CIS 422 Project 1:
Ducks on Deck
Software Requirement Specification

Kalyn Koyanagi (kek), Ellie Kobak (erk), Kelly Schombert (ks), Liza Richards (ljr), Luying Cai(lc) - 5-6-2019 - v1.0

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.

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. 4 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

Student engagement is critical to the learning environment. 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.

With the current system, the instructor has to manually pick and choose students from the roster to call on while keeping track of the number of times each student is called to make sure that no student is called on more than the others. This method is time-consuming, which takes away valuable lecture time. In addition, it is rather distracting to the instructor because they have to continuously switch their focus back and forth from the material they are teaching to filling up the on deck list. This not only makes it hard for the instructor to maintain focus on the lecture, but also hard for the students to focus on the lecture, which can negatively impact their learning. The new cold-calling system takes far less time by doing the picking, choosing, and sharing for the instructor much faster while also allowing the instructor and the students to have an uninterrupted focus on the lecture.

2.3. Operational Features of the Proposed System

The apparent disadvantage of the existing questioning system is that only a few people interact with the professor in each class. Only a small number of people will interact with the professor, and many students may not participate in the classroom. Professors cannot get their feedback, nor can they know whether the students are understanding or focusing on the content of the class. Therefore as a solution, the cold call system can help the professor solve the problem of being unable to interact with students who are afraid of raising their hands. The selection of students in the cold call system is random, so there will be an equal distribution of participation opportunities. After getting the student's response, the professor can choose to mark the student to facilitate follow-up on the student's questions or topics of interest after class.

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: Day-to-day usage

Brief Description: Daily instructors use cold-call-assist software installed on their laptop. After the instructor opens the software, it will automatically be positioned on the top of PowerPoint with four names displayed. The instructor can choose any displayed student to call on, and if they respond then the instructor has the option to flag and remove a student or just remove them from the displayed list. If the instructor flags a student, they can write a note regarding their response. To raise a "flag" for a student, the instructor can use arrow key input. The system will re-jumble the names between runs while making sure students who were not cold called at the previous class will be prioritized. After class, the instructor can easily copy students' information to write an email to contact the flagged students.

Actors: A student and instructor

Preconditions: An instructor wants to encourage students to participate in class discussions, is comfortable "cold-calling" on students, wants to make sure each student is treated equally with the cold-calling, and uses a computer as part of the in-class presentation.

Steps to Complete the Task:

1. The instructor is in the classroom with their computer, ready to start the lecture. They plug in their computer to the projector and open Powerpoint and start lecturing.
2. While lecturing, the instructor wants to check in with the class to see if students are comprehending lecture material, so they begin asking questions to the class and wait for a response.
3. When no students offer to respond to the question, the teacher uses the cold-call-assist software installed on their laptop. The instructor searches in the hard drive for the program within 5 seconds.
4. When the system is opened, a list of four student names automatically appears in an "on deck" horizontal window positioned to take up as little space as possible to avoid blocking the presentation. The instructor places the focus back on the Powerpoint slides, but the on deck window stays in the front.
5. The instructor can note which students were cold-called at the beginning or end of the previous class to ensure that the system is randomized between run times.
6. The instructor asks if the students on the list would like to answer the question.

Postconditions: The system stops running, but is still ready for the next time it is used.

Use Case: After class, the instructor reviews if any students might benefit from encouragement.

Brief Description: After checking the cold call log file, the instructor can send emails to the flagged students with notes in the response code, otherwise the response code will have “X.” The instructor can use the copy command to copy a student’s email address.

Actors: The instructor

Preconditions: After class, the instructor knows that they did some cold-calling in class that day and that a student demonstrated special interest in a topic. They want to email a conference paper on the topic to that student.

Steps to Complete the Task:

1. Teach the class using cold calling software
2. After class, the instructor reviews the daily log file and its contents. At the top of the file, there is a heading to indicate that this is the daily log file for the cold call-assist program.
3. Today’s date is under the heading, and then there is one line for each cold-call that is made that day. Each line is formatted as:

   <response_code> <tab> <first name> <last name> “<” <email address> “>”

   The response code is blank if there was no flag and “X” if there was.
4. The instructor can recall how the student acted and participated in class. One student was having too many side conversations and did not hear when they were called on, so the instructor wants to email this student to remind them to pay more attention in class. Or the instructor can remember that the student has a special interest in a topic and wants to send them a paper or article regarding the topic.
5. The instructor drags the mouse across the portion of the line containing the students’ email addresses in brackets and uses the copy command. Then paste this email address into the “To:” header of her email client.
6. The instructor composes and sends the email.

Postconditions: The instructor is done with their after-class review. Students have been emailed.

Use Case: At the end of the term, the instructor reviews a summary of class participation.

Brief Description: The instructor will use students' term performance data to see their cold-call performance. Data import into spreadsheet will using <total times called> and <number of flags> to compute a cold-call participation score for students. Those scores can be imported into Canvas for students to check.

Actors: The instructor

Preconditions: The instructor wants to review a summary of each student’s performance across all of the times that each student was cold-called during the term.

Steps to Complete the Task:

1. The instructor opens the summary performance file and reviews its contents
2. The instructor will see the heading at the top of the file, which indicates this is the summary performance file.
3. The instructor sees the heading of the file, which indicates this is the summary performance file for the cold-call-assist program, and headings for the columns in the lab-delimited file.
4. The instructor sees the list of students with performance data after each student, each line will be formatted as:

   <total times called> <number of flags> <first name> <last name> <UO ID> <email address> <phonetic spelling> <reveal code> <list of dates>

   There should be a tab between each field and a Unix linefeed at the end of the line. The list of dates includes all the days the student was cold-called, and they are formatted as YY/MM/DD and are in chronological

order.

5. The instructor imports the data into a spreadsheet and figures out an Excel formula that uses the two numbers at the start of each line to compute a cold-call participation score for each student. Now there are three numbers related to cold calls for each student.
6. The instructor imports all three numbers into Canvas so the students can see their cold-call performance for the term.

Postconditions: The instructor is done using the system for the term. Students have received feedback, the rewards for their preparation for being called-on in the class

3. Specific Requirements

3.1. External Interfaces (Inputs and Outputs)

Must-Have:

1. Arrow Keys

   Description: Used to choose and move between names in the "on deck" list for them to be removed and potentially flagged.

   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> <tab> <phonetic_spelling> <tab> <reveal_code> <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> <tab> <email_address> <phonetic_spelling> <reveal_code>. 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.

4. Cold Calling Display

Description: A small, horizontal window at the top of the user's screen above lecture materials with four student names. 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 an "X" will replace the <response_code>. If the student is flagged, then the instructor will type out any notes or comments he has on the student's response or question.

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 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 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 is not in the correct position on the screen
2. If the display is too small or too large
3. If the display does not respond while in the background of other apps

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. A file containing every student in the class and their associated information.
2. A display with four random student names visible to be chosen from using the arrow keys
3. Arrow keys the user can use to choose and move between students.
4. A file containing all cold-calling data collected from previous uses, which can be imported and exported.
5. Keystrokes must work while the application is in the background.
6. A flagging system for the instructor to write notes on student responses.

Should Have:

7. The system should be up and running after a second of double-clicking on the icon.
8. A daily log is provided after system usage that shows the cold calling data from that day.
9. The student names should shift in the on-deck list after a name is removed.

10. End of term summary of each student's performance across all of the times that each student was cold-called during the term.
11. The system should run with any number of students on the student list.
12. On deck display must take up as little screen space as possible.
13. The system must run and respond to keystrokes while in the background of other applications.
14. Save cold-calling data to a file after every keystroke so that no data is lost when exiting the system.
15. At system startup, the random number generator will be re-seeded with a different seed, so the system is ensured to be random.

Not Required:

Could Have:

16. An optional secondary system which is a photo-based name learning system.
17. The system should not generate any error messages that interfere with the running of the program.
18. The instructor chooses a number of students to be on the on-deck list.
19. Is possible to change the system's use of tab-delimited files to the use of comma-delimited files.
20. A way to switch all key mappings.

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.

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.