CIS 422 Project 1:
Ducks on Deck
Software Requirement Specification

Kalyn Koyanagi (kek), Ellie Kobak (erk), Kelly Schombert (ks), Liza Richards (ljr) - 1-30-2022 - v2.0

Table of Contents

1. SRS Revision History

Date	Author	Description
1-8-2022	ljr	Created the initial document and began writing a portion of the first draft to be shared at the second meeting. Revised all “ConOps” and “Specific Requirements” sections.
1-9-2022	ljr	Updated section 2.2, 2.4, 2.6, 3.3, 3.5, and 5
1-9-2022	lc	Updated section 2.3, brief description
1-10-2022	ljr	Updated section 3.2.
1-10-2022	lc	Updated section 2.5.
1-11-2022	ljr	Made final adjustments to sections for initial project plan
1-24-2022	ljr	Made changes to sections 2.1 and 2.3.
1-25-2022	ljr	Made changes to section 2.6.
1-26-2022	ljr	Made changes to section 3, specifically sections 3.1 and 3.3.
1-27-2022	ljr	Made changes to section 2.2, 2.3 and 2.6.

2. The Concept of Operations (ConOps)

2.1. Current System or Situation

The “cold calling” system is designed to assist the instructor in choosing students to answer their questions. The students are randomly chosen in order to encourage participation from everyone in the class, especially those who do not usually raise their hands. This system is meant to evenly distribute the cold calls to ensure that no student is called more or less than another. The system will always be given a list of students in the class to choose from. Four students will be picked from this given list to be “on deck,” which allows those chosen to “warm-up” to being called on. The system is also designed to allow the instructor to “flag” the students who the instructor might want to follow up with after class.

2.2. Justification for a New System

In 2012, Dallimore, Hertenstein, and Platt conducted a study to investigate the effects of cold-calling on voluntary student participation. Their results revealed that there was a significantly higher number of students who answer questions voluntary in classes that use cold-calling than those that do not. Their results also proved that high-cold calling classes lead to an increase in students’ comfort participating in class.

Research conducted by Bekkering and Ward in December 2020 proved that class participation as a product of attendance and attentiveness is a valid way to measure and predict student performance. The results showed that there was a positive relationship between participation and scores on the final exam. Therefore, using both studies, we can make the assumption that since cold-calling increases student participation, and high student participation increases student success in class, then cold-calling can increase a student’s success in their learning environment.

Based on this information, we can recognize the importance of promoting student engagement during lectures. The cold calling system is an easy way to encourage this engagement which therefore promotes students to practice critical thinking skills and learn course material. While “cold calling” is not necessarily a new concept, the opportunity for students to “warm-up” before answering or asking questions is. This could make students more comfortable with cold calling since they have more time to prepare for their turn. In addition, the instructor has the option to flag a student, which makes it easier for the teacher to not only remember to reach out to the student individually but also easier to reach out in general.

2.3. Operational Features of the Proposed System

The cold-calling system will allow the instructor to randomly call on each student in the class an equal number of times while lecturing. The instructor will provide a list of all the students in their class to the system which it will then in turn parse into a queue data structure. The system will then display a list of four randomly chosen names from this list. From this list, the instructor can choose who to call on by using the keyboard arrow keys. They can use the left and right arrow keys to move between names, the up arrow key to flag and remove a student, and the down arrow key to only remove a student from the list. These keystroke decisions will be recorded in a file which can be exported when exiting the system. Once a student is removed from the On Deck list, their name will be placed in the back of the queue and a new student name will be chosen at random to replace the old name in the On Deck list. This cycle will continue through the entire class list until the instructor exits the system.

The instructor has the option to review the daily log file that recorded the cold-calling information from class that day. This file will contain all the student’s information as well as how many times they were called on during class along with how many flags were recorded for them. If a student has no flags recorded, there will be a 0. This file will help remind the instructor of any student’s they may want to follow up with after class. At the end of the term, the instructor also has the option to view a complete summary of how many times each student was called upon and flagged during the entire term. This enables the instructor to be able to share this information with each student in the class so that they may view how many times they participated.

The cold-calling system is designed to help encourage student participation during class. With the current system, student’s may not feel as comfortable or motivated to participate during lectures when the instructor asks questions. Low class participation

levels can be a possible indicator of a lack of understanding of the material being presented. With cold-calling, not only will students be able to demonstrate their understanding, but the instructor will have more knowledge on whether or not their class understands lecture material. The flagging system also assists the professor with remembering which students to reach out to and with what topic they want to follow up on.

2.4. User Classes

The main user class for this system is the instructor. The instructor will use the cold-calling system to randomly select students to answer their proposed questions or ask questions. The motive for the instructor to use this system during class is to encourage participation during lectures which in turn helps students learn lecture material and practice critical thinking skills. This system makes it easier for the instructor because they do not have to manually take valuable lecture time to continuously pick out random students to cold-call on. This system will also help the instructor keep track of the participation of each student in the class and reach out to them if necessary.

2.5. Modes of Operation

The mode instructors can use the cold-call system to randomly select a student to answer the question during the lecture. Also, the instructor can use Arrow keys to flag students whom they want to follow up with after class. All these processes go through the instructor's laptop, and students cannot see anything during the lecture. On the other hand, students can gain more and better experience during class by this cold-call system since this system randomly selects students from the list, they will focus more rather than only a few students participating in class.

2.6. Operational Scenarios (Also Known as "Use Cases")

Use Case: Everyday in class usage

Brief Description: An instructor wants to encourage class participation by cold-calling on students during class.

Actors: A student and instructor

Preconditions: A class instructor notices that his student's are not participating enough during lectures. To encourage more class participation, they decide to use a cold-calling software installed on their computer to equally but randomly call on all of the students while lecturing.

Steps to Complete the Task:

1. Instructor is ready to start lecturing and runs the cold-calling software through the terminal application on their computer.
2. During the lecture, the instructor notices low student engagement and decides to start cold-calling on students to answer their questions
3. The instructor selects the "New Session" button on the cold-calling main menu and an On Deck window is displayed with four student names visible.
4. The instructor uses the keyboard arrow keys to move between student names. They find the student they want to call on in the list.
5. The instructor can press the down arrow key to remove the student from the list or the up arrow key to flag and remove a student from the list.
6. If an up arrow key is pressed, then the instructor flags the student to remind themselves to follow up with them after class. The student name will be removed from the list, and a new name will replace the old on in the On Deck list.
7. Once a student's name is removed from the list, a new student name replaces the old one in the On Deck list.
8. The cold-calling information will be saved to a file that the instructor can access
9. Once the class is done, the instructor can exit the system

Postconditions: The instructor exits the system, but the cold-calling information collected during runtime is saved into files that are ready to be used the next time the system is used.

Use Case: The instructor wants to review the daily log file

Brief Description: After class, the instructor remembers they flagged a few student's on their responses during lecture. They review the daily log file produced by the system and obtain the student's email in the class roster file to reach out to them.

Actors: The instructor

Preconditions: While lecturing, a student asks a question that the instructor does not know the answer to. After class, the instructor finds the answer to the student's question and wants to send them the answer.

Steps to Complete the Task:

1. The instructor selects the Export Daily Data button on the cold-calling main menu
2. The daily log file is exported and available for instructor viewing.
3. The instructor can view if a student was flagged by looking at the response code. If the "Number of flags" is a 0, the student was not flagged. If there is a number larger than 0, then the instructor knows that student was flagged and will follow up with them.
4. The instructor composes an email to each flagged student to follow up on their topic of interest. This may include a paper that further explains something or an answer to a question that could not be answered in class.
5. The instructor sends the emails to each flagged student.

Postconditions: The instructor has reached out to every student that was flagged during class that day.

Use Case: At the end of the term, the instructor wants to send each student their cold calling summary for the term

Brief Description: Throughout the term, the system has recorded all of the cold calling data for each student in the class such as the number of times they were called on in addition to how many times they were flagged. The instructor wants to send each student in the class this information.

Actors: The instructor

Preconditions: The instructor wants to review all of the cold calling data collected for the term and send it to the students in their class.

Steps to Complete the Task:

1. The instructor chooses the "Export Performance Summary" button from the cold-calling main menu.
2. The instructor will see the list of all the students in the class along with each of their cold calling information. This will include all of the original information in the class list file along with the number of times they were called on and the number of times they were flagged and the days they were flagged. This file should be tab-delimited
3. The instructor saves this file to their computer
4. The instructor sends this file to each student in the class for them to review their class participation record from the term.

Postconditions: The instructor has sent each student in the class their correct cold-calling data summary for the term.

3. Specific Requirements

3.1. External Interfaces (Inputs and Outputs)

Must-Have:

1. Arrow Keys

   Description: Keyboard arrow keys which are used to choose and move between names in the “on deck” list for them to be removed and potentially flagged. Left and right arrow keys are used to move between names. The up arrow key is used to flag and remove a student from the On Deck window. The down arrow key is used to just remove a student from the On Deck window.

   Source of Input: User input from pushing the arrow keys on a keyboard.

   Valid ranges of inputs and outputs: Up, down, left, and right arrow keys

   Units of measure: None

   Data formats: None

2. Student List

   Description: A file containing each student’s first name, last name, UO ID, email address, phonetic spelling, and reveal code that is tab-delimited and is uploaded into the system.

   Source of Input: User input

   Valid ranges of inputs and outputs: .txt file

   Units of measure: Megabytes (MB)

   Data formats: Tab delimited and formatted as <first_name> <tab> <last_name> <tab> <UO ID> <tab> <email_address> <LF>. The UO ID will be nine digits. The reveal code is used as a place for the instructor to write notes.The first line until the first <LF> will be a comment that is to not be modified.

3. End of Term Log of Cold Calling Information

   Description: A file containing each student’s first name, last name, UO ID, email address, phonetic spelling, and reveal code that is tab-delimited and is to be uploaded into the system. This file may already contain past cold calling data.

   Source of output: .txt file

   Valid ranges of inputs and outputs: .txt file

   Units of measure: Megabytes (MB)

   Data formats: Tab-delimited and formatted as <total times called> <number of flags> <first_name> <last_name> <UO ID> <email_address> <phonetic_spelling> <Dates_cold_called>. The UO ID will be nine digits. The phonetic spelling is the pronunciation of the student name. The Dates cold called is a list of all the dates a student was called on in the format YYYY-MM-DD. The first line until the first <LF> will be a comment that is to not be modified.

4. Cold Calling Display

   Description: A small window at the top of the user's screen above lecture materials with three buttons to choose from. It will sit in the foreground of other open applications and will listen to keystrokes while being in the background.

   Source of output: The user's screen

   Valid ranges of inputs and outputs: A small, horizontal window displaying four names.

   Units of measure: None

   Data formats: Small, horizontal window at the top of the screen.

3.2. Functions

Must-Have:

1. Arrow Keys

   Validity checks on the inputs: Ensure that each arrow correctly corresponds with its associated movements.

   Sequence of operations in processing inputs:
   1. Check which arrow key was pressed first
   2. If the left arrow key is pressed first, the name at position one is highlighted first. If the right arrow key is pressed first, the name at position four is highlighted first.
   3. From the starting positions, use the left and right arrow keys to move between students whose names should highlight as they are reached.
   4. Once the desired name is reached, either use the up arrow key to flag and remove the student from the current on deck queue or use the down arrow key to only remove the student from the queue.
   5. After a student is removed from the queue, add another random student to the on-deck list visible on the screen.
   6. If a student is not flagged using the up arrow, then a 0 will be in the "Number of flags" section. If the student is flagged, then a number bigger than 0 will be present in the "Number of flags" section in the performance summary.

   Responses to abnormal situations:
   1. With the initial opening of the program and no names are highlighted, if the left arrow key is pressed, then the name in the first position is highlighted; if the right arrow key is pressed, then the name in the fourth position is highlighted.
   2. If you are in an end position like one or four and you press the left or right arrows, then no movement happens.

   Relationship of outputs to inputs:

   a.) input/output sequences: Input is the user input from keystrokes. Output is the associated event that occurs from pressing an arrow key.

   b.) formulas for input to output conversion: Taking user input of the keystroke and performing the correct movement or action associated with the arrow keys is the output.

2. Student List

Validity checks on the inputs: Making sure that the file is of the correct type and correct format. Ensure that Each information item for each student is of the correct format.

Sequence of operations in processing inputs:

1. At the start of the program, the system should import a file.
2. Upon import of the file, it will be checked if it is of the correct type.
3. Also upon export of the file, it will be checked if it is of the correct format.
4. If it is of the correct type and format, the system will continue with the import of the file for its use in the system.

Responses to abnormal situations:

1. If the file is not of the correct type, display an error indicating that the file is not of the correct type, and the file should not import until it is the correct type.
2. If the file is not of the correct format, display error indicating that the file is not of the correct format and the file should not import until it is the correct type.
3. If no file is provided, display an error message indicating that a file of a certain type and format is necessary to continue.

Relationship of outputs to inputs:

a.) input/output sequences: File is inputted into the system, and the output is using the file contents in the display. Output also includes daily logs and end of semester summary containing collected cold-calling data

b.) formulas for input to output conversion: Once the file is imported, it's contents will be parsed into a queue to be used by the system. While the system is running, this file will be collecting the cold-call data and storing it in a tab-delimited file that can be exported and seen by the user.

3. End of Term Log with Cold Calling Information

Validity checks on the inputs: Making sure that the file is of the correct type and correct format. Ensure that Each information item for each student is of the correct format.

Sequence of operations in processing inputs:

1. At the start of the program, the system should import a file.
2. Upon import of the file, it will be checked if it is of the correct type.
3. Also upon import of the file, it will be checked if it is of the correct format.
4. If it is of the correct type and format, the system will continue with the import of the file for its use in the system.

Responses to abnormal situations:

1. If the file is not of the correct type, display an error indicating that the file is not of the correct type, and the file should not import until it is the correct type.
2. If the file is not of the correct format, display an error indicating that the file is not of the correct format and the file should not import until it is the correct type.

Relationship of outputs to inputs:

a.) input/output sequences: File is inputted into the system, and the output is using the file contents in the display. Output also includes daily logs and end of semester summary containing collected cold-calling data.

b.) formulas for input to output conversion: Once the file is imported, its contents will be parsed into a queue to be used by the system. While the system is running, this file will be collecting the cold-call data and storing it in a tab-delimited file that can be exported and seen by the user.

4. Cold Calling Display

Validity checks on the inputs: Make sure that the display is in the correct position of the screen window. Make sure it is of the correct size. Make sure that it responds to keystrokes while in the background.

Sequence of operations in processing inputs:

1. If an arrow key is pressed, the display should update and respond to this keystroke and move to highlight the next name.
2. If an arrow key is pressed while the system is in the background of other apps, the system should still update and respond to the keystroke and move to highlight the next name.

Responses to abnormal situations:

1. If the display does not respond, remove and redownload the zip file and run the program.
2. If the display does not respond while in the background of other apps, remove and redownload the zip file and run the program.

Relationship of outputs to inputs:

a.) input/output sequences: The input into the display will be a tab-delimited file containing student information. The output of the display will be the window. The input into the display will also be the arrow keys, and the output

b.) formulas for input to output conversion: Display always responsive to user input from keystrokes. Uses keystrokes to add information to output files.

3.3. Usability Requirements

Required:

Must-Have:

1. The system shall import and use a file containing every student in the class and their associated information..
2. The system shall produce a display with four random student names visible
3. The system will allow the user to move between student names with the keyboard arrow keys.
4. The system will exit with the straightforward keyboard command “ESC”.
5. The system will record the number of times a student has been called upon along with any flags produced.
6. The system will have an option to export a file with the recorded cold-calling information
7. Keystrokes must work while the application is in the background.
8. The system will have a student flagging option for the instructor to write notes on responses.
9. The user must provide the initial student list to the system for it to execute.

Should Have:

10. The system will be up and running after a second of running the program through the command line in the terminal application.
11. The system will have the option to produce a daily log file after usage that shows the cold calling data from that day.
12. The system will automatically replace student names in the on-deck list after a name is removed from it.
13. The system will have the option to produce an end of term summary of each student's cold-calling performance during the term.
14. The system will prompt for user input on the student list file name.
15. The system should run with a maximum number of 150 students and a minimum of 10 students on the student list.
16. The system must run and respond to keystrokes while in the background of other applications.
17. The system will save cold-calling data to a file after every keystroke.
18. At system startup, the random number generator will restart with a value.

Not Required:

Could Have:

19. There is an optional secondary system which is a photo-based name learning system.
20. The instructor has the option to choose a number of students to be in the on-deck list.
21. System is able to change the use of tab-delimited files to the use of comma-delimited files.
22. A way to switch all key mappings from the arrow keys to the 1, 2, 3, and 4 number keys, "f" key for flags, and "r" key for remove.

3.4. Performance Requirements

Static:

1. .txt file of the class roster with any amount of students.
2. .txt file containing cold calling data.
3. The software runs and responds to keystrokes while in the background of other applications.
4. Keystrokes from user input.

Dynamic:

1. System startup within one second of double-clicking on the application
2. When the system starts, the random number generator will be re-seeded.
3. No data is lost during runtime or upon exit.

3.5. Software System Attributes

Software Attributes:

1. Reliability
2. Security
3. Privacy
4. Maintainability
5. Portability

Most Important Attributes:

1. System must run on Macintosh OSX 10.13 (High Sierra) or 10.14 (Mojave).
2. All system-related and system-development-related documents that are intended for human reading bust be in either plain text or PDF.
3. System will be built using Python 3 along with The Python Standard Library, but no other imports except for pyHook.
4. Python code must run in Python 3.7 through 3.10.
5. Instructions must be provided for how to compile the code.

6. No server connections may be required for either installing or running the software.
7. No virtual environments may be used.
8. No gaming engines such as Unity may be used.
9. There can be at most 20 user actions to compile the code and run the program.
10. An experienced computer programmer should not require more than 30 minutes working alone with the submitted materials to compile and run the code.

4. References

IEEE Std 1362-1998 (R2007). (2007). IEEE Guide for Information Technology–System Definition–Concept of Operations (ConOps) Document. https://ieeexplore.ieee.org/document/761853

IEEE Std 830-1998. (2007). IEEE Recommended Practice for Software Requirements Specifications. https://ieeexplore.ieee.org/document/720574

ISO/IEC/IEEE Intl Std 29148:2011. (2011). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/6146379

ISO/IEC/IEEE Intl Std 29148:2018. (2018). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/8559686

Faulk, Stuart. (2013). *Understanding Software Requirements*. https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf

Oracle. (2007). White Paper on "Getting Started With Use Case Modeling". Available at: https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

Tom, Sherrington. (2021). "Cold Calling: The #1 strategy for inclusive classrooms - remote and in person". Available at: https://teacherhead.com/2021/02/07/cold-calling-the-1-strategy-for-inclusive-classrooms-remote-and-in-person/

van Vliet, Hans. (2008). *Software Engineering: Principles and Practice*, 3rd edition, John Wiley & Sons.

Dallimore, E. J., Herteinstein, J. H., & Platt, M. B. (2012). *Impact of cold calling on student voluntary participation*. University of Minnesota Libraries. Retrieved January 28, 2022, Available at: https://uminntilt.files.wordpress.com/2015/09/impact-of-cold-calling-on-student-voluntary-participation.pdf

Bekkering, E., & Ward, T. (2020, December). *Class participation and student performance: A tale ... - eric*. Information Systems Education Journal. Retrieved January 28, 2022, from https://files.eric.ed.gov/fulltext/EJ1258148.pdf

5. Acknowledgements

This template was given to the UO CIS 422 class by Anthony Hornof. This template is similar to a document produced by Stuart Faulk in 2017, and uses publications cited within the document, such as IEEE Std 1362-1998 and ISO/IEC/IEEE Intl Std 29148:2018.