow keys must enable the user to tab through the list of students who are currently on deck. *(Appendix A2)*

2.2 Pressing the down arrow key lets the user remove the name of the student from the "on deck" list without raising a flag for that student.
2.3 Pressing the up arrow key allows the user to "raise a flag" as a student is removed from "on deck."

3. Student queue functionality
    3.1 When a student is removed from the queue, the students who are on deck must shift to the left (ie, forward in the queue) to accommodate this change.
    3.2 When a student is removed from the queue, a new student must be added to the list of on-deck students.
    3.3 Students will not be immediately placed back in the queue after being removed from the on-deck list (except in the rare case that a roster is so small that all students must always be on deck).
    3.4 Students will be randomly re-inserted into the queue after they are removed to ensure fairness, without violating requirement 3.3.
4. Data modification
    4.1 Instructor users will be able to import properly formatted rosters of students to be read into the system. *(Appendix A1)*
    4.2 Instructor users will be able to change to a new roster after a roster has already been loaded into the system.

[FR] ***Not absolutely required:***

1. Persistent data
    1.1 The system will store previously loaded student roster text files internally to be loaded into the CoolCall software again on startup.
    1.2 The student queue, including the ordering, will be maintained through internally saved files between runs of the program.
    1.3 The daily log data will be updated every time a student is removed from the on-deck list so that, in the event of an unexpected crash or shutdown, the system will maintain the most up-to-date log information possible.
    1.4 The internal queue file will be updated every time a student is removed from the on-deck list so that, in the event of an unexpected crash or shutdown, the system will maintain the most up-to-date log information possible.
2. Error handling
    2.1 The importing of incorrectly formatted roster files will be reported to the user using specific, directed error messages.
3. Daily log summary data
    3.1 For each run of the program, a daily log file will be saved containing information about the day's cold-calls.
    3.2 The daily log file will begin with a line indicating that it is a daily log file for the CoolCall software, followed by the date, followed by lines reporting the cold-call instances in the following format:

<flag_code> <tab> <first name> <last name> <email address>
Each item on the line is tab-separated. The <response_code> element is blank if there was no flag, and "X" if there was.

3.3 The daily log file will be written to every time a student is called on, so that in the event of a sudden system crash or other error, the daily log data is as up-to-date as possible.

4. End-of-term summary data

    4.1 A summary log file will be saved by the CoolCall system to report summary data about how often each student in the roster has been called on throughout the duration of the course.

    4.2 The summary log file will begin with a line indicating that it is the Summary Performance File for the CoolCall software, followed by a table header indicating the list of the fields to follow. The rest of the file will contain lines in the following format:
    <**total times called> <total times flagged> <first name> <last name> <UO ID> <email address> <phonetic spelling> <reveal code> <dates called>**
    The fields here will be tab-delimited.

    5.3 The summary log file will be written to every time a student is called on, so that in the event of a sudden system crash or other error, the summary log data is as up-to-date as possible.

5. Random distribution verification

    5.1 The system will implement a mode for generating data about the randomness of the student queue system.

    5.2 The Random Distribution Verification (RDV) Mode will be entered by hitting the left arrow key 10 times.

    5.3 RDV Mode will restart the program 100 times with the currently loaded roster file, randomly calling on on-deck students 100 times each run.

    5.4 RDV Mode will create a file called "random_distribution_verification.txt" which contains the in-order list of all 10,000 student cold-calls during the test.

    5.5 RDV Mode will create a summary of the 10,000 cold-calls in a file called "RDV_summary.txt" which has a header indicating that it is a summary file of the RDV Mode, followed by tab-delimited lines of the following format:
    <student name (first and last)> <number of times called>

[FR] ***Future considerations:***

1. Accept alternative forms of input

    1.1 Support numeric key input instead of arrow key input for moving through the names in the queue.

    1.2 Support input from non-keyboard hardware such as a clicker or remote.

2. Photo roster

  2.1 The system will support a view for displaying the students' photos next to their names, viewable only to the instructor.
  2.2 A "name-to-photo" training mode will be created to allow the instructor to better remember students' names.
3. Support for multiple courses run by a single user
  3.1 Allow the instructor to store persistent roster and queue data for multiple courses in a single term.
4. Ability to mark a student who is absent *(Appendix A3)*

[FR] ***Specifications omitted:***
1. The CoolCall program will not be able to accept keyboard input even while running as a background process. *(Appendix B2)*

3.2 Non-Functional Requirements

[NFR] ***Absolutely required:***
1. Keep numerical parameters and keystroke assignments at the top of a source code file
2. Do not lose or overwrite data, no matter when the instructor quits the program
3. The system should continue working even if it is unable to read or write from the disk
4. When re-inserting a student into the queue, avoid n% of the possible positions at the start
5. Output log files as text files
6. The system must run on Macintosh OSX 10.13 (High Sierra) or 10.14 (Mojave)

[NFR] ***Not absolutely required:***
1. The system will run using Python (3.7 through 3.10) and the Python Standard Library
2. Do not use student 95 numbers as dictionary keys

[NFR] ***Future considerations:***
1. Give the system admin an option of creating a new RDV log file instead of overwriting the previous RDV log file.

[NFR] ***Specifications omitted:***
1. The CoolCall program will not be started by double-clicking on an application icon.

4. References

Hornof, A. (2019). CIS 422 Software Requirements Specification Template. Available at: https://classes.cs.uoregon.edu/22W/cis422/Handouts/Template-SRS.pdf

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

5. Appendix A: Alternatives Considered

This appendix details alternatives which were considered during the development of the CoolCall Software Requirements. The Software Requirements of the CoolCall application were based on the specifications found in the document titled "Project 1 - Develop a Classroom Cold-Call Assist Software Program" (Hornof, 2022).

Appendix A1: *Method of import/export*

We considered how we wanted the user to be able to import and export a file. After considering sequences of keystrokes, we decided that on screen buttons would be the easiest to use.

Appendix A2: *Method of user input*

We considered using numerical input, but since arrow key import was preferred and easy enough to implement, we decided to use arrow key input.

Appendix A3: *Consideration of "Marking Absences" use case*

Although not discussed in "Project 1 - Develop a Classroom Cold-Call Assist Software Program" (Hornof, 2022), one important potential feature that was considered was a way for an instructor to mark a student who is absent. In the current system, the only way to get a student off of the queue is by calling on them. We discussed adding a functionality to remove the student from the queue without calling on them, which would be an important function, especially during a pandemic when attendance is spotty. However, in the interest of time and keeping the system simple, we decided not to implement this feature, but to consider it, if we were ever to make "CoolCall 2.0".

6. Appendix B: Requirements Omitted

This appendix details certain requirements which were originally described in the document titled "Develop a Classroom Cold-Call Assist Software Program," but were not implemented in the current version of the CoolCall Program (Hornof, 2022).

Appendix B1: Image display

The Cold-Call Assist SRS (Hornof, 2022) described the potential of showing images of students alongside their names in the on-deck display. This was considered, but the decision was made that this would take up too much space and be unnecessary.

Appendix B2: Responding to keyboard input in the background

The Cold-Call Assist SRS (Hornof, 2022) also described a requirement of listening and responding to keyboard input while the application is running in the background. It was decided that this requirement would conflict with the other implicit requirement that the system not interfere with normal operation of the PowerPoint slides. Since pressing arrow keys would also advance the slides when the slideshow was the active window, this requirement was not pursued.