On Deck
Software Design Specifications

K. Hosaka (kah), A. Huque (ash), L. Jim (lmj), M. Terry (mht) - 2-03-2020 - v1.03

Table of Contents

1. SDS Revision History

Date	Author	Description
1-15-2020	lmj	Created the initial document.
1-15-2020	lmj	Wrote sections 3, 6, and 7.
1-16-2020	lmj	Wrote section 4.
1-16-2020	kah	Proofreading, small edits, wrote section 5.
1-16-2020	ash	Proofreading, small edits.
1-17-2020	lmj	Proofreading, small edits, reformatted section 5.
1-17-2020	ash	Formatting, small edits.
1-17-2020	kah	Final review and edits for initial submission.
2-01-2020	kah	Initial revision for final submission.
2-02-2020	kah	Editing diagrams for updated accuracy.
2-02-2020	lmj	Made quick updates to section 4.3 for accuracy.
2-02-2020	mht	Reviewing and checking accuracy.
2-02-2020	ash	Proofreading and editing
2-03-2020	lmj	Made small updates to section 4.3 and reviewed the entire document for accuracy making small edits.

2. System Overview

Upon starting *On Deck*, the user will be greeted with the home view. This allows the user to access the Record System (through the Roster Information window), Presentation View, and System Randomizer Tester (under Settings) as well as interact with the Student Selection System. These four main parts of the program are the four modules in the software architecture. The Record System stores student information and allows the user to import and export information. The Student Selection System randomizes the names it gets from the Record System into a queue for the Presentation View to display the names that are on deck. The Student Selection System then allows the user to dismiss and flag on deck names, and sends this information to the Record System to store and eventually turn into the Daily Log and Performance Summary Report. Finally, the user can enter the System Randomizer Tester mode, which will utilize the Student Selection System to run a test of the randomization ability of the software. The Record System will produce a report of the test results it receives from the Student Selection System.

3. Software Architecture

Components:

- Record System - import/update roster information, update/export daily log, and update/export performance summary report
- Student Selection System - randomize the student queue and provide functionality for flagging and dismissing the names on deck

- Presentation View - present the on deck names in front of any other open windows
- System Randomizer Tester - allow the user to test the randomization functionality of the queue

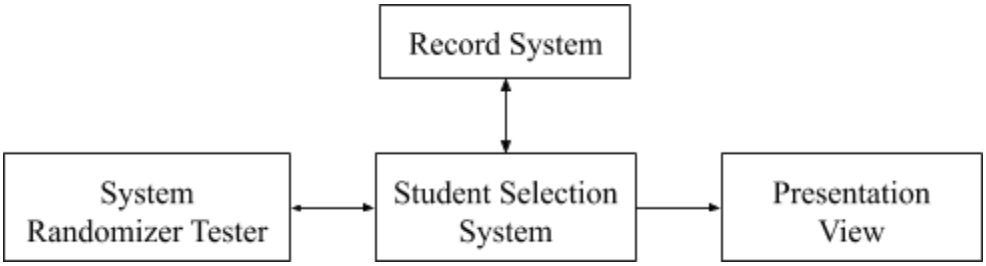

Figure 1. *On Deck* Architecture Diagram. The boxes represent components. The arrows indicate the flow of information sharing between modules.

Component Interaction:
The Record System allows the user to import, update, and store roster information. The Student Selection System will utilize the information stored in the Record System to randomize the names in the queue. The Student Selection System will send the first four names in the queue to the Presentation View. When the Student Selection System receives a command to dismiss a name on deck, it will update the Presentation View. It will also send data to the Record System, including who was called on and whether they were flagged or not. The Record System will take this information to update and produce a daily log and a performance summary report. The System Randomizer Tester will utilize the Student Selection System to run a simulation on the randomness of the student selection. The Student Selection System, when run by the System Randomizer Tester, will communicate the test information to the Record System. The Record System will then produce a test report of the randomization efficiency as to not override actual student information.

Design Rationale:
- The Student Selection System sends information to both the System Randomizer Tester and Presentation View because they both require the use of the randomized queue with student names and a way to dismiss names from on deck. The process to dismiss a name is not included in the Presentation View module because, while necessary to the presentation functionality, the System Randomizer Tester should not need to use functions from the Presentation View to work nor should have to have its own dismissal function.
- Information travels both directions between the Record System and the Student Selection System. The queue needs the names that were imported into the Record System, and the Record System contains performance information to export based on information received from the Student Selection System which can indicate if a name has been called or flagged. The information is not saved in the Student Selection System because importing, exporting, and storing the information is a functionality that can be ported to another program and does not need to be tied in with the queueing functionality.
- The Presentation View is its own module due to the labor required to make the presentation user interface run in the foreground while an active application is in full

screen mode. This functionality could ideally be utilized by other programs. Thus, it is its own module.

- The System Randomizer Tester is separate from the Student Selection System because *On Deck* provides a specific testing mode for the user to select and run the randomization functionality. This functionality can be utilized by the user to see if each student is getting cold-called at roughly the same rate as other students. Additionally, the System Randomizer Tester is not necessary for the functionality of the Student Selection System, thus it is considered its own module.

4. Software Modules

4.1. Record System

Primary Function:
The Record System contains information pertaining to students. It allows the instructor to import new and updated class rosters as well as export a Performance Summary Report and the Daily Log. The Record System is also responsible for producing test results when the Student Selection System is ran through the System Randomization Tester.

- Import and Update Roster Information:
  Using Python I/O functions, the Record System reads in a tab-delimited file and inserts the information into a dictionary of student information. It also allows the user to export the information contained in the dictionary in tab-delimited format for the user to see what roster information is in the system. The user can also upload an updated roster without clearing all student information first.
- Update and Export Daily Log:
  While the system is running, the Record System keeps track, in a dictionary, the students who were dismissed from on deck and whether or not they were flagged. At the end of each session, the Record System updates and exports a daily log file containing information about which students were called on and whether they were flagged. It cross references the student information dictionary to include the student's email address in the daily log.
- Update and Export Performance Summary Report:
  When the desired session is over, the user can export a tab-delimited file summarizing each student's performance. Each student's information appears on a single line with a Unix line feed at the end of the line. The report includes how many times each student was called or flagged, their name, student ID, email, phonetic spelling, and a list of dates that they were called on. In order to accomplish this, each time the program runs, student information is updated to increment the total number of times they were called on or flagged, as well as append the date to the student's list of information.
- Testing:
  On occasion, the Student Selection System may be ran in testing mode. Students' actual records are not edited when in testing mode. A copy of the students' information is made with the number of calls and flags set to zero with no existing 'called on' dates. While in testing mode, this dictionary copy is what gets updated. When the test completes, the

Record System outputs a file containing information about the testing results. This includes each student's name, how many times the student was cold-called, and the percent of cold-calls that the student received out of all possible calls.

Interface Specification:
The user will import and export files from the user interface. The Student Selection System retrieves and updates information in the student dictionary. The Record System should provide a public dictionary of student information for the Student Selection System. It also provides a public function so a student's total number of times called or flagged can be incremented. This function also triggers the system to append the day's date to the student's information. For testing, the Record System provides an additional function to update the test information instead of students' actual information.

Models:

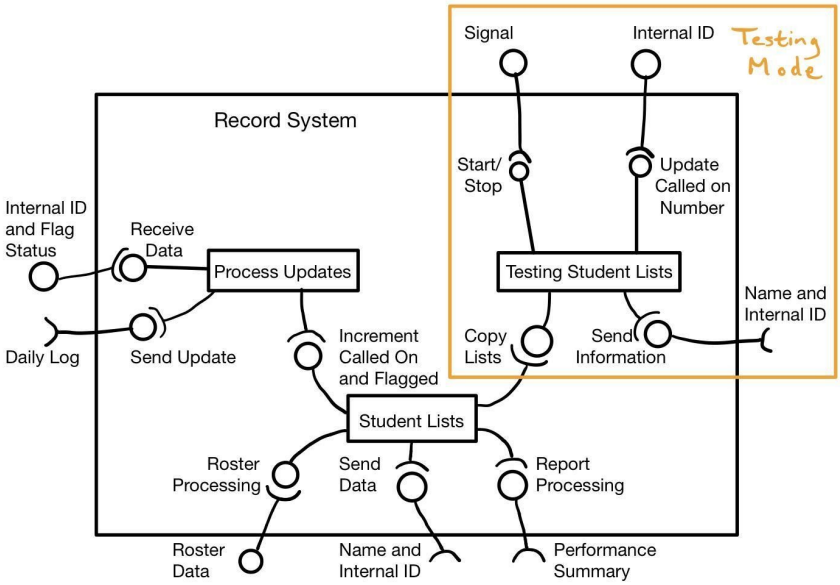

Figure 2. Record System UML Component Diagram.

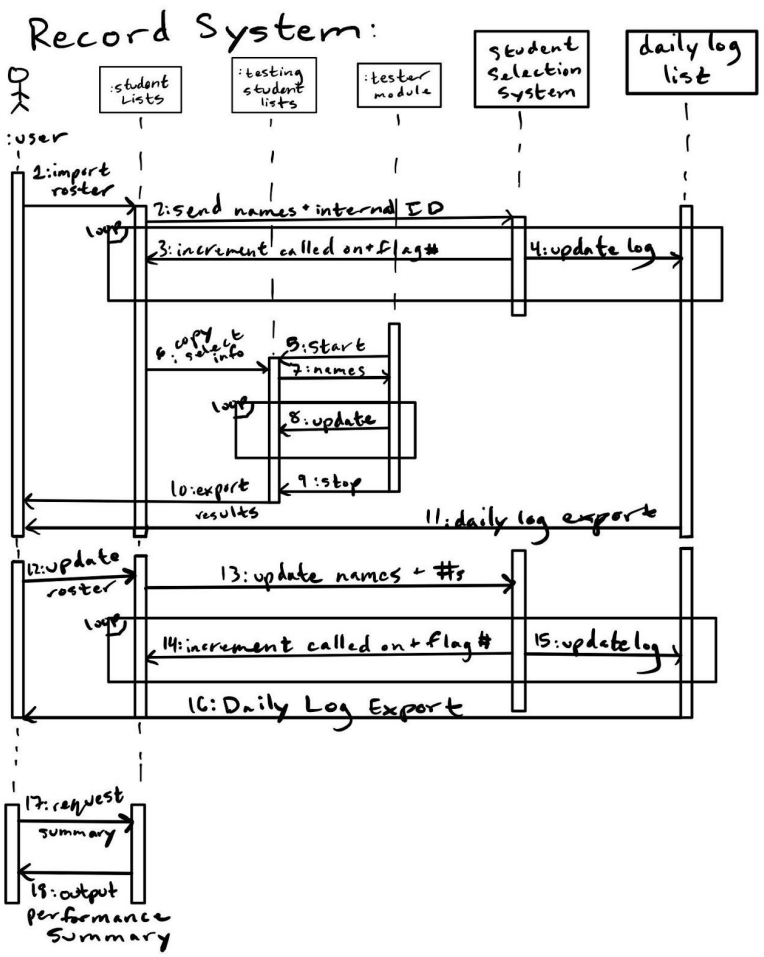

Figure 3. Record System UML Sequence Diagram.

Design Rationale:
The Record System keeps track of all the information except for the order of the queue. It is not set up to keep track of the order of the queue. This is because that exact functionality fits better in the module that runs the randomization of the queue in addition to taking inputs to dismiss and flag students on deck. The Record System does, however, keep track of the number of calls and flags a student has had. This is because the Student Selection System does not maintain the rest of the information needed for the Daily Log or Performance Summary Report. It only maintains the names of the students and their unique student ID numbers to locate the student's information in the Record System. This means all the information the Record System needs for its output files are kept in the same place.

4.2. Student Selection System

Primary Function:
The Student Selection System retrieves the names of students and randomizes their order in a queue. The first four students in the queue are considered on deck, and the user can dismiss and flag any of the on deck students. Students who have been removed from on deck are

re-randomized behind those who have not been called yet in the queue. Between program sessions, the order of the queue is saved in a python file. As such, the queue state is maintained in its exact form.

- Randomized Queue:
  The names of each student will initially be randomly put into a list. As each student is dismissed from on deck, the name will be randomly placed in a secondary list. Once the first list has been almost entirely emptied (4 students remaining), the secondary list is randomized and appended to the primary list. The secondary list is then set to empty. This pattern continues until the end of the term. This insures no student will be called on twice before another student has been on deck once. When there are less than four students on the primary list, the "on deck" portion will start to go through the secondary list.
- Dismissal and Flagging:
  The first four names in the randomized queue are considered on deck. The user can select any of those four names by using the left/right arrows. This system will keep track of which of the four names is selected. Looping from the first to the fourth name or fourth to first name is not supported. Hitting the up arrow simply dismisses the highlighted name, while hitting the down arrow dismisses and flags the name from on deck.
- Testing:
  In the testing mode, the existing queue remains untouched. When testing mode is entered, the Student Selection System creates an empty instance of the existing queue. The key strokes, which are randomly generated, will be received from the System Randomizer Tester instead of user input. When indicating dismissals and flags to the Record System, they will be marked as test results and not affect students' actual data. The results of the test are then output into a tab-delimited file.

Interface Specification:
The Student Selection System gets the names of all of the students from the Record System. It also listens for keystrokes. When a student is dismissed from the queue, the Record System is triggered to increment that student's number of calls as well as flags if they were also flagged. When the queue is initially created, the Student Selection System sends the first four names to the Presentation View so they can be displayed as the on deck students. When the user hits the left or right arrow keys, the Presentation View needs to be notified to highlight the next name. Whenever a name is dismissed from the queue, the Presentation View needs to be updated with which name to remove from the display and which one to add. If the Record System receives a roster update, any names that were added or deleted need to be reflected in the queue as well.

When using the System Randomizer Tester, the Student Selection System will not communicate with the Presentation View. It also listens for keystrokes from the tester and not from the user. It will then send the on deck information to the System Randomizer Tester. Next it will send the Record System dismissals, indicating that they are from a test, so they are logged appropriately. When the Student Selection System receives the signal that the test is done, it will trigger the Record System to export the testing results.

Models:

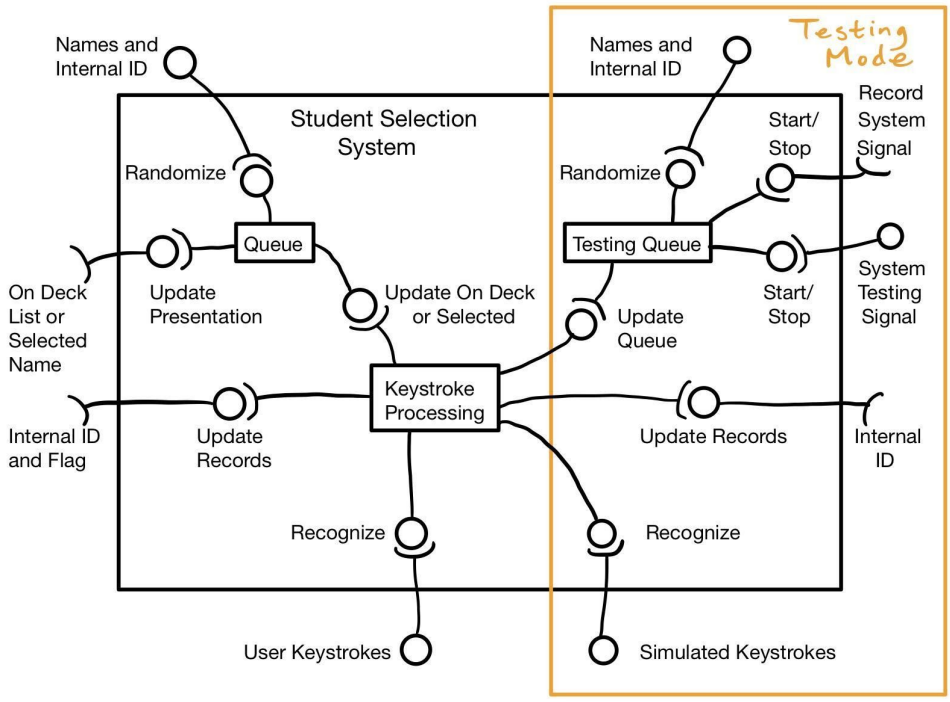

Figure 4. Student Selection System UML Component Diagram.

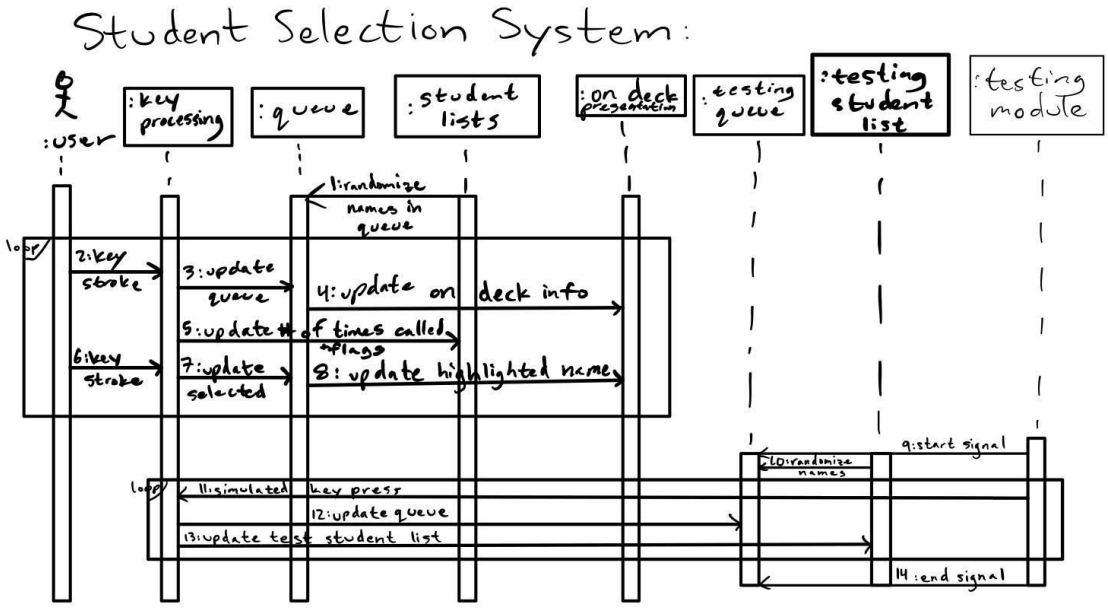

Figure 5. Student Selection System UML Sequence Diagram.

Design Rationale:
The Student Selection System contains the ability to dismiss and flag names from on deck, instead of the presentation mode module, because the tester needs to be able to interact with the selection system, dismissing names as well. By having the dismissal functionality in the Student

Selection System, both the presentation mode and testing mode have access to this functionality. The downside is that the dismissal and flagging is not built in to the Presentation View module.

4.3. Presentation View

Primary Function:
The Presentation View displays the names that are on deck in the foreground even if another application is active. The On Deck window remains in the foreground as long as it is open, but it will not receive key inputs unless it the active window (clicked on). While in this active mode, a user can use the arrow keys to select and dismiss names.

- Display:
  Instead of using tkinter's default settings to display the application, modifications were made to maintain viewability. Namely, the window was set to be displayed in the foreground. Once active, a list of the four students on deck is maintained. The name that is currently selected is highlighted in yellow. If the name that is dismissed is not the furthest right name, any names already on the screen will slide over to the left. The new name on deck will then appear on the far right. This ordering is also reflected in the on deck list that the Presentation View is maintaining.
- Input:
  When in Presentation View mode, the application should be clicked on so that the application can begin listening for input from the keyboard. The Student Selection System handles the keystrokes, but this module makes sure that *On Deck* will receive any keystrokes passed.

Interface Specification:
The Student Selection System notifies the Presentation View when the name selected by the arrow keys has changed. It also updates the on deck list when a name is dismissed and added, so a public function was created.

Models:

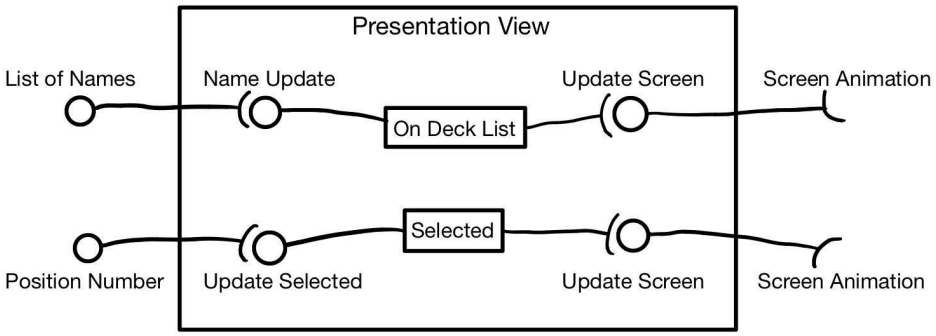

Figure 6. Presentation View UML Component Diagram.

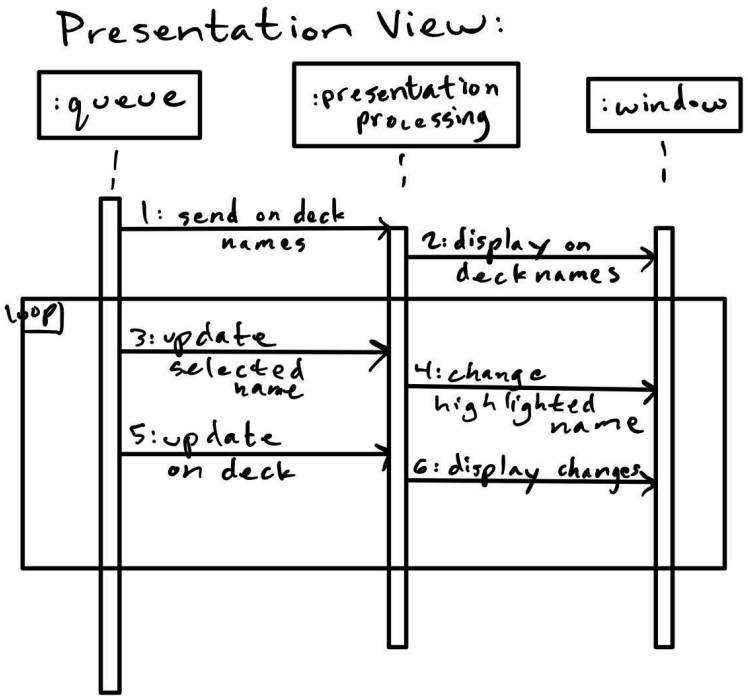

Figure 7. Presentation View UML Sequence Diagram.

Design Rationale:
The main focus of this model is for the display to work correctly. This is why it does not need to keep track of keystrokes or maintain the entire queue. It simply needs to know what four names to display and which one to highlight. This makes the module more portable as a list of any four things can be fed in and any keys can be mapped to change the list, but the display functionality remains the same.

4.4. System Randomizer Tester

Primary Function:
The System Randomizer Tester allows the user to verify the randomness of the queue.

- Testing:
  When the user selects to start a test, the program asks the user if they are sure they want to continue. If yes, the program enters testing mode. This module triggers 1000 random cold calls on the queue. The information containing which students were called is then sent to the Record System to output the testing report. The testing report contains each student's name, the number of times they were called, and a percentage indicating how many times they were called out of all total calls. The testing report is in the format of a .tsv file.

Interface Specification:
The System Randomizer Tester sends keystrokes to the Student Selection System to make changes to the testing queue (and the Student Selection System interacts with the Record

System). Once 1000 random calls are executed, the information is saved in a dictionary ready to be outputted. The Student Selection System then calls the Record System's function to output the testing report.

Models:

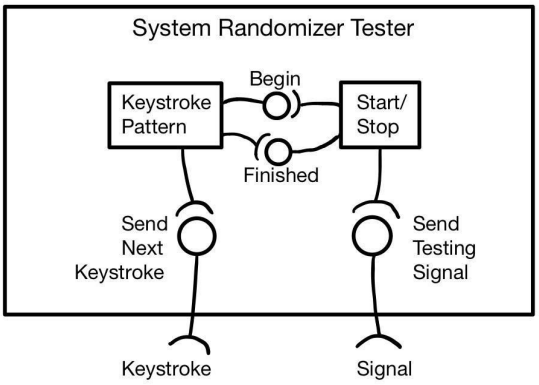

Figure 8. System Randomizer Tester UML Component Diagram.

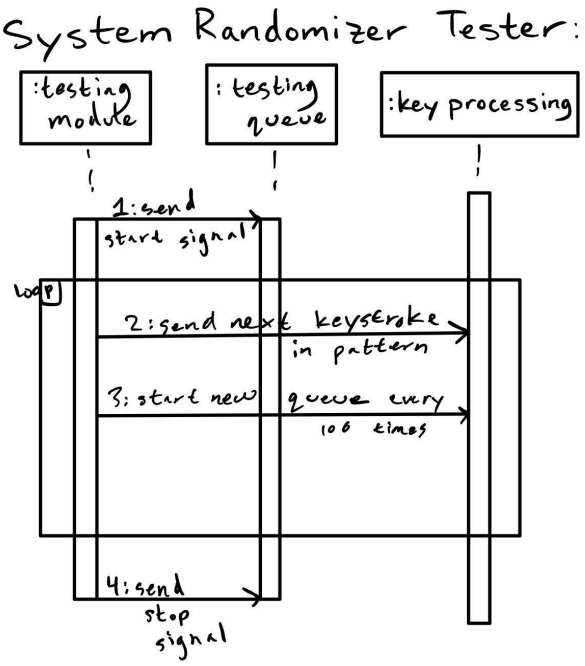

Figure 9. System Randomizer Tester UML Sequence Diagram.

Design Rationale:
The purpose of this module is to have a way to switch the system into a testing mode and send numerous keystrokes to the program to test the randomness of the queue. This is not included under the Student Selection System, as it mimics the purpose of the Presentation View in regards to seeing who is on deck and dismissing names.

4.5. Alternative Designs

- Instead of using arrow keys as the selection method, the user can utilize a Nintendo Switch Joy-Con. This would allow the instructor to move around a classroom and still dismiss and flag students on deck.
- When in Presentation View mode, the application should still listen for input from the keyboard, even when a different application is active.
    - The user can select any of the four on deck names by using command + left/right arrow. Hitting command + down arrow dismisses and flags the highlighted name while hitting command + up arrow simply dismisses the name from on deck.
    - Instead of using arrow keys as the selection method, numbers 1, 2, 3, and 4 select and dismiss the corresponding name. When using the number keys, the name will be immediately dismissed from on deck, so to add a flag, the user must follow the number key with key Q, E, or R within one second.

5. Dynamic Models of Operational Scenarios

5.1. Day-to-Day In-Class Usage

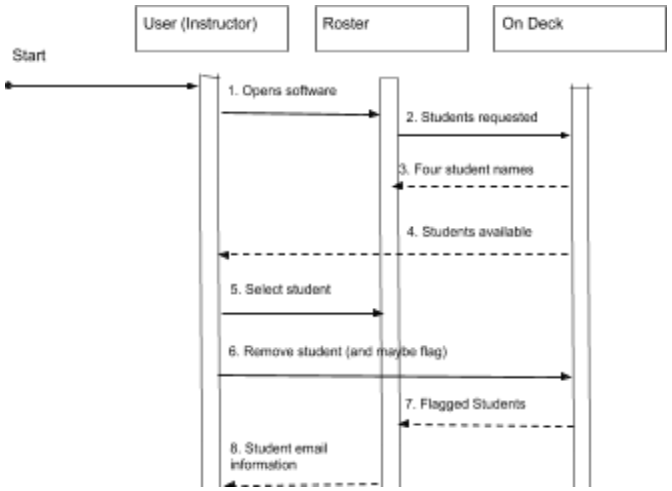

Figure 10. UML Sequence Diagram for day-to-day in class usage.

5.2. After Class Instructor Review

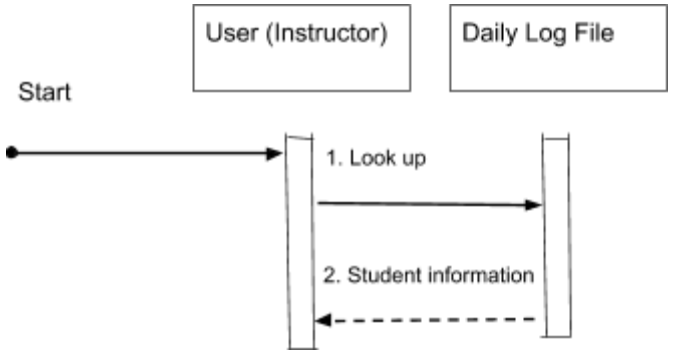

Figure 11. UML Sequence Diagram for after class instructor review.

5.3. Update Roster Information

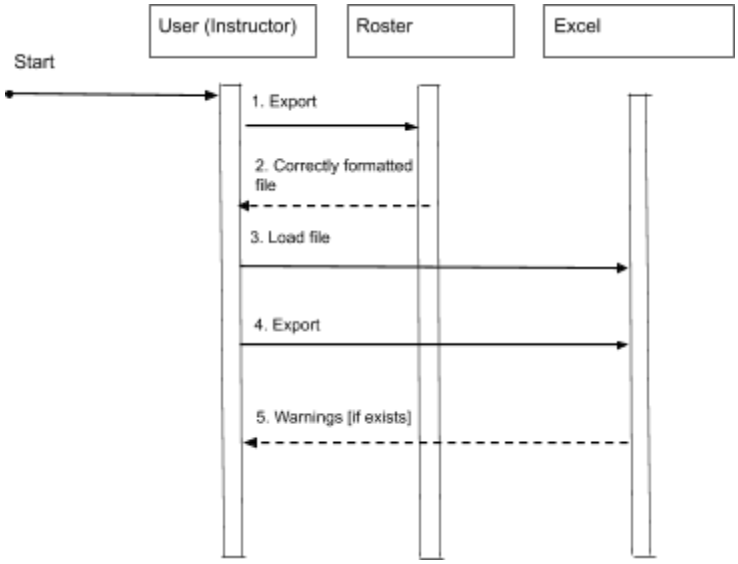

Figure 12. UML Sequence Diagram for updating roster information.

5.4. End of Term Performance Summary

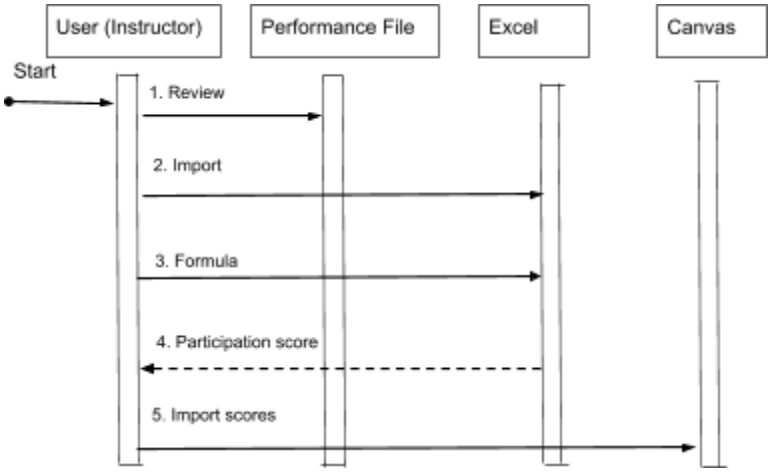

Figure 13. UML Sequence Diagram for end of term performance summary.

5.5. Start of Term Preparation

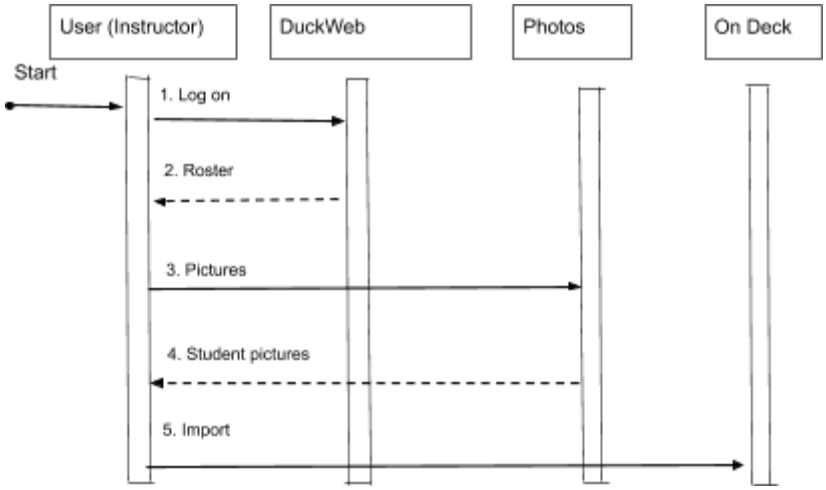

Figure 14. UML Sequence Diagram for start of term preparation.

6. References

Hornof, Anthony. (2019). CIS 422 Software Design Specification Template. Downloaded from https://classes.cs.uoregon.edu/20W/cis422/Handouts/Template-SDS.pdf in 2020.

Hornof, Anthony. (2020). CIS 422 Software Requirement Specification (SRS) for a Classroom Cold-Call Assist Software Program. Downloaded from https://classes.cs.uoregon.edu/20W/cis422/Handouts/Cold_Call_System_SRS_v2.pdf in 2020.

Visual Paradigm. (2020). What is Component Diagram?. Accessed at https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/.

Visual Paradigm. (2020). What is Sequence Diagram?. Accessed at https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.

7. Acknowledgments

This document uses a template provided by Anothony Hornof in CIS 422 at the University of Oregon. The ideas contained in this document were generated from his Software Requirement Specification for a Classroom Cold-Call Assist Software Program.

To create the diagrams, the site Visual Paradigm was referenced for formatting purposes.