Marie Hartung
Professor Macleod
COS 420 - Childhood Immunizations
May 6, 2020

Final Progress Report

Progress Made in Each Iteration:

Iteration One -- In this first iteration, our group's main goal was to set up the project foundation and conceptualize the functionality of the software. The first programming team (John and Alex) spent their time during this iteration setting up the model-view-controller framework for the project, as well as creating a simple text-based user interface. The MVC framework allowed us to organize our code and manage communication between classes, and the UI allowed us to simulate what we wanted the user experience to look like. The second programming team (Calen and Marie) dedicated their time to creating a model for the Immunization Register (dubbed ImmReg) objects, and creating various diagrams to aid in our group's understanding of the software functionality. Though the ImmReg model was only a draft because we were still determining how to best represent the registry form, it served as the base class for storing vaccination information. The diagrams we created, including a class diagram[1] and two system sequence diagrams,[2-3] provided references for the group to use when discussing the software. Both groups were able to complete their tasks during the iteration, but we did discover that we severely underestimated the amount of time required per task. To remedy this, we made a note as a group to allocate more time for tasks in the following sprints.

Iteration Two -- Our group's goal for the second iteration was to refactor our code almost entirely. Most of the code we developed in Iteration 1 was either deliberately non-functional (e.g. the Daos to store objects in a JSON), or needed to be updated to reflect new information that we had learned. We also needed to begin adding functionality to the system, so we started with the use cases for adding and modifying ImmReg objects stored in the JSON.[4-5] The use cases were divided evenly among the two programming teams, with team one (Calen and Alex) working on updating ImmReg objects, and team two (Marie and Fazil) working on creating new ImmReg objects. For patients with no existing immunization record, a new ImmReg object would need to be created and stored. If they were to return at a later date, their record (the ImmReg object) would need to be retrieved from storage, modified, and then stored again. These use cases represent the main uses of our system, so it was important to get them working correctly. Both use cases were ultimately completed, but there were some underlying last-minute bugs in the returning patient use case that would have to wait until Iteration 3 to be fixed.

Iteration Three -- In the third iteration, our group had a lot to complete. The first programming team (Calen and Fazil) were tasked with incorporating the Consulting Register's patient search service into our code, developing a graphical user interface with Swing,[6-7] and adding unit tests for important parts of the program. The second programming team tackled creating a vaccine class and integrating it with the current system for creating and storing ImmReg objects, and refactoring the code to add clear and consistent documentation. The second programming team also spent a significant amount of time planning out how to generate monthly reports from the stored immunization records, primarily by creating a system sequence diagram.[8] There were two major risks in this iteration: The first risk was that only one person in our group had previously used Swing, which meant the rest of the group had to watch tutorials and do some independent studying to learn how to work with it. The second risk was that the monthly returns contain information that we weren't tracking in our ImmReg objects, so we had to decide what additional fields to add to the ImmReg class and what to have the user input when generating a report. Both teams made a significant amount of progress this iteration, with the first team successfully integrating the patient search service, unit tests, and a large chunk of the Swing GUI, and the second team successfully creating a vaccine subclass, adding documentation, and starting the monthly return generation code.

Iteration Four -- The fourth iteration was the one in which our group fully shifted from a text-based UI to the Swing GUI. We split the two use cases between the two programming teams, with team one (Marie and Alex) working to finish the monthly report generation, and team two (John and Fazil) transitioning the rest of the project view from its text UI to the Swing GUI. While this work wasn't necessarily difficult once we familiarized ourselves with Swing, it was more than time consuming. Every single line of code that involved the text prompts we had originally made (e.g. "Enter the numbers corresponding with the vaccines received today:") had to be replaced with a panel, labels, buttons, drop-down menus, and/or text fields. What was even more challenging was team one's decision to skip creating a text-based UI for the monthly report functionality and instead jump right to integrating it with Swing. By the end of the iteration, team two was able to complete the system's transition from text UI to Swing,[9-11] and team one was able to get roughly 90% of the monthly report generation completed, with the remaining 10% being the printing of the report for the user to view.[12-13] At this point, the software was, for all intents and purposes, complete. Our next steps were to finish getting the monthly report to print, and to migrate our project to the cloud.

Iteration Five -- Iteration 5 for our group was, as mentioned multiple times in class, "disruption." While 4 out of 5 group members had previously taken a database course, the group still ran into trouble properly setting up the Google Cloud projects on each of our local systems. A big part of the issue was lack of experience; we all struggled to properly get the lab projects running and deployed, which led to some confusion when it came to doing the same for our project. Another part of the issue was consistency in our operating systems. Some of the group members primarily use Linux for their work, and so certain extra steps have had to be taken (compared to Windows) to allow Maven and its respective plugins to properly set up, run, and deploy the Google Cloud projects. Some group members were still unable to get the patient lab projects running prior to starting work for this iteration, which made things especially difficult. Regardless, our group was determined to get a database for our Immunization Registry (ImmReg) objects up and running. Both teams were essentially merged into one for the purposes of completing this one use case, with all four members working together around one screen (Fazil's, virtually) to debug build errors and create the various classes needed for the setup. Ultimately, the team was able to get both the create and update functions working in some capacity, but was unable to get the rest of the functions to work

Strengths and Weaknesses:

All in all, the system is a solid digitized replica of the original Immunization Registry and Immunization Monthly Returns forms that runs smoothly as a local Java application, though it is not quite ready to deploy as a web app. Given the vague nature and minimal instruction on how to complete both forms (with the exception of the informational section for filling out the registry that was provided to us), it was ultimately left up to the group to determine what data to store where and as what type. While this has been beneficial in terms of granting the team creative liberties to design the system, it has also led to some difficult decisions regarding how best to represent the form; i.e. having to decide whether we should match the form exactly, or if it would be better to add certain data fields to the form so we can track them. The team has done their best to satisfy both options, sticking to modeling the form directly for the most part, but adding additional fields as needed. The parts of our program that do have error-checking are robust, but we could definitely have spent more time adding error-checking to more parts of our program.

Remaining Challenges:

As mentioned above, the program could definitely utilize a lot more error checking, especially in the web application if we were to get that working. As of now, the web app would be vulnerable to various injection attacks given the lack of bounds checking. This is something we'd have to fix if we were to launch this web app outside of a classroom setting. Given more time, we would also want to finish developing the Swing GUI for the monthly reports for the local Java application, so that the user can actually see a fully detailed report of each month's vaccination statistics.

Diagrams by Iteration (Links):

Iteration 1 --

[1] Class Diagram
[2] System Sequence Diagram for Modifying Records
[3] System Sequence Diagram for Viewing Records

Iteration 2 --

[4] System Sequence Diagram for New Patients
[5] System Sequence Diagram for Existing Patients
Updated UML Diagram

Iteration 3 --

Immunization Register/Vaccine Class Diagram
[6] Swing UI Concept Diagram Top Half
[7] Swing UI Concept Diagram Bottom Half
[8] System Sequence Diagram for Generating Monthly Returns

Iteration 4 --

[9] Swing GUI Example Diagram 1
[10] Swing GUI Example Diagram 2
[11] Swing GUI Example Diagram 3
[12] Swing Monthly Return Generation Draft
[13] Swing Monthly Return Output Draft

Iteration 5 --

Deployment Diagram