VermaMS

A Disease Tracking EHR for Ghana

Project Team:

Ebenezer Kwame Ampiah ekampiah@wpi.edu

Dominik Smreczak desmreczak@wpi.edu

Alexander Kasparek ajkasparek@wpi.edu

Project Advisor:

Professor Wilson Wong

Abstract

The goal of the project was to develop a disease tracking electronic health record (EHR) system for Ghana in order to improve efficiency within medical facilities and to increase the quality of patient care. There are only a few medical facilities that have implemented EHR systems, and even fewer with immunization and disease tracking. Our proposed system, VermaMS, stores a history of patient visits for every patient diagnosed with malaria, tuberculosis, or meningitis. Users of the application are able to view patient data graphically as well as generate a report that contains information on each patient's visit. VermaMS is intended to be used in conjunction with the current record tracking system used by a medical facility, but it could potentially be integrated with another system or developed into a full EHR system of its own in the future.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

In Ghana, 37% of the leading causes of death are related to diseases ("Ghana: WHO statistical profile", 2015). With about a third of the deaths involving diseases, one would expect that majority of medical facilities are tracking these cases in order to increase prevention. On the contrary, studies have shown that some recordkeeping systems in Ghanaian hospitals are not designed to collect information related to diseases at all (Teviu, et al., 2012). Even if medical facilities track disease related information, a majority of them use a paper-based recordkeeping system. This type of system often includes problems of its own such as slow file retrieval times, lack of data security, and no backups. By implementing an electronic health record (EHR) system as a solution, there is a great potential to improve the quality of patient care and to increase the productivity and efficiency of medical facilities.

As of 2014, there were only a handful of EHR systems used by Ghanaian medical facilities. A few of them have been implemented nationally to generate health information. However, most of these systems are confined to specific institutions, primarily to overcome the difficulties presented with paper based recordkeeping systems ("Towards eHealth 2.0 in Ghana: A programme and opportunities for private and public ICT initiatives," 2014). As a result, these early adopters of EHRs have experienced decreased patient wait time and an increase in revenue (Menachemi & Brooks, 2006). Nonetheless, further examination of these EHRs in section 2.4 display a lack of evidence that EHRs include records that aid in tracking the spread of diseases in Ghana.

The goal of this project was to develop a software application that stores records in a cloud based database of patient visits for every patient diagnosed with malaria, meningitis, or tuberculosis. There are three main features that our EHR system achieves:

1. Protecting the privacy of patient demographic and biographic information
2. Creating graphs on collected patient information for data analysis
3. Generating a summary report of a patient's visits for patients and other medical facilities

With these components, our EHR system will be compliant with Ghana's Data Protection Act and will be able to provide adequate information to other medical facilities. To accomplish these features, the project will follow the Agile methodology with the Kanban variation as discussed further in section 3. Section 4 discusses the project's software environment used to fulfill the requirements gathered in section 5. The background information in section 2 reviews Ghana's current EHR status along with the benefits and difficulties of implementing an EHR system. Our final application is discussed in sections 6 and 7. Section 6 explains the application's design and rationale while section 7 discusses the implementation and major changes encountered while completing the project. The application is assessed in section 8 by examining the accomplishment of the goals and requirements. The final sections 9 and 10 conclude our project with future work and a conclusion to summarize the outcomes.

2. Research

2.1 Patient Record Tracking in Ghana

Paper records have been the basis for record keeping by medical facilities across the world for hundreds of years ("The Casebooks Project", 2015). It was not until the 1960's that a man named Larry Weed proposed the idea of a Problem Oriented Medical Record (POMR) in the medical field ("Introduction: As medical care gets more and ... - nasbhc.org", 2009). A POMR is a medical record that records the health status of a patient in a problem-solving manner (Manuswath, 2015). The POMR was one of the first implementations of electronic records in e-health, which is defined as the use of Information and Communication Technologies (ICT) for health (WHO, 2014). Other forms of records began to spawn afterwards with minor variations between each other such as computer patient records (CPR), electronic medical records (EMR), and electronic health records (EHR). CPRs are database systems used to store and access patients' healthcare information ("Security of Computerized Patient Record") while EMRs contain patients' medical and treatment history (Garrett & Seidman, 2011). EHRs are larger than EMRs as they encompass all the information of an EMR and provide a broader view on a patient's health by collecting more data than what is recorded at the provider's office (Garrett & Seidman, 2011). In 1972, Regenstrief Institute was the first to implement a full EHR system establishing a new era for patient record tracking. The acceptance of EHRs by health care providers was minimal until 2000, when many physicians began to see an increase in quality of patient care along with faster and easier access to patient information when using an electronic system ("EMR: The Progress to 100% Electronic Medical Records", 2015). Because of the

effectiveness of EHRs, many new medical facilities in developed countries start with an EHR implementation. However, in countries such as Ghana, the use of EHRs did not start until 2014 ("Ghana adopts electronic health records for patients", 2014). A majority of medical facilities still use paper records.

In 2012, a scientific report evaluated the data collection management of four hospitals in Ghana: Korle-Bu Teaching Hospital (KBTH), Effia Nkwanta Regional Hospital (ENRH), Komfo Anokye Teaching Hospital (KATH), and Tamale Teaching Hospital (TTH). The findings were that KATH was the only hospital to have an EHR in place while the remaining hospitals were either in the process of implementing one, or did not have one at all. User acceptance was low and data maintainability was difficult for the hospitals in transition, and these added to the challenges of transferring from a paper record system to a full EHR system (Achampong, 2012). Despite the difficulties, the increase in quality of patient care and efficiency within the medical environment made it worth implementing one.

2.2 The Benefits of an EHR System

2.2.1 Increased quality of patient care

In order to increase the quality of patient care, errors in medical records need to be reduced. Illegibility is one of the most common medical errors as a study in 2002 showed that 39% of medical errors consisted of pharmacists being unable to read handwritten prescriptions ("A CALL TO ACTION: Safeguard Drug Administration Within 2 Years!", 2002). By recording patient information electronically, legibility will be improved and treatments can be given in a timely manner to improve the quality of patient care. Another study showed that clinicians using

paper records experience missing patient information in about 14% of visits. The reported cause was due to test results along with other data not being available when necessary (Smith, 2005). With EHRs, incomplete information can be tracked and outdated information can be reminded to be updated in order to prevent re-diagnosing patients. With the use of EHR systems, 82% of clinicians have reported an increase in quality of clinical decision making (DesRoches, et al., 2008).

2.2.2 Efficiency and Productivity Improvement

Reducing the time spent searching for patient records and reducing manual entry improves efficiency and increases productivity dramatically. Studies have shown that about 52.2% of clinics in developing countries with paper based medical records take up to an hour to retrieve a patient's record (Teviu, et al., 2012). This is largely due to duplicate health records and poorly organized storage rooms. Using an EHR reduces the patient wait time significantly as search queues are much faster. As a result, patient care is improved since physicians and clinicians have time to diagnose and treat more patients. A literature review on computer patient records (CPRs) have shown that clinicians who have used CPRs to access patient information have noticed a reduction in information retrieval time and an increase in efficiency for patient management (Erstad, 2003).

2.2.3 Protection of Privacy

One of the biggest issues with paper records is the security of patient information. A paper record system does not offer the ability to track which users have looked at certain records as well as when they have viewed it. Patient records may contain sensitive information such as abortions, diseases, and abuse which is kept private under the practice of patient-doctor

confidentiality (Rindfleisch, 1997). If this type of information were to be released, the

consequences for the patient could result in embarrassment, loss of health insurance, or loss of a

job ("Eliminate Handwritten Prescriptions Within 3 Years", 2000). For this reason, EHRs have a

huge advantage in security since all instances of viewing and modification of patient information

can be recorded and screened for proper user authorization.

2.2.4 Backups and Paper Reduction

During a natural disaster, paper records may be destroyed. Destruction of these records creates a

huge setback in a medical facility's data tracking and patient progression, as returning patients

would have to have their medical records rewritten. However, this is not necessarily the case

with EHRs, especially with cloud-based databases since all of the records are preserved in a

remote location. Furthermore, vendors of cloud services perform frequent backups of data, while

offering virtually uninterrupted access to the data, leading to increased robustness. However,

there is some risk involved in the case of a hacker attack, which would grant unauthorized access

to confidential patient information. A full EHR system reduces or eliminates the need to keep

paper records. A clinic in Eugene, Oregon reported a 90% reduction in paper after two years of

using an EHR system (Ewing & Cusick, 2004). By reducing the supply and printing costs,

medical facilities have seen an increase in profit (Menachemi & Brooks, 2006).

2.3 The Difficulties of EHR Implementation

2.3.1 Insufficient Resources

More information can be stored online than on paper for a given amount of space. Several EHRs

use the Internet or have features that take advantage of it. A study done on issues with e-health in

Ghanaian clinics found that Internet connectivity was ranked third out of ten major issues. These clinics often have limited bandwidth, experience low speeds, and incur high utility costs (Bedeley & Palvia, 2014). In these conditions, EHRs experience slowness and take longer to retrieve files (Durrani & Khoja, 2009) which negatively affects diagnoses and treatments (Stuchfield, Jagilly, & Tulloh, 2007).

Although Internet service is one issue, power is another. In 2012, Ghanaian clinics were not able to depend on the Electricity Company of Ghana to provide consistent power and power outages were common (Achampong, 2012). Without power, EHRs are not able to function. In one news article, the assistant of a Ghanaian clinic mentions that the clinic has had to repair medical equipment damaged by power outages (Adu, 2013). Since EHRs operate on physical devices, power outages can result in corrupt files which can lead to partial or complete failure in software functionality (Patkar, Price, & Lee, 2014).

2.3.2 Financial Instability

One major struggle that developing countries face in the medical field, as compared to developed countries, is the lack of financial support from government (Sood, et al., 2008). Because of this, many clinics do not possess Information and Communications Technology (ICT) resources and have a weak digital infrastructure. A weak ICT infrastructure was ranked the number one biggest problem with e-health in Ghanaian clinics. Issues in this area include limited number of computers, poor computer conditions, and out-of-date software (Bedeley & Palvia, 2014). Implementing an EHR system involves purchasing new hardware and software, installation, training, and maintenance, all of which can negatively impact the current financial situation of

15

any clinic (Bedeley & Palvia, 2014; Al-Shorbaji, 2008).

2.3.3 Inadequate Knowledge and Resistance

When EHR systems are implemented within a medical facility, a challenge arises in the area of proper knowledge and skills to use the equipment. A majority of professionals in Ghana lack the basic ICT knowledge and skills to produce effective results from using an EHR system (Bedeley & Palvia, 2014). This is partly due to the average Ghanaian lifestyle. Most Ghanaians live without computers or electricity and therefore have resistant feelings towards using technology (Bedeley & Palvia, 2014). Additional feelings of resistance come from changes in traditional medical practices (Acquah-Swanzy, 2015). Experienced health professionals do not want to change what they have been doing for years, especially if there is a threat of reduction of staff or a high difficulty in operation (Achampong, 2012).

2.4 EHRs in Ghana

As of 2014, there are only a select few EHR systems that are in use in Ghana ("Towards eHealth 2.0 in Ghana: A programme and opportunities for private and public ICT initiatives," 2014). Medical facilities that use these EHR systems are using a hybrid system between paper and EHR (Achampong, 2012). There are several reasons for this, one of which is that EHR systems are fairly new in Ghana. A study estimated that it would take a medical facility 10 - 15 years to migrate from a paper based system to an EHR system (Varga, 2011). Another reason why clinics have gone with a hybrid system is because of a requirement from the Ghana Health Service. This body of government has ordered that all medical facilities provide paper documentation for

services rendered for auditing purposes (Acquah-Swanzy, 2015). One of the EHRs used in a hybrid system is the Health Administration Management System (HAMS).

Swanzy (Acquah-Swanzy, 2015) performed an evaluation of HAMS, an EHR that is used by the Effia Nkwanta Regional Hospital (ENGH) in Ghana. HAMS was developed by the *InFotech Dot Net System Limited* (IDNS) company in Ghana. In 2012, the ENGH began to use HAMS in addition with its current paper based system. The hospital adopted the EHR primarily to manage health insurance forms as well as other general health records. As a result, the hospital was able to reduce the loss of paper from the manual procedure. The user-friendly interface made it easier to use for people with different computer backgrounds and was ultimately what made it the best choice when compared to other EHR systems (Acquah-Swanzy, 2015).

Another EHR system used in Ghanaian clinics is DocuTAP. DocuTAP is a company that was founded in South Dakota in 2000. The company created its own EHR system that is used in the Sanford World Health Clinics located in the Cape Coast of Ghana. Sanford Health is a non-profit health care system that started a World Clinic initiative. It was projected to construct 10 clinics in the Cape Coast of Ghana by 2016 and it partnered with DocuTAP as its primary EHR system for those clinics ("DocuTAP Implements Electronic Medical Records System for Sanford World Clinics in Ghana," 2012).

2.4.1 Health Administration Management System (HAMS)

HAMS is a client-server based EHR system. The front-end design was created with Microsoft

Visual Studio and Microsoft Dot Net Framework. HAMS is grouped into 11 modules, but only 5 are commonly used by the Effia Nkwanta Regional Hospital. The following modules are used: eFolder Management, Patient Registration and Records Management, Pharmacy and Dispensary Management, Pay Group and Service Charges Management, and Billing Management.

The eFolder Management module handles the filing and retrieval of patient records. It assigns unique identification numbers to files being stored within the hospital as well as in the EHR system. The Patient Registration and Records Management module records patient biographical information. It is used to simplify the processes of identifying patients and filing insurance claims. The Pharmacy and Dispensary Management module provides information on drug stock levels, expiration dates, and side effects. It is primarily used for recording prescribed drugs and dosages. The Pay Group and Service Charges Management module tracks patients and their corresponding medical insurance. It checks the validity of the patient under his or her insurance based upon the unique identification given by the insurance company. Finally, the Billing Management module generates patients' bills, invoices, and payments (Acquah-Swanzy, 2015).

2.4.2 DocuTAP

DocuTAP is a web-based EHR system designed to be used on a tablet for ease of use. A majority of information selection is done with checkboxes in tables which is added to a final report. The key design principle behind DocuTAP is to function the way physicians think and work ("GetApp," 2016). The concept was validated through the order of sections shown in a demonstration. The sections are as follows: Real Time Chart Room, History of Present Illnesses

(HPI), Review of Systems (ROS), Exam, Assessment, Plan, and Coding.

The Real Time Chart Room section lists patients that the physician is scheduled to see for the day. The list includes the status of the patient as well as the reason for visit. The patients can be flagged for discharge from this section too. Underneath the list of patients is a search bar for patient searching. The History of Present Illnesses section allows physicians to record symptoms shown by the patient. Additional details for each symptom can be selected such as: duration, frequency, and outcomes of currently used medication as shown in Figure 1. The Review of Systems section lets physicians record additional details from the visit into an imported template. Templates are the documentation that a physician has on the current visiting patient.

Figure 1 - A portion of the History of Present Illnesses section.

 In the Exam section, physicians document findings from examinations through a series of tabled checkboxes within tabled checkboxes. Additional tests and procedures can be scheduled if needed. In the Assessment section, physicians select which diagnosis they would like to perform.

All of the diagnoses are listed in an alphabetical table menu as shown in Figure 2. First the

physician selects the letter that the diagnosis begins with and then selects the desired diagnosis.

The Plan section allows the physician to select an order set. An order set is a recommended

treatment and dosage which is determined from previous patient visits and other similar

diagnoses. Physicians can either select the recommended treatment or create a new one. The

Coding section recommends an evaluation and management (E/M) code for the user to select the

proper form of billing based on the visit results. Physicians can either agree and select the

recommended code or disagree and select a different one (DocuTAP, 2016).
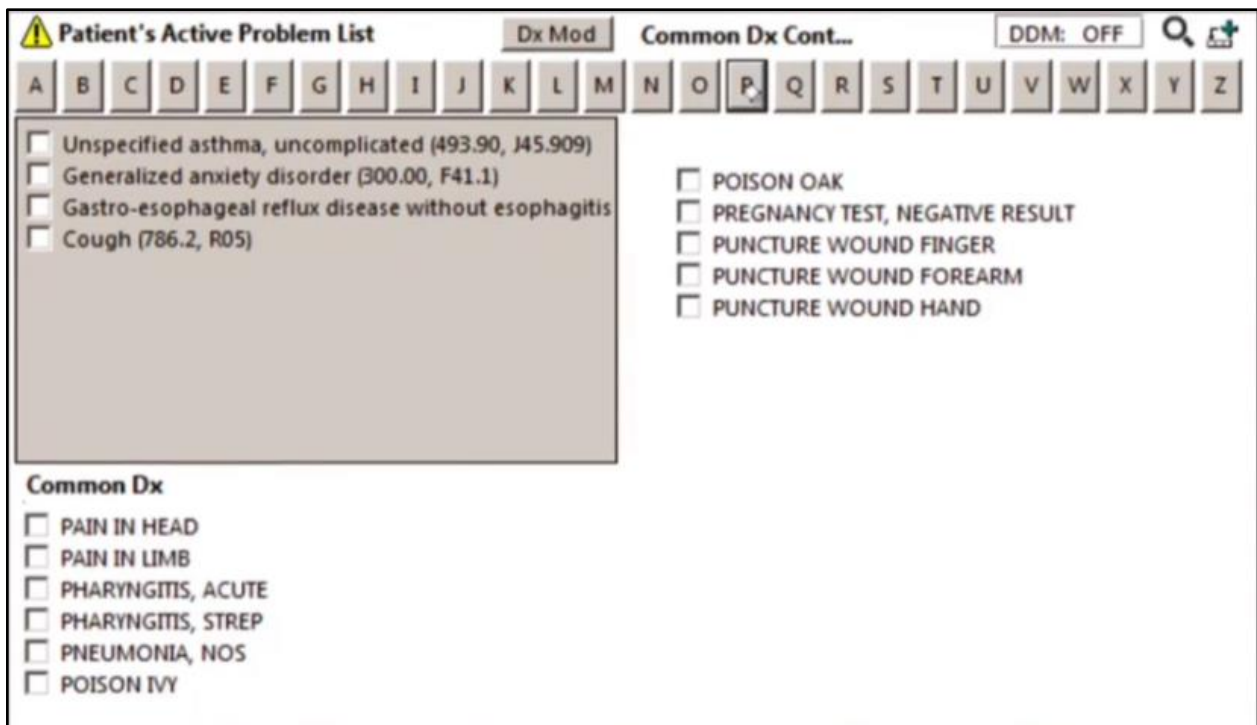
Figure 2 - The Assessment section

2.5 Diseases in Ghana

2.5.1 Malaria

In Ghana, malaria is ranked third for the most deaths killing about 17,000 people in 2012 ("Ghana: WHO statistical profile", 2015). The majority of these deaths are children, pregnant women, and travelers without previous exposure to malaria. In these cases, the lack of immunity to malaria is what leads to a higher chance of death. There are five different species of malaria, with *plasmodium falciparum* being the deadliest. 90% of the malaria cases in Ghana are *P. falciparum* while the other 10% is *P. ovale* ("Yellow Fever & Malaria Information, by Country", 2015). Although *P. ovale* is not considered a life threatening species, it can cause relapses if not treated thoroughly.

Malaria is caused when parasites infect red blood cells. It is transmitted by mosquito bites and cannot be diagnosed solely on the symptoms a patient exhibits. Aside from *P. falciparum* which has more severe symptoms, the other species of malaria display similar symptoms to that of the flu and a microscopic test is necessary to confirm that the cause is malaria. Other tests such as Rapid Diagnostic Tests (RDTs) are used in the case that a microscopic test cannot be performed. However, all results from RDTs should be confirmed by a microscopic test. Microscopic tests are the standard for confirming malaria in patients as it involves looking for the parasites in the red blood cells, therefore giving a more accurate result ("Malaria Diagnosis (United States)", 2015).

Once a patient is diagnosed with malaria and the species is determined, the stages of assigning the correct treatment can begin. Several factors determine what kind of treatment a patient

receives but the most influential are:

- Area where the infection was acquired

- The drug resistance of the species

- Health status of the patient

- Drug allergies

- Other medication taken by the patient

The most successful drug in treating malaria has been chloroquine. A patient must only take four doses within four days. However, since all types of malaria found in Ghana have developed a resistance to chloroquine ("Malaria Information and Prophylaxis, by Country [G]", 2016), the prevalent drug is atovaquone-proguanil (Malarone®). The same dosage is given as with chloroquine, four tablets of Malarone®, once a day every day for four days ("Guidelines for Treatment of Malaria in the United States ...", 2013). According to our survey results, a majority of malaria patients administer the medication themselves and do not need a follow-up appointment, although a follow-up would be ideal.

2.5.2 Meningitis

Meningitis is a contagious disease which can be easily transferred through coughing. In 2015, outbreaks of meningitis have occurred in Nigeria and the northwestern parts of Ghana. Out of the 205 meningitis cases, 23 deaths had been reported in Ghana ("Highly contagious meningitis outbreaks continue in African countries"). As of 2010, meningitis was ranked 10th in leading causes of death in Ghana ("HIV/AIDS CDC in Ghana", 2013). Each case is treated as a medical emergency due to the high risk of death in a short period of time ("Highly contagious meningitis

outbreaks continue in African countries").

Meningitis is the inflammation of the membranes surrounding the brain and spinal cord. There are five different categories of meningitis. The most common in Ghana is the bacterial form, Neisseria meningitidis. The disease shows common flu symptoms such as a fever, headache, and stiff neck which can cause difficulty in diagnosis. Treatment is generally initiated before the diagnosis due to the high rate of fatality (Theobald, "Understanding the 5 Types of Meningitis, 2014).

The standard diagnosis for meningitis involves a lumbar puncture, also known as a spinal tap. This process involves taking a sample of the patient's cerebrospinal fluid in order to determine the presence but more importantly the type of meningitis. Other early diagnoses include a CT scan, which looks for abscess in the brain, and blood tests, which can signal an infection. Along with the type of bacteria, the patient's age as well as the severity of his or her illness determine which antibiotics are prescribed (Theobald, "How Meningitis Is Diagnosed in Its Early Stages", 2014).

The two most effective prescribed antibiotics for bacterial meningitis are ampicillin and cefotaxime (Claforan). These antibiotics are generally given by injection and are frequently combined with others in order to eliminate the bacteria ("Antibiotics for Bacterial Meningitis", 2014). Although treatment can cure a patient of the disease, the best way to prevent meningitis is by receiving a vaccination. A campaign for meningitis vaccination began in 2010 where over 3 million people in Ghana were inoculated. The vaccine used in the campaign was MenAfriVac, and it prevents Neisseria meningitidis type A from the ages of 1 year to 29 years ("WHO |

Meningitis vaccine provides hope to people in Ghana", 2012). Although this eliminates approximately 90% of meningitis cases ("Meningococcal Disease in Other Countries", 2016), other variations of the disease continue to spread and need to be treated with medication.

2.5.3 Tuberculosis

Tuberculosis (TB) is an infection caused by the bacterium *Mycobacterium tuberculosis ("*TB terms," 2016*)*. It most often affects the lungs, but in the worst cases it spreads to other areas of the body ("Tuberculosis (TB)," 2016). The bacteria that causes the disease is spread through the air by an infected person when that person coughs ("How TB spreads", 2016). Ghana, despite national efforts to eradicate the disease, recorded 122 new cases every day as of March 2016; however, early detection and treatment has generally proven successful ("T.B. cases increasing," 2016). Tuberculosis still ranks as one of the leading causes of death in the country, with a mortality rate of 37 in 100,000 in January 2016 ("Ghana: Tuberculosis Profile," 2016). Patients who contract HIV are prone to also contracting and dying from tuberculosis, and this caused 19 deaths per 100,000 people in 2015. ("Ghana: Tuberculosis Profile," 2016).

Some common symptoms include: a cough that may bring up blood, fever, loss of appetite, and night sweats ("Tuberculosis (TB)," 2016). There are two categories of tuberculosis infection, TB disease and latent TB infection (CDC, 2016). TB disease occurs when TB bacteria are actively attacking the body, and latent TB occurs when bacteria are present, but inactive (CDC, 2016). Latent TB can become active at any time ("Tuberculosis (TB)," 2016). TB can be prevented with a vaccine, and if contracted it can be treated with antibiotics ("Tuberculosis (TB)," 2016).

There are two tests that check for the presence of *Mycobacterium tuberculosis*, the Tuberculin

Skin Test (TST) and Blood Test. According to the United States CDC (2016), in order to perform a TST, a healthcare provider will inject tuberculin (a liquid) into a patient's lower arm. In two to three days, the patient comes back in and the physician then looks at the area of the injection for a "raised, hard area or swelling and if present, measure its size using a ruler." If the area is a certain size, the test is positive and additional tests are needed to determine whether it is latent or active. If a patient has received a TB vaccine, this test may show a false positive. In order to perform the Blood Test, blood is collected from the patient, where it is sent to a lab to detect the presence of TB. Such as with a TST, if the result is positive, additional tests are needed to determine whether it is latent or active. After a positive test for the presence of TB bacteria, a second stage of testing, such as a chest x-ray, is performed to determine if the bacteria are active ("Testing for tuberculosis (TB)", 2016).

3. Methodology

There are two main paradigms for software development; the traditional software development life cycle, or the more contemporary Agile methodologies. This section discusses these areas in more detail, presenting the reasons for our choice in methodology.

3.1 Traditional Software Development Life Cycle

The Software Development Life Cycle (SDLC) has been the industry-standard approach to building high-quality software. The SDLC organizes development into sequential phases, with each phase usually forming the input for the next. This sequential nature is the reason why it is sometimes referred to as "a waterfall," and implies that the only motion is forward ("SDLC - overview," 2016). Consequently, it is expected that all work on any particular phase is completed before the next.
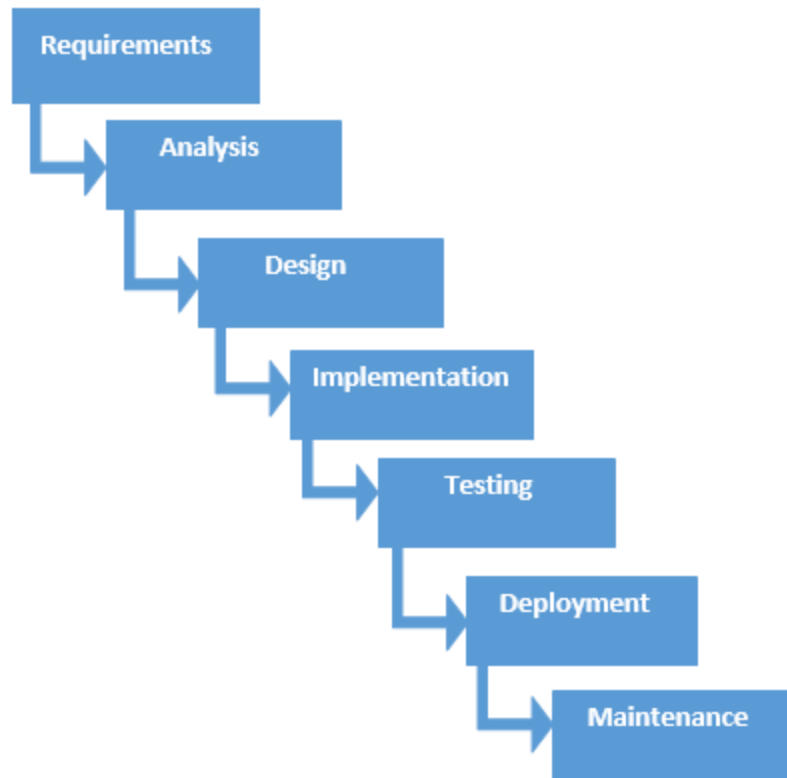
Figure 3 - Visual depiction of the Software Development Life Cycle

3.1.1 SDLC Stages

Figure 3 is a visual representation of a generic SDLC model, showing how each step in the

process feeds into the next (Hussung, 2016). Although the specific terminology for each phase

varies among teams, these are the core steps performed. Each phase is discussed in more detail

below.

Stage 1: Planning & Requirements Gathering

This step encompasses the activities pertinent to project initiation, such as team formation and

decisions on the scope and objective(s) of the project. In order to make the most informed

decisions at this step, the team must engage in rigorous requirements gathering. The

requirements gathering process can involve many steps such as literature reviews, surveys, and interviews in triangulating an application's true software requirements (Shelly et al., 2009).

Stage 2: Analysis

In this phase, all the information gathered in the planning stage is collated and organized. Then, each piece of information is used to develop the functional and nonfunctional requirements for the system. Questions like "Who is going to use the system?" and "How are they going to use the system?" are answered here.

The resulting set of functional and nonfunctional requirements will serve as the basis for developing the Analysis Model. Also called the Domain Model, this is a primarily visual document representing the entire system to be implemented, using real world terms. Since it is meant for all stakeholders to be able to understand, everyday language is used rather than technical terms. A common model used in this phase is the use case diagram, which visually depicts interactions between actors (consumers and producers of information). Activity diagrams, which graphically illustrate scenario-specific interaction flow, often supplement use case diagrams (Pressman, 2014).

Stage 3: Design

In this stage, the team develops an Object Model for the system. An Object Model is a document detailing the exact design to be implemented, using computer-specific terms such as classes and objects. This document contains various diagrams including, but not limited to: class, sequence, and state diagrams. (Justin, 2013).

.

Stage 4: Implementation

At this stage, developers attempt to implement the features noted in their requirements documents using their preferred coding tools. This code-intensive stage is one of the longer periods of the development cycle, possibly extending up to a few years.

Stage 5: Testing

During this stage, developers perform various automated and manual tests on the product, to ensure its adherence to the requirement specifications. Dynamic testing is done based on a predetermined test strategy, which is the team's chosen approach to ensure methodical and thorough tests. There are several types of tests, each with a specific purpose. For example, unit testing is used to evaluate individual components of the application, and is usually done by programmers. This is commonly followed by integration tests, where different modules are combined to have their collective functionality tested. Other testing schemes such as regression and acceptance tests are also often performed, to identify modifications to any module and ensure customer requirements are met, respectively. (Justin, 2013).

Stage 6: Deployment

In this stage, the team packages and ships the application to users. After ensuring the integrity of the product through testing, the team must decide how to deploy the application to the end user. This is the crux of the deployment step, and determines the team's distribution medium.

Stage 6: Maintenance

After getting the product into the client's hands, the team must ensure that it continues to work for a period of time agreed to by both the client and software development team. This step is also

essential because software is almost guaranteed to have errors and the development team needs a plan on how to deal with the problems that arise. Furthermore, teams must adapt the application to changes in the real world, as necessary; a task done in the maintenance stage. (Shelly et al., 2009).

3.1.2 Evaluation of SDLC

Advantages

The phases and guidelines of the SDLC are explicitly stated, allowing for the use of less experienced staff for software development. Furthermore, by being explicit with each step, the methodology promotes consistency across projects, allowing staff transfer between projects. Also due to the clearly structured phases, SDLC teams are able to better predict and meet deliverable deadlines; thorough planning is done for each step, accounting for contingencies

Disadvantages

For most projects, the SDLC is overly complicated, as it was designed to solve large, complex project issues. This is a hindrance to productivity, and can also result in a pile of redundant documentation. Additionally, the waterfall or "river of no return" model is infeasible, given that teams are highly unlikely to gather all requirements at the beginning of a project. Furthermore, the customer is involved occasionally, rather than being a full participant in the development process. This could lead to misinterpretations in requirements which could be left uncaught for too long. Sometimes, business changes invalidate the initial system design by the time of release, especially for long-term projects. With frequent involvement from clients, the problem of

development going off-track could be circumvented at an early, relatively inexpensive stage.

3.1.3 Variations

There are some variations of the SDLC methodology which allows backtracking to the previous

step in the model. This is to enable the project team to adapt to changes that are bound to arise

throughout the lifecycle. For those processes, the diagram is more similar to Fig 4.
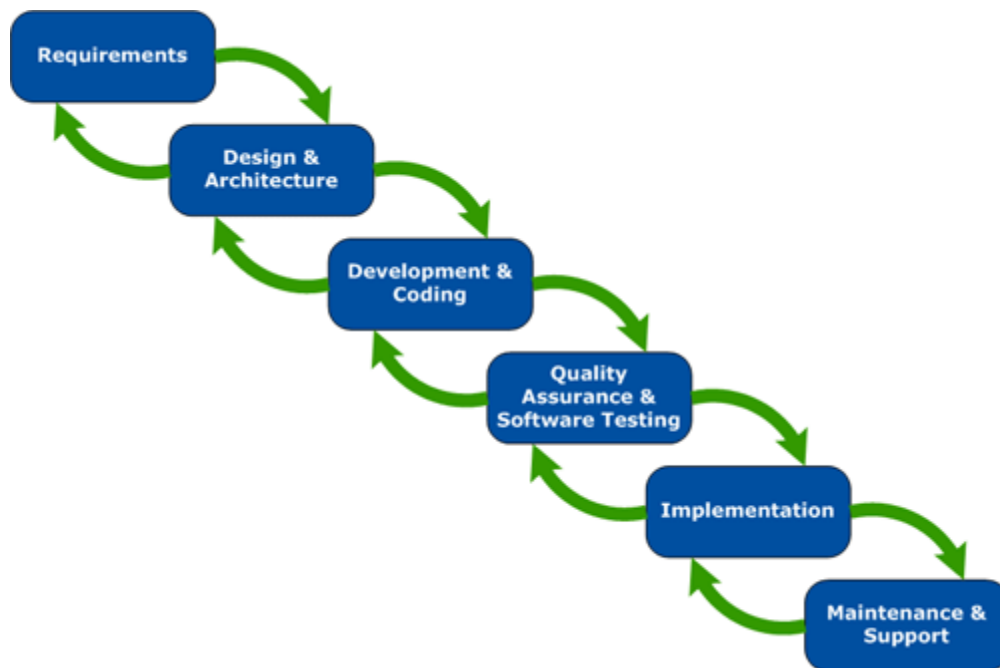
Figure 4 - Waterfall model with "backtracking" (Valentine, 2012)

3.2 Agile Methodologies

The Agile approach was developed to counteract the limitations of the traditional waterfall

methodology, particularly those brought on by over-complexity and lack of flexibility of the

process (Agile & Waterfall Methodologies - A Side-By-Side Comparison, 2017). Paramount was

the loss in productivity to practically meaningless tasks, such as heavy documentation.

Furthermore, the lack of frequent stakeholder input often led to disparities between stakeholder expectation and software reality (Shiotsu, 2017). In searching for solutions to the problems of the SDLC, developers adapted the manufacturing industry's "Lean Manufacturing techniques" to form the Agile software development techniques. Being agile denotes "flexibility and responsiveness within a well-defined context", with a strong emphasis on results and the business value of each step in the process (What is agile? What is Scrum?, 2013).

The Agile approach is not limited to a single methodology, but rather a collection of methodologies. These methodologies are all based on iterative development, in which requirements can evolve through constant collaboration between self-organized teams, but only within a well-defined scope. The aptly-named Agile Manifesto governs Agile development, and it reads (Beck, et al., 2001):

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

> *Individuals and interactions over processes and tools*
>
> *Working software over comprehensive documentation*
>
> *Customer collaboration over contract negotiation*
>
> *Responding to change over following a plan*
>
> *That is, while there is value in the items on*
>
> *the right, we value the items on the left more."*

We will focus on two main Agile approaches; Scrum and Kanban. However, before diving into

specifics, below are the features of Agile methodologies that make them more attractive for an increasing number of teams.

3.2.1 Key benefits of going Agile

Stakeholder Involvement
A stakeholder refers to any person who will be involved with the application, whether during development or after deployment. The Agile approach strongly emphasizes constant collaboration between all stakeholders, particularly between clients and project teams, to ensure that the product meets expectation (Shiotsu, 2017).

High Flexibility
As mentioned above, stakeholder input is key at each step of the Agile approach, i.e. each iteration. As such, clients are able to communicate any feedback on development over short time periods. This allows the team to adapt quickly to minute changes, rather than present a totally unsuitable application after tedious development (Agile & Waterfall Methodologies – A Side-By-Side Comparison).

Timely Delivery Schedule
This pertains more to the fixed-length cycles of Scrum, where development is limited to a fixed amount of time, usually 2 weeks. Consequently, feature development is done in sizeable chunks, which fit in the development schedule as enforced by the project manager (What is agile? What is Scrum?, 2013).

Focus on Business Value
By making sure that every step taken is crucial to the development process, teams can avoid any

worthless or redundant tasks. This focus ultimately increases team efficiency and productivity

(Agile & Waterfall Methodologies – A Side-By-Side Comparison).

3.2.2 Scrum

Scrum is an application of Agile methodologies, tailored to projects with rapid development

cycles, and characterized by fixed-length iterations. It is the most widely utilized approach, and

places a strong emphasis on project management. In order to ensure a smooth development

process, there are three key roles associated with Scrum; the Product Owner, Scrum Master, and

the development team. (What is Agile? What is Scrum?, 2013).

Scrum Roles

Product Owner

The product owner is responsible for the product requirements, and must know them inside-out

to be successful. He or she must maintain the product backlog, which is a complete list of

features to be implemented. They are also responsible for the iteration backlog, which is a

prioritized list of user stories to be implemented for the current iteration, and is determined from

sprint to sprint. A sprint is the period allocated to each iteration of development, and is typically

a couple of weeks. Figure 5 presents a visual depiction of the scrum development cycle.

Scrum Master

This team member is placed in charge of the scrum process, with the sole responsibility of

ensuring its smooth running. During each sprint, the Scrum Master deals with any impediments

to development via a daily check-in team meeting called the daily scrum. These meetings are

held daily, as the name suggests, preferably at the same time and location if possible. They are

usually limited to 15-minute periods during which each team member reports activities after the last sprint, and sets the context for the upcoming one (The Daily Scrum Meeting). The scrum master also has the following additional duties:

- Remove any impediments to the process, e.g. ineffective development tools

- Organize and facilitate critical meetings such as the scrum planning, review and reflection meetings.

- Increase productivity by facilitating creativity and empowerment

- Document team progress
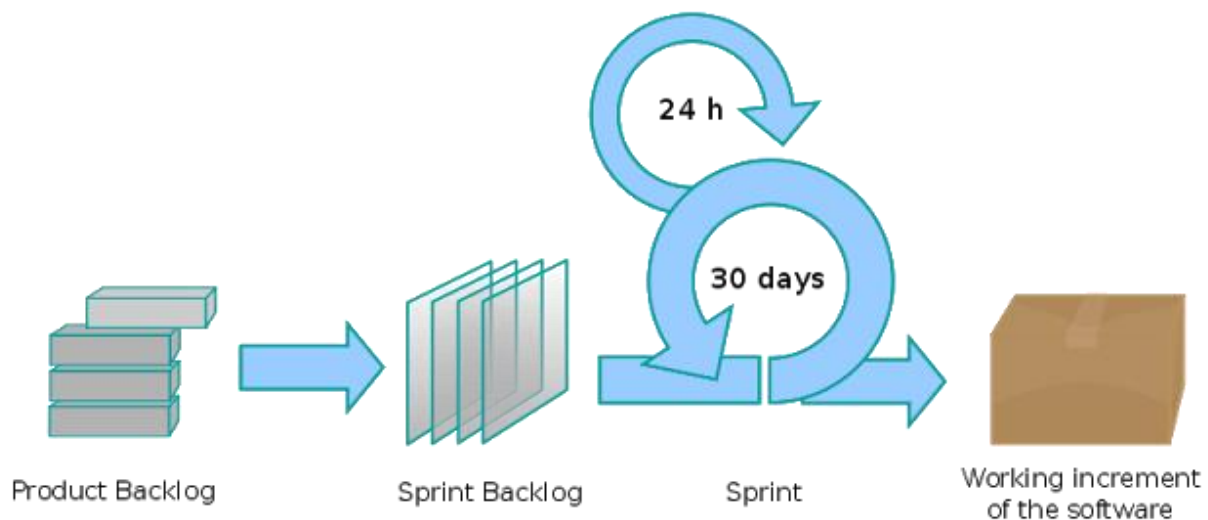
Figure 5 - Scrum Development Cycle (Gupta, 2014)

3.2.3 Kanban

Kanban is another application of Agile methodologies that has recently seen a surge in popularity. It was adapted from the "just-in-time" (JIT) manufacturing model at Toyota motors,

which was in turn borrowed from the supermarket industry. The model is simple; stock just enough goods to meet customer demand. This prevents inventory-related issues by minimizing excesses, while serving the needs of customers. To understand how teams adhere to Kanban, it is essential to discuss Kanban boards, an integral part of the approach (Atlassian, 2016).

Kanban Boards

This is a physical or digital tool used to visualize and optimize the team's workflow. A basic Kanban board consists of a 3-step workflow categorized under "To Do", "In Progress" and "Done". However, as in Figure 6, intermediate steps may be included to match the needs of the team. Using this board, teams follow JIT principles by matching the amount of work in progress (WIP) to the team's capacity. For our sample board, WIP would include both "in progress" and "code review" columns.
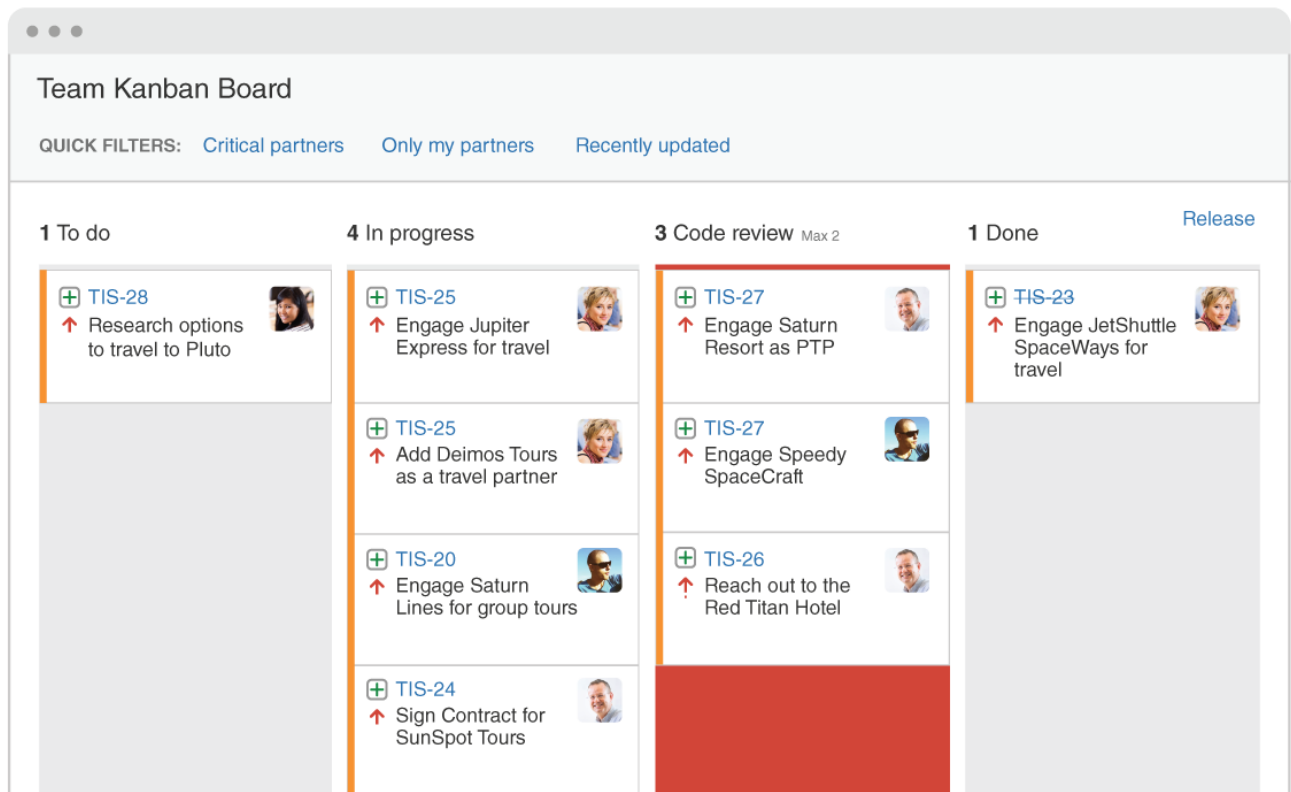
Figure 6 - Typical Kanban board for software teams (Atlassian, 2016)

Benefits of Kanban

Extreme flexibility in planning

In Kanban, developers simply pull the next item off the product backlog and implement it, paying no attention to anything else. As such, product owners are able to freely organize the backlog in a way that makes practical sense for the team, without consequence to the team's work.

Limited bottlenecks; shorter cycle times

The amount of time it takes a unit of work to go through the entire workflow process - from start to finish - is known as its cycle time. This is one of several key metrics for Kanban teams, enabling teams accurately predict future delivery times.

As a preventive measure, Kanban teams employ work-in-progress (WIP) limits to different stages in the workflow process. For example, a team may place a limit of 2 items under "code review," signifying that before any work items can be added, the current ones must be dealt with. If it becomes necessary, work in other steps can be halted, to get all hands on deck to deal with a bottleneck, cutting down on overall cycle time.

Continuous delivery

Continuous delivery (CD) refers to the practice of releasing products to customers regularly and frequently. This is only possible by a highly flexible team that can quickly respond to customer needs, making Kanban with its JIT principles the perfect solution.

	SCRUM	KANBAN
Cadence	Regular fixed length sprints (ie, 2 weeks)	Continuous flow
Release methodology	At the end of each sprint if approved by the product owner	Continuous delivery or at the team's discretion
Roles	Product owner, scrum master, development team	No existing roles. Some teams enlist the help of an agile coach.
Key metrics	Velocity	Cycle time
Change philosophy	Teams should strive to not make changes to the sprint forecast during the sprint. Doing so compromises learnings around estimation.	Change can happen at any time

Figure 7 - Table showing the differences between scrum and Kanban (Atlassian, 2016)

As a team, we used Agile methodologies, specifically Kanban principles, in our development. This was primarily in order to ensure maximum flexibility. As a WPI team, we are well aware of the various issues that confront teams, standing in the way of meetings or other productive activities. We believe that by using the Kanban approach, we will be able to work efficiently by constantly picking work items, without having to wait on other teammates. Contrast this to Scrum or SDLC, for example, where a more productive teammate would have no choice but to wait until the sprint or stage is over in order to procure new work items. This continuous workflow will be fostered by constantly keeping the entire team up to date with any changes and decisions through Slack (Section 4.2.1). We will also employ Trello boards to maintain our backlog, which will not only serve as a visual aide to project management, but will serve as a reference for team members to know who is working on what.

4. Software Development Environment

In this section, we will discuss our development environment. These tools and processes were carefully selected to provide a streamlined solution that would help speed up development. The decisions for each of these were based on several areas including cost, feature support and past experience. Although we strived for a fully integrated development, some of the criteria constraints prevented this.

4.1 Project Management Software

There are numerous project management software, available, each with their own pros and cons. The most popular options we discovered were Jira, Trello and Wrike, which we explore below.

4.1.1 Jira Software

Jira Software is a web-based issue and project management tool and also comes with a desktop version. Starting out as an issue tracker, the application has evolved into a comprehensive project management tool with a strong Agile focus. As a result of this, there are a number of features that are tailored to Agile development such as Kanban and Scrum boards (Atlassian, 2016). The software can be used either as a cloud-hosted version which is managed by Jira, or an in-house dedicated server.

There are a few advantages to using the cloud-based version. To start with, JIRA on the cloud is incredibly easy to install and configure, and can be set up with just a few clicks. This rapid

instantiation with the basic settings is sufficient for most users. JIRA also offers monthly

subscription pricing, which enables a team to pay for it only when needed. In addition, there are

many third-party applications that are easily integrated into the web-based version to offer an

improved user experience (Atlassian, 2016).

On the other hand, the self-hosted option also has its perks. In this case one can choose to host

his or her JIRA software on a private server, which gives complete control of the development

environment. This version requires only a one-time fee as opposed to the cloud's per-month

billing. Nevertheless, its web version also comes with add-ons to improve user experience. This

option is more suitable for teams with greater experience with server administration as well as

project management (Atlassian, 2016). Below, we compare the features of both versions of JIRA

software.

FEATURE COMPARISON	CLOUD	SERVER	DATA CENTER
Project and issue tracking	✓	✓	✓
Scrum and kanban support	✓	✓	✓
Backlog prioritization and sprint planning	✓	✓	✓
Flexible workflow	✓	✓	✓
Developer tool integrations	✓	✓	✓
Out-of-the-box agile reporting	✓	✓	✓
Rich APIs	✓	✓	✓
Plug-and-play add-ons	✓	✓	✓
Active-active clustering			✓
Disaster recovery			✓

Figure 8 - A comparison of features offered by the different hosting options (Atlassian, 2016)

Now we discuss certain areas where Jira Software falls short. For teams limited to the cloud

version, the JIRA environment does not lend itself to full customization. As such, one would

have to use the tool with default installation settings, no matter how incompatible one is with it.

Furthermore, the cloud's monthly billing might not be suitable for some teams, especially

smaller ones, making it hard to maintain its use. The pricing chart is shown in Table 1. Teams

that would rather use the self-hosted version would need to invest in gaining the requisite

expertise to manage the server. This could end up costing the team much either in time or

salaries.

Number of users	Price ($/month)
0-10	10
15	75
25	150
50	300
100	450
500	750
2000	1500

Table 1 - Jira Pricing Chart (Atlassian, 2016)

4.1.2 Trello

Trello is another web-based tool which is primarily built for team collaboration, but it has

excellent project management capabilities. It features an easy-to-use card system, where tasks are

placed on individual cards and can then be assigned to team members. These cards have

associated boards, which are designed to represent projects but can be organized as desired due

to its vague specification. Trello allows team members to see what is being worked on and the status of those tasks by organizing cards into lists. Typical categories include: TODO, In-Progress, Done, Review. This helps the team keep up-to-date on how much functionality has been completed within a given iteration and allows it to more effectively plan the next iteration.

In this section, we will take a look at the key features of Trello. Although seemingly basic, Trello offers a rich, customizable visual interface. This means that colors and shapes are designed to a team or individual's specifications. The customizable interface enables the use of color-coding as well as other productivity aids to improve team efficiency and output. (PCMag, 2016). Furthermore, due to its inherent cloud-based nature, teams have access to their boards at any time from a variety of devices. It is available on PC via a web app, and can be used on several mobile devices including Android and iOS devices. Many users also find Trello extremely easy to use with its intuitive interface. This ease of use is further enhanced by the integration of third-party applications through Zapier, which is itself a third-party app dedicated to connecting various cloud-based applications to facilitate information flow. Most importantly for our project, Trello is free to use although special features are only accessible with a premium membership.

4.1.3 Wrike

Wrike is a project management tool that has increased in popularity since its inception in 2007. The company chalks up this success with claims of easy collaboration and efficient project management with features such as dynamic timeline (Gantt chart), dashboards and real-time reports, among many others. Pricing is structured into different tiers, as discussed below (Hillsberg, 2015).

Free Tier (up to 5 users); $0.00/month

Despite shipping with limited project management features, this tier is suitable for very small teams such as our project. In the free tier, Wrike offers task management, similar to Trello boards, as well as file-sharing capabilities. It also provides 2GB of storage, as well as cloud storage integrations with Google Drive, Dropbox, Box, and Microsoft OneDrive. These options allow users to easily save and retrieve any shared files from within the Wrike console. To enhance collaboration, Wrike's free tier has basic integrations with popular services such as Google Drive and Office 365. Finally, all these features come in Android and iOS applications, available to every tier (Hillsberg, 2015).

Paid Tiers

Wrike also provides a number of paid subscription levels for teams of different sizes. Each tier builds on the features of those below, which the following table summarizes. Were we to choose a paid tier, the most cost-effective would have been the Professional tier. Besides being the cheapest offering, it would provide us with all the essential features, and enough user accounts for each member of the team. Moreover, the more expensive tiers provide many services which would be redundant to our development and would therefore be a waste.

Tier (number of users)	Cost ($/month)	Key Features
Professional (5, 10, 15)	9.80	- Dynamic Timeline (Gantt Charts) - Advanced Integrations (MS Project, Excel, RSS) - Sharable Dashboards - Unlimited Collaborators 5 GB storage

Business (5 - 200)	24.80	• Real Time Reports • Resource Management • Time Tracking • Custom Fields & Workflows • User Groups & Permissions • Salesforce Integration • 50 GB Storage
Enterprise (5-unlimited)	special pricing	• Active Directory Integration • SAML 2. Single Sign On • Two-Factor Authentication • IT Controlled Admin Permissions • User Audit Reports • Network Access & Compliance Policies • 100 GB Storage

Table 2 - Table presenting the features at various paid tiers of Wrike

Cons

Unfortunately, Wrike is not as heavily customizable as Trello. This makes the learning curve a little steeper for some users. This is especially troublesome for this application since it is feature-rich, even at the basic level. Such conditions can easily create a workspace with a cluttered feel. Although Wrike provides mobile apps for Android and iOS, there are none for desktop platforms, which could deter some potential users. (Hillsberg, 2015).

4.2 Team Communication Software

Communication is vital to any team's success. In an Agile environment, communication becomes even more crucial with an increased reliance on smooth and regular information flow among teammates. With a variety of applications to choose from, we selected a handful of the most popular to analyze. We categorized these into two groups; those made with teams in mind (Slack and Skype), and general-purpose chatting applications (Facebook Messenger, WhatsApp

and GroupMe).

4.2.1 Slack

Slack is a real-time team communication tool. It organizes conversations into channels, allowing teams to create multiple public and private ones for different purposes. These channels can be between just two people, or opened up to groups of varying sizes as necessary. It is extremely popular among teams, especially software teams (Slack).

Key features

Slack channels provide a unique user experience. To begin with, teams are unrestricted in how many channels can be created. Further, users are free to participate in any number of public channels, and private channels to which they are granted access. This creates an intuitive way to organize communication, allowing teams to easily develop an efficient system of transmission (Slack).

Users are able to share code snippets or entire source files via a built-in code viewer. In addition, members can share files via a simple drag-and-drop. Slack then provides extensive search, even to the contents of the shared documents. In addition to its advanced sharing capabilities, Slack comes with features such as comments and reactions to message items, providing teams with a fully comprehensive communication tool (Slack).

Further, Slack offers several integrations essential to software teams such as Trello and GitHub, among others. All of these amazing features are available to use for free for a team of any size.

4.2.2 Skype

Skype is a video chat application that has evolved over the years to serve business needs. At its core, Skype allows registered users to make free video calls to each other, individually or as a group. Skype also offers instant messaging and file-sharing functionality, as well as the ability to make low-cost calls to phone numbers. After its acquisition by Microsoft, the Skype Business platform has surged with a myriad of features tailored to business needs (Skype, 2016).

We now take a closer look at the various features of Skype's business offerings, which comes in two varieties. Skype Meeting, which is free to use, unlike its paid version, Skype Business.

Although free to use, Skype Meeting has several useful features to cater to business needs. For starters, Skype meeting provides free unlimited meetings for teams of up to 10 persons. More tailored to business needs, Skype Meeting enables PowerPoint presentations as well as a whiteboard for collaborative functions such as drawing and critiquing on a universally accessible virtual whiteboard.

Skype Business builds on the services provided in the free version. By purchasing it with an Office 365 subscription, you get access to the suite of Office apps with enhanced integration capabilities. For instance, you can schedule Skype meetings in an Outlook calendar and receive reminders as necessary. This is also able to handle a multitude of users; up to a 250-person meeting, and a 10k-people Broadcast. Microsoft also supplements accounts with a hefty 1TB of cloud storage.

4.2.3 Group Chat Apps

In our search for a communication tool, we came across several platforms that offered similar

capabilities and were essentially direct competitors in their fields. Since these applications are

not tailored to businesses as those previously discussed, we categorized these applications as

"group chat apps". Table 3 explores the differences in the most popular of these platforms

(Khalid, 2016; Plutis, 2016).

Feature	Facebook Messenger	Whatsapp	GroupMe
End-to-end encryption	✗	✅	✗
Two-way opt-in	✗	✅	✗
Requires SIM card	✗	✅	✗
Document sharing	✅	✅	✅
Voice calling	✅	✅	✗
Video calling	✅	✅	✗
Free SMS	✗	✗	✅
Price	Free	$1/anum	Free

Table 3 - Feature comparison of Facebook Messenger, WhatsApp and GroupMe (Khalid, 2016; Plutis, 2016)

We now evaluate these software based on security of transmission, sharing capabilities, and

calling service. As the only platform on our list with end-to-end encryption, WhatsApp offers the

most secure transmission medium. Hence, for situations that require utmost privacy, this would

be the most ideal option.

All three apps have document and photo-sharing capabilities, but Facebook has quite the

advantage in this arena. WhatsApp and GroupMe are currently limited to the user's storage

restrictions, which could be as low as 10 photos per conversation. However, since Facebook Messenger belongs in the Facebook ecosystem, it is able to tap into Facebook's photo storage, removing any such limitations.

Despite both Facebook Messenger and WhatsApp offering video and voice calling capabilities, WhatsApp has the superior service from our personal experience, with better clarity over slow Internet speeds. Having started off serving international communities, WhatsApp has been optimized to work with low-bandwidth networks such as 3G and Edge, and with low signals. It provides a smoother calling experience than its competitors, especially in international communities.

4.3 Repositories

Every software project needs storage not only for code, but any forms of documentation that might be generated. To maximize team productivity, we opted to take advantage of the best technologies available, primarily based on reliability and ability to work collaboratively. We selected a couple of the most popular options for consideration; Google Drive and GitHub.

4.3.1 Google Drive

Google Drive is cloud storage that everyone with a Google account has access to. One of its main offerings is a cloud-based suite of productivity applications which are the equivalent of Microsoft Office. These applications support collaborative editing with updates made in real-time. There are also a host of third-party applications that are easily integrated to perform specialized tasks, such as Draw.io for diagrams. Plans start at 15GB for free, and go up to 30TB. The pricing chart is shown below.

Storage	Price
15 GB	Free
100 GB	$1.99
1 TB	$9.99
10 TB	$99.99
20 TB	$199.99
30 TB	$299.99

Table 4 - Google Drive price chart (Lardinois, 2014)

4.3.2 GitHub

GitHub is a service which offers free code hosting, providing version control and collaboration.

Currently, GitHub offers free hosting for public and open source projects, however, a paid plan is

required for private repositories. Nevertheless, starting at $7/month, it is relatively inexpensive.

To understand GitHub's functions, it is essential to explore the two main forms of version

control: centralized and decentralized (GitHub, 2016).

Centralized Version Control System (CVCS)

As the name suggests, this popular form of code management is based on the idea of having the

code stored in a single central location. Developers then need access to the source in order to

record any work progress, which is written directly to the central system. This foundation gives

rise to a simple workflow for version control:

1. Pull changes from other developers

2. Perform coding tasks and ensure they compile

3. Commit changes; save to central repository

There are many popular tools for CVCS tasks, with the most widely used being Subversion

(SVN) (Atlassian, 2012).

Distributed Version Control System (DVCS)

GitHub, on the other hand, follows a contrasting principle where the code base is distributed among developers. What this means is, each team member has a local copy of the code base, on which they can work and perform commits which will save changes locally. Once the developer is ready to share his or her changes, he or she then "pushes" the changes, which is simply saving these changes to the remote repository. The resultant workflow is slightly more complex than for a centralized system, given the addition of intermediate commits. Mercurial, GitHub and Bazaar are among the most popular DVCS's available (Atlassian, 2012). We will now perform a comparison of both systems.

Distributed systems have some advantages over centralized ones. First, developers are able to perform every all code-related tasks at high speed, with the exception of pulling and pushing. This is due to having a local working copy, which allows bypassing server connections required in a more centralized system. Also, by keeping a local copy of the code, developers can continue working on a project for as long as desired, only pushing and pulling changes once network connectivity is restored. Finally, team members have total control over local copies of the code, and can choose to share changes with a subset of the team to review, before broadcasting the changes.

Nonetheless, using a distributed system has its drawbacks. First, if the project contains several

large files in the repository, a huge burden would be placed on local storage to keep all the different versions in a distributed system. Furthermore, projects with extensive code-base or long histories, it could take unbearably long to clone the repository.

4.4 Database

A database is an essential component of our project as it provides efficient management of data as well as the capability of performing multiple tasks simultaneously. Without a database, data in our application would not be persistent and would have to be re-entered upon each new instance of our application. Re-entering all patient information is practically infeasible and would render our application useless. Fortunately, the benefits of having a database, particularly a relational database, solves the problem of persistence by storing data as rows in tables with columns defining the elements of the data. A relational database uses multiple tables and a schema to define the relationships between them. The queries can merge the results from multiple tables allowing for a more organized and versatile database (Aveda, 2015).

Seeing how crucial a database is for our project, we looked towards the current industry standard in order to increase our own success for this project. According to DB-Engines Ranking, the top three most popular databases are as follows: Oracle, MySQL, and Microsoft Server SQL (MS SQL). The website scores the databases by standardizing and averaging parameters such as frequency of technical discussions, searches, and profiles in professional networks in which the system is mentioned ("DB-Engines Ranking", 2016). We considered these three databases and evaluated each in the following subsections.

4.4.1 MySQL

Because MySQL started as an open source database, there is a huge community of developers and experts. The database is easy to use for any beginners and any difficulties can be resolved with the large community. MySQL is considered to be the industry standard as most startup companies use it due to its ease of use as well as the inexpensive implementation cost. However, the database is limited in terms of functionality. Basic functions such as ACID compliance and text search need additional add-ons. Since MySQL is not fully SQL compliant, other limitations include data warehousing and performance diagnostics.

4.4.2 Oracle versus MS SQL

The difference between Oracle and MS SQL functionality is minimal. Each provides equal amounts of functionality in its own style. Both include object oriented data storage, the support of cursors, and large database management ("What are some advantages and disadvantages of Oracle database?"). Oracle does have slightly better error handling as it makes changes only in memory and only commits when the COMMIT command is executed. On the other hand, MS SQL executes and commits each command individually causing more difficulty when undoing changes (Stansfield, 2014).

4.4.3 Selected Database

MySQL was not chosen for our project primarily due to its lack of scalability (Mack, 2014). It is ideal for companies with small to mid-ranged databases which according to John Feminella, can range from 100,000 to 10 million stored records ("Just what is 'A big database'?", 2009). Seeing that the population of Ghana is about 26 million people (2013), a standard hospital could

potentially surpass 10 million records due to multiple patient visits and the storage of these records over a given period of time. Due to size and scalability being our biggest criteria for the project, Oracle and MS SQL are the two best alternative candidates.

Although there were minor differences in functionality between MS SQL and Oracle, Oracle provides slightly better error handling making the database easier to work with in general. In addition, it is ranked the most popular database according to DB-Engines Ranking List ("DB-Engines Ranking", 2016), meaning that there is significant preference for Oracle in the current industry. Our previous experience with Oracle in combination with the aforementioned reasons is what made it our preferred database for the project.

4.5 Integrated Development Environment

For our Integrated Development Environment (IDE), we had three main options: Eclipse, IntelliJ IDEA, and NetBeans. All three are fully-featured IDEs that support the Java language and the Oracle database. For our criteria, we looked at each IDE's features, ease of use, and add-on support.

Eclipse is a usual choice for Java development. It is fully-featured with autocomplete suggestions, syntax checking, and it supports a vast array of add-ons. However, because of the many features and large add-on collection, Eclipse has a steep learning curve (Tee, 2012). Also, the add-ons that are available for download are not guaranteed to work with the newest version of Eclipse or other add-ons (Tee, 2012).

NetBeans is another solid IDE for Java development and its development was started by Sun

Microsystems, the same company that created Java (Viswanathan, 2017). Like Eclipse, it has

syntax checking and autocomplete. It also contains version control and code refactoring tools.

While the base version of NetBeans is easy to learn how to use, there is a steep learning curve for

its advanced features (The NetBeans IDE: Pros and Cons, 2014).

IntelliJ IDEA was another choice for an IDE. It is easy to learn how to use, it provides syntax

checking and provides autocomplete suggestions, and it has the ability to automatically refactor

code (Viswanathan, 2017). The only disadvantage to IntelliJ IDEA is that of the cost for the

Ultimate version.

We decided to use IntelliJ IDEA for our IDE. The team had previous experience using it in past

projects. Also, it is easy to use and has many add-ons available for download. The only con, the

cost for the Ultimate version, was remedied since this version is free for students.

4.6 Cloud Services

In order to make our application more ubiquitous, we decided to investigate the kind of support

we could get from cloud services. Since each of these cloud services has a multitude of products

under differing categories, we focused on the features that we require from a cloud platform.

Additionally, we limited our scope to three popular cloud services with these requisite features;

Google Cloud, Amazon Web Services, and Microsoft Azure. We used cost as our main criterion

for selection since this is an academic project, but highlight any noteworthy services.

4.6.1 Hosting

We required two forms of hosting from cloud providers; SQL and storage hosting. For SQL

hosting, it was essential that our chosen cloud solution have excellent support for Oracle

database, which we had selected as for database management. During the project, we primarily

ran the Oracle back-end from our university's installation. Nonetheless, choosing the right cloud

solution would increase sustainability of the project. On the other hand, cloud storage was also

required to hold any files that become necessary to the application, either during development or

after deployment. This is a basic resource and we simply considered price per memory as a

determinant.

4.6.2 Computing Resources

Business Intelligence

Business intelligence refers to the use of various data discovery techniques to find interesting

patterns from raw data, which can help in decision-making. Ultimately, our application sought to

provide key insight into population health trends using all our collected data. Hence, we needed a

platform with excellent intelligence capabilities, that could also handle the magnitude of data we

provided (Mulcahy, 2007).

Analytics

In order to get insight into application performance, on a holistic and modular level, it was

essential to pick a service with analytic tools. These analytics measures include tracking button

clicks, frequency of visiting specific pages, among others. The information gathered from these

would help us continuously improve upon various aspects of our application, helping shape the

project as we progressed.

Virtual Machines

In order to host any potential backend server, we need access to virtual machines that can be run in the cloud. As each of the cloud vendors provides this service, we will look at ease of use and pricing as our determinants.

4.6.3 Security

Access Control

User management is a very important issue with respect to our application. Since we will be handling sensitive patient data, it is crucial that we prevent any unauthorized persons from accessing information. Developing a user management system robust enough to handle such requirements is another MQP on its own, so we have decided to explore some cloud services such as IAM. We will therefore be looking to select a platform that provides a secure but easy-to-use user access control scheme.

Key Management

For our system, which might potentially have a backend service such as a REST API, we will need to have effective key management. This is a security measure that ensures that only authorized endpoints are able to access the server resources. Table 5 below gives a summary of some key pros and cons of the services under investigation.

Service	Pros	Cons
Azure	1M free push notifications per month - Active Directory - Hybrid cloud approach	- Free services last only 30 days ($200) - Only MS SQL Server - No Temporary VMs
AWS	- High profile and large customer base - Open and flexible; "mix and match" - Oldest vender (10 years; 2006), hence unparalleled breadth & depth of services - Multiple database offering: Oracle, MySQL, MS SQL + more	Per hour billing
Google Cloud	Sustained Use Discounts (SUD)	- Only supports MySQL - Only supports IAM for identity management - Limited application services

Table 5 - Pros and cons of cloud services: Azure, Amazon Web Services, and Google Cloud

4.6.4 Pricing

All three services offer on-demand compute services, however, Google is the only provider

without a fixed-term plan. Microsoft and Amazon's fixed-term tiers are both consistently

cheaper than their pay-as-you-go services, with increasingly attractive discounts for longer

terms. Google, on the other hand, offers Sustained Use Discounts (SUD), which automatically

applies discounts to the on-demand prices depending on how much a specific instance is used. In

this scheme, higher resource usage results in higher discounts. Additionally, AWS charges by the

hour, whereas Google and Microsoft both charge per minute for their services.

4.6.5 Selection: AWS

The single offering which delivers the services vital to our application is AWS. In addition to its

various advantages over the other cloud vendors, we have access to AWS student credit, which

will enable us to use full-fledged features of the system.

Consequently, our software development environment has the following form. We chose GitHub and Google for our code and document repositories, respectively. Trello was used for task management, with Slack as the primary communication medium. Finally, as discussed above, Amazon Web Services (AWS) was selected as our cloud platform.

# 5. Architectural Spike

5.1 Requirements

5.1.1 Survey Findings

As part of our requirements elicitation, we depended on surveys to receive information from healthcare professionals in Ghana. We chose surveys over interviews because:

1. Vast differences in time zones made meetings inherently hard to schedule, with one party needing to make sacrifices.

2. Doctors, who will be the primary users of our system, are notoriously busy, and it was nearly impossible trying to schedule meetings given issues with availability, coupled with time zone constraints.

3. Finally, surveys would allow us to potentially receive a multitude of responses which would be crucial in ensuring the versatility of our application.

Although we did not receive as many survey responses as hoped, we managed to unearth a few interesting trends from our limited data collection. Given the small sample size, none of our findings are conclusive; they only serve as a guide to our development.

Limited experience with EHR's

80% of our respondents stated they have no experience with EHR's. This is alarmingly high, and provides further motivation for the application, given the clear need. It also foreshadows some difficulty in implementation, as healthcare professionals would require adequate training given the dearth of computing skills.
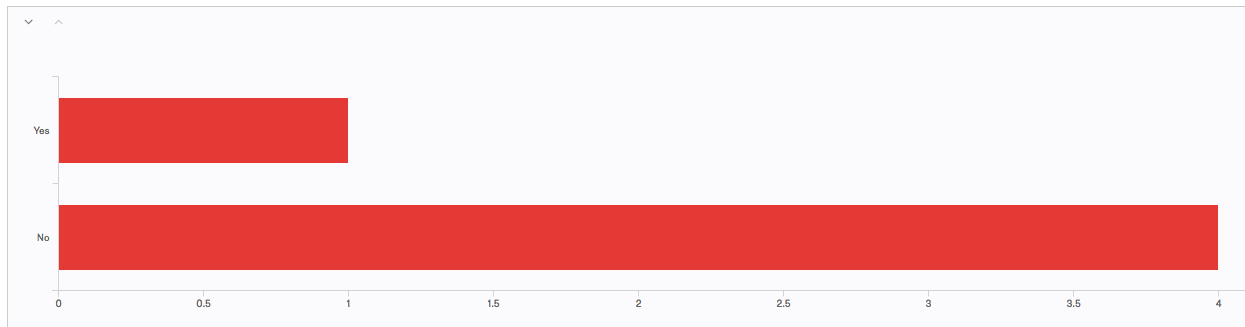
Figure 9 - Graphic showing EMR experience of respondents

Lack of disease tracking

Figure 9 helped guide our decision to include disease-tracking capabilities in our proposed

application; the single respondent who had used an EHR system reported none such

functionalities. Our application looks to employ valid disease-tracking methods to find solutions

to local health issues from public health statistics. Those without EHRs did not track diseases

either.

Paper-based records

We received two responses detailing how information was recorded, both indicating paper-based

record-keeping. Not only does this create inefficiencies in data storage and retrieval, the practice

is also detrimental to the earth, as large volumes of paper are required. Our application would

serve this need directly, especially for our proposed diseases. This sentiment is reflected with all

respondents saying such a system would be "Extremely useful". We excluded EHR respondents

from this question, since we believed this need had already been met and would not bring any

additional insight.

Pressure on drugs

The graphic below shows the response to our survey question 15:

"Please rank the following drugs by how frequently you prescribe or administer them for the

treatment of malaria:"
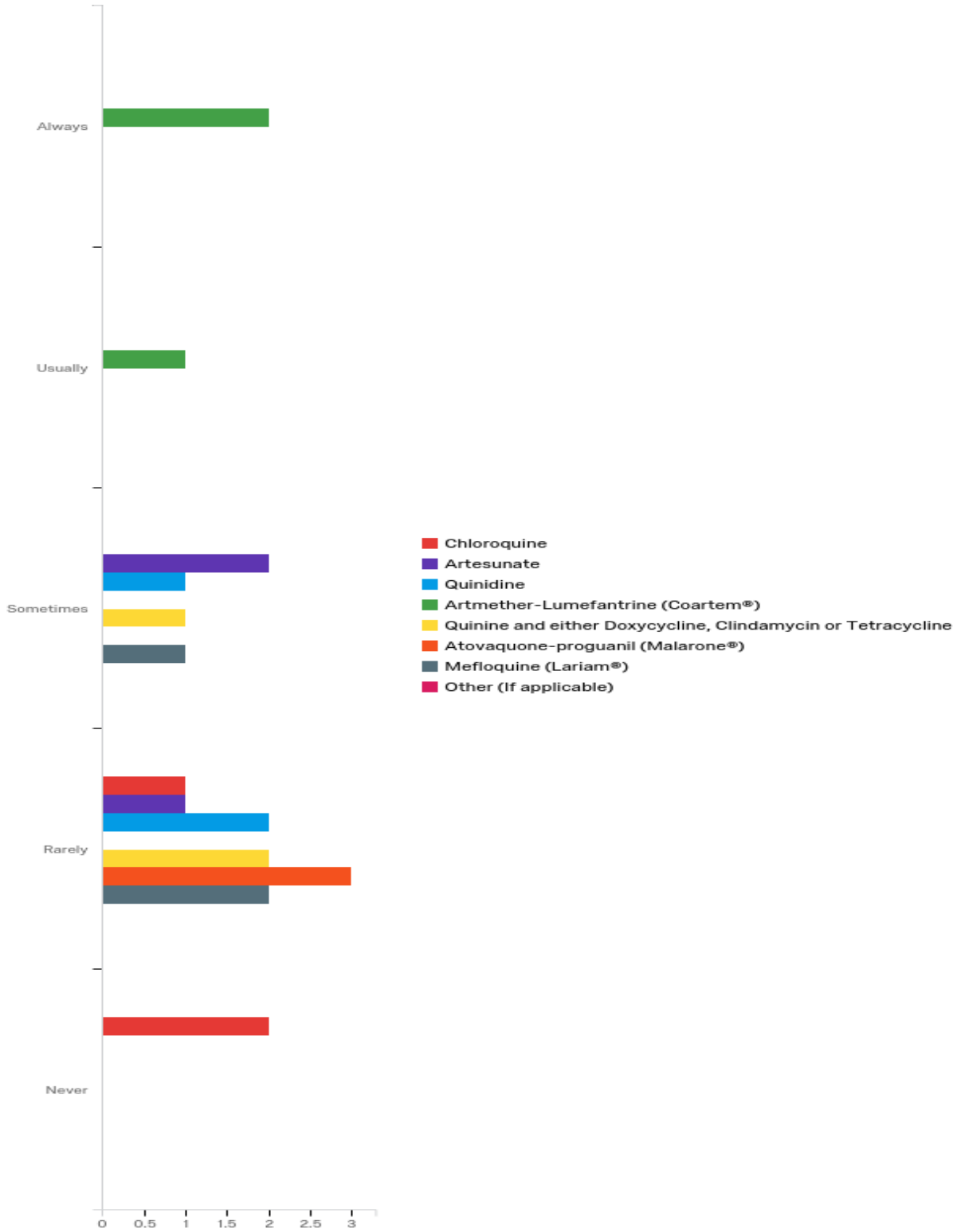
Figure 10 - Graphic showing drug usage distribution

As visualized in Figure 10, there is one dominant drug used in combatting malaria cases:

Coartem. This indicates significant pressure on this drug, as the most in-demand option. Others,

such as Artesunate and Quinidine, are used as alternatives in cases where patients are allergic to

Coartem, or during shortages of the drug. Chloroquine was deemed an ineffective malaria drug

due to an increased resistance of the parasites, and has been phased out completely. This was the

sole drug which is no longer prescribed by any of our respondents ("Never" column).

5.1.2 Functional and Nonfunctional Requirements

There are several functional and nonfunctional requirements for our application, VermaMS.

Functional requirements are "what the system should do" and nonfunctional requirements

"describe how the system works" (Eriksson, 2012).

Functional Requirements

Medical Data - VermaMS will track medical data on patient visits. These data will primarily

include: any diagnoses, treatments, and diagnostic test results. This will help the physician to

better treat the patient if he or she knows any tests were run to diagnose a disease and if any

treatments did not work or the ones that did work. Also, all changes made to system data will be

logged, i.e. the deleting of a patient from the system will be logged along with the date and time

and the username of the person who deleted it.

Authorization levels - VermaMS will be designed according to the principle of least privilege,

which is discussed below in the security requirement. The application will contain a username

and password for each user, and will be required to log into the system each time it is started. Users will be given one of two primary authorization levels: administrator and physician.

Administrative functions - Administrators will have the necessary level of access to delete old records either on their own or via a request of a physician. Administrators will also have access to modify or delete physician files from the database.

Notifications - Patients and physicians should be notified of upcoming appointments - the physician so he or she can prepare and the patient so he or she knows to show up. The doctor will receive notifications automatically while the patient can opt-in or out at any time.

Generate Reports - According to the Data Protection Act of Ghana, the hospital is required to give the patient a summary on all of the information stored in the database about that patient in permanent form. When a patient makes such a request, VermaMS will generate a report of the patient's file and print it out.

Save Data - VermaMS will be able to export data to an XML file and save it to a physician's computer.

Retrieve Data - Since the database will be integrated on the cloud, VermaMS will be able to retrieve that data and display it as needed for physicians and system administrators.

Enter Data - VermaMS will allow physicians to enter data about a patient and that data will be saved to a database so that it can be retrieved later.

Non-Functional Requirements

Security - The most important non-functional requirement is going to be application and data security. In order to make our application secure, we are going to use the principle of least privilege (Gegick & Barnum, 2013). This security principle is defined by giving users of software the minimum level of access that they need to perform their duties for the least amount of time that is necessary to complete those duties (Gegick & Barnum, 2013). For example, in our application, a doctor or nurse cannot sign into the application and delete any medical records, nor can they access or modify any records of patients that are under another physician's care. Also, VermaMS will enforce good password practices by requiring that all passwords have at least eight characters, at least one number and at least one special character. The system will also lock an account if the password is entered incorrectly ten times and then the person will need to see an administrator to unlock the account.

Reliability - VermaMS needs to consistently function the way that it was intended to function.

Data Integrity - Data Integrity is the "accuracy and consistency of data stored in a database, data warehouse, data mart or other construct" (Teeling, 2012). Since our application is built upon the premise of tracking information, we need to ensure that data integrity is maintained to the highest standard. Our database will be stored in a separate place from the local machines that connect to it. In case something happens to the local machine, the data is preserved and can be retrieved later. Also, we will build our database using best practices to ensure that it is secure so that entries cannot be accessed or modified by the wrong person.

Scalability - Since VermaMS is intended to be used by multiple users at once in a hospital, system scalability is an important non-functional requirement. One user needs to be able to use the application without interfering with another user's session.

Extensibility- Our team will implement VermaMS so that it can be built upon in future projects by adding additional capabilities, such as an automatic billing system. We are focusing our initial version of the application to target four diseases (HIV, Malaria, Meningitis, and TB) so we will code with scalability in mind.

Usability - While other EMRs are available, they are often complicated to use or do not have needed features. For example, in our survey, one of the respondents indicated that while the hospital he works for uses an EMR, it does not have the ability to track disease information like VermaMS. We also want to design the application so that is easy to use with an intuitive and simple design.

Performance - VermaMS needs to perform well and execute tasks quickly. Hospitals see many patients on a daily basis, and the application needs to be able to keep up with this demand. It also needs to be available and minimize crashes and downtime if hospitals in Ghana are going to use it instead of the current paper system.

Supportability - VermaMS needs to be designed in a way so that the system administrator can

easily navigate the application and make sure it functions correctly.

Legal - There are laws in Ghana that govern the collection and storage of medical data. VermaMS needs to meet all of these requirements to be legally compliant with the laws of the nation. Legal requirements have been discussed earlier, with data security and access being the two main facets of the legal requirements for the application.

After the list of software requirements was determined we performed functional modeling. In order to model the functionality of our application, we wrote epic stories, user stories, scenarios, and UI mockups. By creating these, we were able to easily focus on implementing the functionality required by our application.

5.1.3 Epic Stories

For our analysis, we have the following three user roles:

1. Doctor

2. Nurse

3. Administrator

Some use cases applied to a combination of users, which we grouped as follows:

1. Medic:

   a. Doctor

   b. Nurse

2. MedicPlus:

     a. Medic

     b. Administrator

1. VermaMS will allow physicians and nurses to record information for treating patients with tuberculosis.

2. VermaMS will allow physicians and nurses to record information for treating patients with meningitis.

3. VermaMS will allow physicians and nurses to record information for treating patients with malaria.

4. VermaMS will provide necessary data security to meet Ghana standards.

5.1.4 User Stories

Malaria
- A doctor wants to be able to enter malaria patient diagnostic information and save it to his or her file, including tests performed and the outcome of those tests, what species of malaria the patient has been infected with, and the treatment that is prescribed. A doctor wants this capability in order to facilitate patient treatment and easily track patient information.

Tuberculosis
- A doctor wants to be able to enter TB patient diagnostic information and save it to his or her file, including the tests the doctor has performed, whether the infection is latent or in

the disease stage (and if in the disease stage, what parts of the body the bacteria are affecting), and the treatment that is prescribed, in order to facilitate patient treatment and easily track patient information.

Meningitis

- A doctor wants to be able to enter TB patient diagnostic information and save it to his or her file, including the tests that are performed, how the patient contracted meningitis, and the treatment that is prescribe, in order to facilitate patient treatment and easily track patient information.

Common

- A doctor wants to be able to generate a XML report of a patient's diagnosis and treatment records so that it can be sent and used by other public medical facilities.

- A doctor wants to be able to give the patient the option to send him or her notifications for treatments and upcoming appointments.

- A doctor wants to be able to enter in background information for a new patient and save it to the database, so that it can be easily retrieved later.

- A doctor wants to be able to search for a patient already in the database so that his or her file can be easily saved and updated during later visits.

- An administrator wants to be able to delete a patient's records, as long as they have been kept at least for the time set forth by law, in order to remove unneeded records.

- A doctor wants to be able to see a patient's health and visit record to determine and prescribe an effective treatment.

5.1.5 Scenarios

Doctor

Dr. Jones is attending Martha Smith, who is experiencing some symptoms. He searches through VermaMS to find if Martha has previous records for previous diagnoses and vaccinations. No data came up from the search, so Dr. Jones enters a new patient entry with the Martha's basic information such as age, name, and gender. Dr. Jones then enters the patient's symptoms in the search bar to see which diseases she may have. All tested diagnoses and their results are stored in Martha's file. Her file is then updated with the appropriate treatment prescribed by Dr. Jones. Martha is asked if she would like to receive notifications from VermaMS. She decides to get notifications by text. Dr. Jones selects text and sets the date for when she will receive the notification.

Medic

Martha Smith, a returning patient, comes in for diagnosis. Dr. Jones looks at her previous records. It turns out that Martha has moved and changed her phone number since her last visit. Dr. Jones updates the Martha's file with her current address and number. Martha's records also show that she has been previously diagnosed twice with malaria. Dr. Jones gives Martha the same treatment as before. He also exports Martha's file and current diagnosis and treatment to an XML file so that it can be sent to another public medical facility if she moves again.

Administrator

Dr. Jones wants to delete some patient data that is no longer needed. He searches through the database of patient records, listing records that are over 10 years old. The records are all selected

and the delete button is pressed. Before the records are deleted, John Edwards, an administrator,

has to sign in again to confirm the delete action.

5.1.6 Use Cases

Use Case #1
1. <u>Name</u>: Enter New Patient

2. <u>Participating Actor</u>: Nurse

3. <u>Entry Conditions</u>:

    a. The patient is not already in the database

4. <u>Exit Conditions</u>:

    a. Patient record exists in database

5. <u>Flow of Events</u>:

    a. Medic requests to add a new patient to the system

    b. VermaMS presents a form to capture new Patient information

    c. Medic enters all required information and clicks button to save

    d. Medic reviews information and confirms action

    e. VermaMS persists Patient information to database

6. <u>Alternate Flow of Events</u>:

    [Cancel]

    a. Medic chooses to cancel at an intermediate step

    b. VermaMS confirms action and removes data capture form

Use Case #2

1. <u>Name</u>: Delete Patient

2. <u>Participating Actor</u>: Administrator

3. <u>Entry Conditions</u>:

   a. The Patient has a record in the database

   b. VermaMS is currently displaying Patient Profile

4. <u>Exit Conditions</u>:

   a. The patient's information no longer exists in the database

5. <u>Flow of Events</u>:

   a. Administrator clicks button to delete Patient Record

   b. VermaMS confirms action

   c. VermaMS removes the Patient record

6. <u>Alternate Flow of Events</u>:

[Cancel]

   a. Administrator rejects action on confirmation request

   b. VermaMS returns to Patient profile

   c. Patient record remains in system

Use Case #3

1. <u>Name</u>: Generate XML Report

2. <u>Participating actors</u>: MedicPlus

3. <u>Entry Condition</u>:

   a. VermaMS is currently displaying the diagnosis page

4. Exit Condition:

    a. XML document saved at desired location on workstation.

5. Flow of Events:

    a. MedicPlus clicks on button to generate XML report

    b. VermaMS presents interface to request where to save file

    c. MedicPlus selects where to save XML file and confirms

    d. VermaMS saves document in selected location

6. Alternate Flow of Events:

[Cancel]

    a. MedicPlus cancels at an intermediate step

    b. VermaMS resumes previous activity

    c. No XML document generated

Use Case #4
1. Name: Save Notification Preference

2. Participating actors: Medic

3. Entry Condition:

    a. [Option A] Medic has diagnosis page open in VermaMS

    b. [Option B] MedicPlus has Patient profile open in VermaMS

4. Exit Condition:

    a. VermaMS persists Patient's notification preferences

    b. Patient receives timely notifications, as scheduled

5. Flow of Events:

    a. Medic requests notification preferences from Patient

    b. Medic enters information into VermaMS

    c. VermaMS saves notification settings for Patient

    d. [Optional] Patient receives notification, at the scheduled time

Use Case #5

1. <u>Name</u>: Send Notification

2. <u>Participating actors</u>: Medic

3. <u>Entry Condition</u>:

    a. Medic has access to VermaMS Notifications Console

4. <u>Exit Condition</u>:

    a. VermaMS delivers notification to Patient

5. <u>Flow of Events</u>:

    a. Medic selects Patient who will receive the notification

    b. Medic inputs message details and select mode of communication

    c. Medic chooses to send message immediately or schedule for later

    d. Medic clicks button to save notification

    e. VermaMS saves a record of this transaction

    f. Patient receives notification at the specified time

6. <u>Alternate Flow of Events</u>:

[Cancel]

    a. Medic chooses to stop at an intermediate step

    b. VermaMS discards progress and returns to Notifications Console

Use Case #6

1. <u>Name</u>: Update Patient Information

2. <u>Participating actors</u>: MedicPlus

3. <u>Entry Condition</u>:

    a. VermaMS has a record for the given patient

    c. MedicPlus has required authorization level

7. <u>Exit Condition</u>:

    a. The patient record is updated.

8. <u>Flow of Events</u>:

    a. MedicPlus opens up Patient Profile

    b. MedicPlus updates information on the page

    c. MedicPlus presses save and VermaMS saves the updated file.

9. <u>Alternate Flow of Events</u>:

[Cancel]

    a. MedicPlus cancels updating the information

    b. VermaMS discards changes, restores original information

    c. Patient record remains unchanged

[Undo]

    a. MedicPlus wants to revert last change; presses undo button

    b. VermaMS restores the value of the last changed field

Use Case #7

1. <u>Name</u>: Lock Account

2. <u>Participating actors</u>: VermaMS

3. <u>Entry Condition</u>:

    a. User at Login Screen with 3 previous failed login attempts

4. <u>Exit Conditions</u>:

    a. VermaMS locks the account and notifies the Administrator

    b. VermaMS displays message to User

5. <u>Flow of Events</u>:

    a. User enters wrong login credentials 3 times

    b. VermaMS locks the account, rejecting further login attempts.

    c. VermaMS notifies an Administrator of the account lock and logs it.

Use Case #8
1. <u>Name</u>: Open new Case

2. <u>Participating Actors</u>: Medic

3. <u>Entry Conditions</u>:

    a. Medic is logged into VermaMS

    b. Medic has required authorization level

    c. Medic has access to new Patient Case information

4. <u>Exit Condition</u>:

    a. VermaMS stores record of case and sets it to open

    b. Medic(s) assigned to Case

5. <u>Flow of Events</u>:

    a. Medic clicks on button to open new Case

  b. VermaMS presents dialog to fill in Case information

  c. Medic saves information and selects assigned Medic(s)

  d. VermaMS notifies assigned Medic(s) about new Case

  e. VermaMS stores Case record and sets it to open

6. <u>Alternate Flow of Events</u>:

[Cancel]

  a. Medic cancels at an intermediate step

  b. VermaMS confirms action

  c. VermaMS discards changes and returns to previous screen

[Undo]

  d. Medic wants to revert last change; presses undo button

  e. VermaMS restores the value of the last changed field

Use Case #9

1. <u>Name</u>: Perform Diagnosis

2. <u>Participating Actors</u>: Doctor

3. <u>Entry Conditions</u>:

  a. Doctor is logged into VermaMS

  b. Doctor has Diagnosis page open

4. <u>Exit Condition</u>:

  a. VermaMS saves Patient diagnosis

5. <u>Flow of Events</u>:

  a. Medic clicks on button to open new Case

     b. VermaMS presents dialog to fill in Case information

     c. Medic saves information and selects assigned Medic(s)

     d. VermaMS notifies assigned Medic(s) about new Case

     e. VermaMS stores Case record and sets it to open
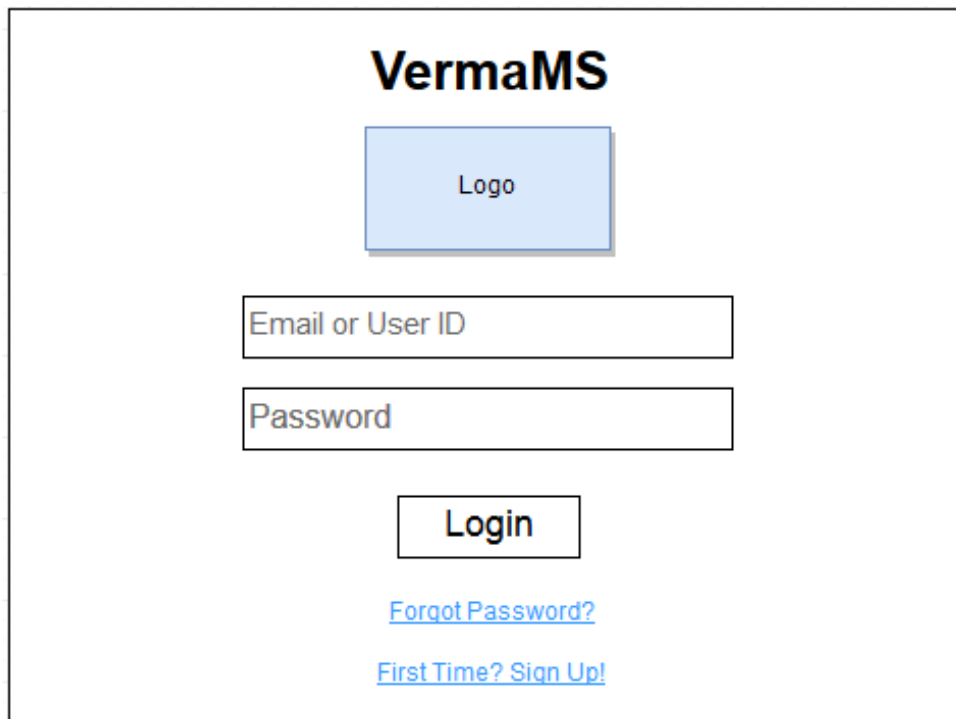
6. <u>Alternate Flow of Events</u>:

[Undo]

     a. Doctor wants to revert last change; presses undo button

     b. VermaMS restores the value of the last changed field

5.1.7 Interface Mockups

Sign In

Figure 11 - Sign In Screen

This is the first screen that users will see when executing the VermaMS application. The users will need to input a user ID or email and a password in order to log in. If the password is forgotten, then the user clicks the "Forgot password?" link to receive a randomly generated password via email. New users will be able to create an account by clicking the "First Time? Sign Up!" link. From there, the user fills in the remaining information to create an account with basic permissions.

Menu Screen
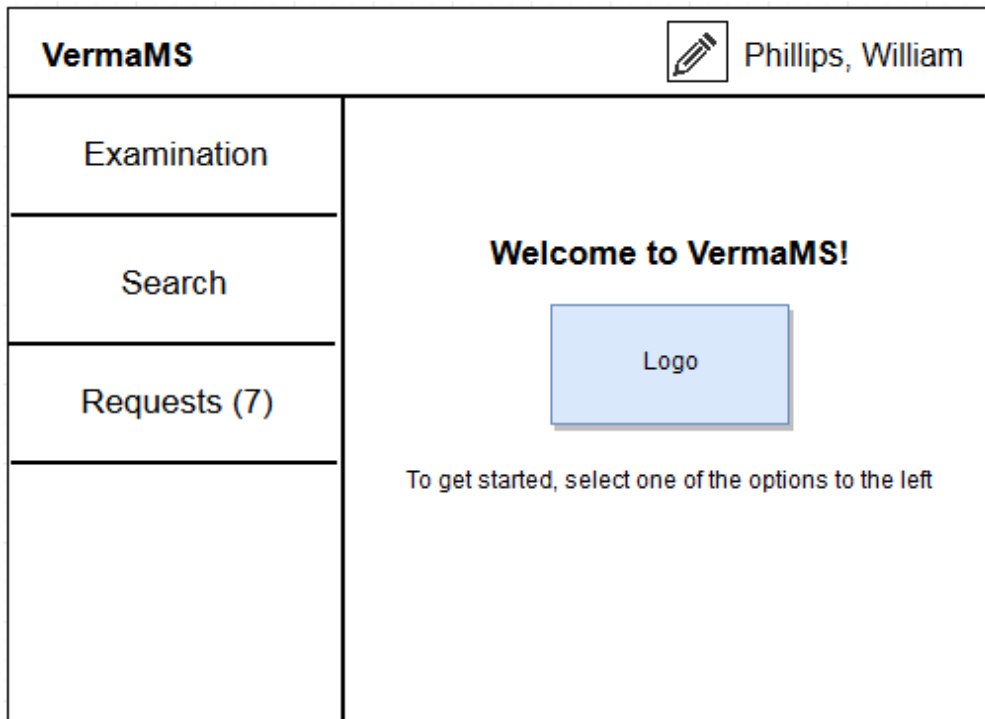
Figure 12 - Menu Screen

This is the menu screen that appears after the user has logged in. On the left is the navigation menu where a user with administrator privileges can choose to do an examination of a patient, search the database, or accept/deny requests. Users with medical staff privileges can only

examine and search while basic users can only search the database and not view any records. The user can also choose to edit their profile at any given time by clicking the edit button.

Search Screen

Figure 13 - Search Screen 1

On the search screen, all users will be able to search for patients as well as medical staff in the VermaMS application. Non-basic users will be able to click on the patients in the search table to pull up their visit records. Medical staff and Administrators can add new patients by clicking the add button. Non-Administrator users that click the delete patient button will have their request subject to approval by the administrator.

Search Screen 2

Figure 14 - Search Screen 2

Once a patient is clicked on, the left screen displays the patient's profile. This screen will have a scrollbar so that users may see more information. Non-basic users can edit the patient's information and once saved, the profile is updated along with the editor's name and date at the top. The right screen displays a table of the patient's visits. Non-basic users can click on any of these records to view them.

Search Screen 3

Figure 15 - Search Screen 3

After the user selects a patient record, the right screen displays the information of that visit. This screen will also have a scrollbar for users to see more information. Users may also edit any information on this record. Users can also generate an XML file from the patient's record and save it on their computers to either print out or send to another medical facility.

Examination Screen

Figure 16 - Examination screen

Once the user selects the examination option, VermaMS guides the user through the examination process. On the left is a list of steps to completing an examination. A user may skip to any step by simply clicking on steps listed. Any information selected or filled out will auto fill for the final report. The figure above displays what the disease selection step will generally look like.

Requests Screen

Figure 17 - Requests screen

This screen displays what requests the administrator has to approve. On the left will show the total as well as what kind of requests they are. On the right screen, the administrator will be presented with a scrollable table that shows all the requests. He or she can select a request to view its details in the screen below it. In this case, the request details screen shows the details of the patient needed to be deleted and the reason for deletion. The administrator can choose to accept or deny this request.

5.2 Analysis

This section presents a detailed look at the model and controller components of the application, and the classes that fall under each classification. For each category, some classes have been

grouped based on an encompassing feature. A few files served multiple purposes and were either placed in the best-fit subcategory or left in their parent packages.

5.2.1 Models

Profile

User
This is the superclass for anyone who intends to use the system. From administrators to all levels of health professionals. The User class is abstract to ensure that it can't be instantiated.

Administrator
This is one of the concrete implementations of a User. The administrator is the most powerful person in the system, having all privileges with the exception of performing a patient exam.

Staff
This is the only other implementation of the User class. We decided against having a separate class for each type of Staff, and rather included a UserType enum to differentiate between Doctors and Nurses.

Address
This class holds all the information to identify locations. It is composed of a streetAddress which is a string, with corresponding *District, Region* and *Country* objects.

Facility
A facility is any health institution which runs the application. This class is necessary to separate cases by origin, especially when performing data analysis from multiple sources.

Patient
This object holds information on each Patient, such as name and contact information, which are stored as string literals. More complex attributes such as Address and HealthProfile are stored as

object references.

HealthProfile

Each Patient has an associated HealthProfile, which contains all records of previous visits and

examinations, as well as other health-related information which is briefly described below.

Record

For each visit a Patient makes to a health facility, a record is created. This object simply stores

the date of said visit, along with a **Report** object.

Report

The Report object contains all information pertaining to a Patient's visit to the doctor, including

all examination findings, test results and treatments. It is made up of several components some of

which will be explained below. In addition to these object references, there is a map of Disease

to boolean, which indicates suspected diseases and its confirmation status.

Symptom

During a Patient exam, the doctor asks several interview questions to get the Patient's status in

his/her own words. Each noted symptom is then recorded in the Symptom object, which has a

couple of string attributes; one representing the symptom name/title, and the other for any

additional comments.

Vitals

This object captures information from a typical physical exam. There are primitive attributes for

height, weight, blood pressure, etc., as well as a string attribute for any notes. It also references a

list of **VitalArea** objects for the examination of specific parts of the body. The VitalArea class

has string attributes for the name of the desired area and notes. The final attribute is an enum

BasicHealthStatus, which indicates the doctor's rating of the area, with values: **Poor, Fair,**

Good, Very Good and **Excellent.** This enum is also present in the enclosing *Vitals* class to represent the overall physical examination status.

AbsLabTest

This abstract class is the foundation for our family of lab tests. It is composed of a couple of enumerations, *TestName* and *LabTestResult*, which represent the test name and the result, a disease object for its targeted disease, a string for notes, as well as a list of generic objects called *TestParameter*. All tests closely follow this template so do not require deeper clarification, however the *TestParameter* class is touched upon below. The available concrete tests implementations are: *BloodCultureTest, ScanTest (CTScanTest, XRayTest), GoldInTubeTest, LumbarPunctureTest, MicroscopyTest, RapidDiagnosticTest, TSpotBloodTest, TuberculinTest,* and a *GenericTest*.

TestParameter

This object represents whichever parameters are required to complete a test. Despite the diversity of tests, the key variable is the object type of each parameter. As such this class takes in a generic type parameter which makes it easy to produce test objects for whichever lab test. In addition to this generic attribute, the object also stores an enum, *LabTestUnit*, which indicates the unit for the parameter.

LabTestFactory

This Factory class facilitates test creation on the fly by providing an easy to use interface which returns the appropriate type of test, wrapped in the abstract superclass.

Vaccination

A class to hold patient vaccination information. For attributes, it has the name of the vaccination, as well as the date it was received and when it will expire, if applicable.

Drug

This class is used in several different contexts, however, it is simply a class to represent a specific type of medicine. In the HealthProfile, it is used to indicate patient allergies. Each Drug object contains a reference to the specific disease for which it is intended.

Disease

A simple class to represent the 3 (potentially more) diseases in the system. Each object just holds an ID and a name string.

Medication

We found it necessary for an object to represent a typical drug treatment (prescription), which is the *Medication* class. It contains references to the Drug, as well as dosage instructions. If applicable, the start and end dates for the treatment can be set.

Request

The request package contains classes pertinent to the request system for administrators and health professionals.

AbsRequest

This is the abstract superclass for all Request objects. It is composed of a primitive ID attribute,

along with several object references which provide information regarding a particular request. There two Staff objects for the person making the request and the one resolving the request. The latter will be a single value for a facility with only one administrator. A **RequestStatus** enum indicates the current status of the request with possible values Open, Closed and Pending. A second enum attribute, **RequestType**, represents the type of the request, with possible values Delete, Add, Register and Promote. Finally, there are two strings, messages and notes, as well as two LocalDate objects; one for when the request was received and the other for its resolution date. Concrete Request implementations are DeleteRequest and RegisterRequest.

RequestFactory

This Factory class makes it easy to spawn new Request objects during runtime. It contains static methods which produce the appropriate Request type based on input parameters.

Exceptions

In order to have comprehensible error handling capabilities, it was necessary to create a few extensions of the standard Java Exceptions. Each of these follows the same template, a self-descriptive name with a sole single-argument constructor.

DataNotFoundException

Thrown when a user attempts to retrieve a non-existent item from the database.

DuplicateInsertionException

Thrown when an attempt is made to store an item with a duplicate unique key.

FailedInsertException

Thrown if the Database is unable to successfully complete an insertion transaction.

ObjectNotInitializedException

Thrown if a user attempts to use an object whose constructor is not executed completely. This

prevents any security leaks from partial object instantiation.

UnspecifiedUserException

Thrown if the system attempts to access the currently logged in user, without setting it first.

5.2.2 Controllers

MainApp

This class is the starting point of our application. It contains numerous static variables to provide

convenient access to information from other classes. Furthermore, in order to prevent redundant

data retrieval, several of these variables hold key pieces of data such as countries, diseases and

symptoms, in memory.

AbsController

This abstract superclass is the base for all controllers in the application. It abstracts out a few

methods which are common to all controllers, mostly pertaining to the user interface.

Examination

This component was essential to fulfill our main requirements. Comprising of several different

sections with a few similar functions, we found it imperative to use the Facade pattern for the

package design. Furthermore, since each screen is described in detail in a later part of the paper,

the controllers will not be extensively discussed in this section. We implemented the Facade

interface with **next**, **skip** and **back** methods. These enabled easy traversal of the individual

sections, from start to finish.

Lab Test Controllers

For each of the previously mentioned lab test classes, we provide a controller with which the

information can be collected and manipulated. One of the examination sections specifically deals

with the ordering and storing of lab tests along with their results, calling these controllers into

action as required.

5.2.3 Preliminary Design

Use Case Diagram
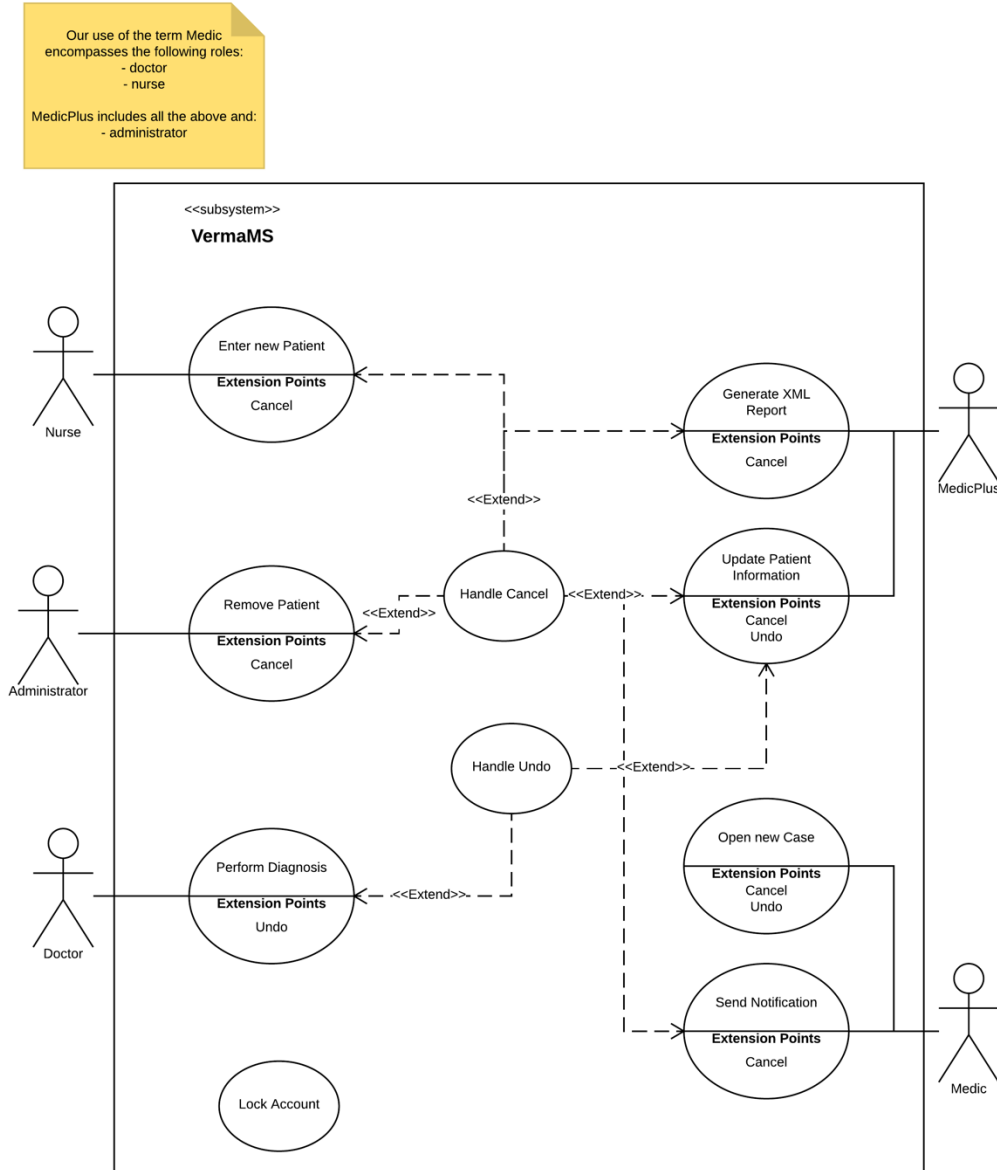
Figure 18 - Use Case Diagram

Context Diagram
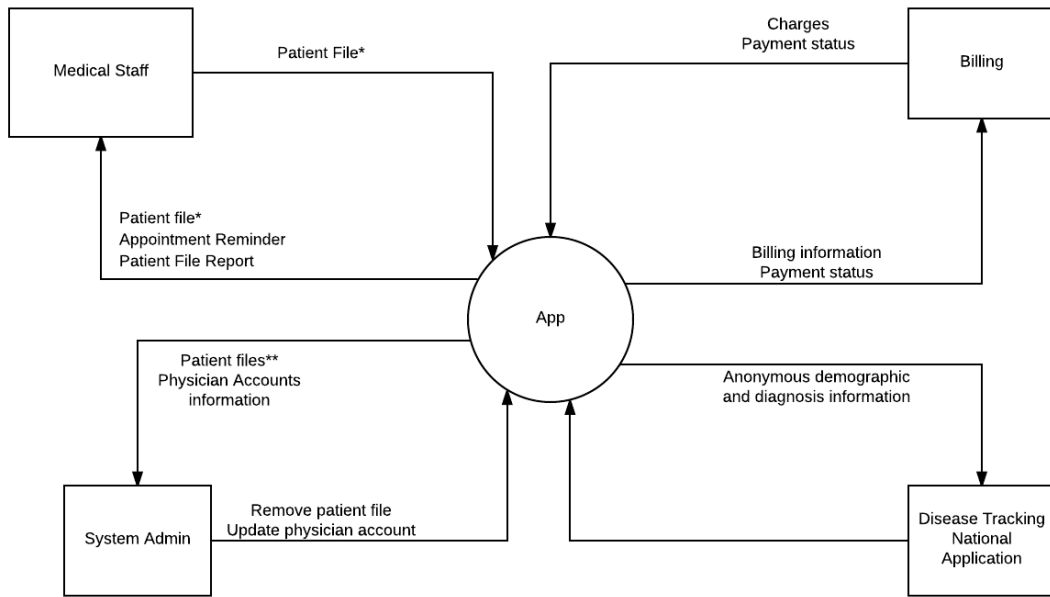
Figure 19 - Context Diagram

Class Diagrams

Figure 20 - Class Diagram

6. Design

6.1 Kanban Design

Following the principles of the Agile Kanban approach, our initial design consisted of high level diagrams, with the minimum details required to progress. As our requirements were refined by new information or a better understanding of existing documents, we refactored our design to reflect our level of insight. In the previous section, we provided our initial class diagrams. In this section, we present our interface design, architectural design, the final class diagrams and the security design.

6.2 Interface Design

6.2.1 Existing Designs

A complete evaluation of the interface design and architectural structure of existing EMR systems proved to be a challenge. Many well-known healthcare companies with EMR systems that are used broadly such as *Epic* or *eClinicalWorks* do not share software unless purchased. However, sample screenshots are shown thus providing us with a general idea of interface layout. We selected images of EMR systems that were the most popular in the U.S. The images selected are shown below.
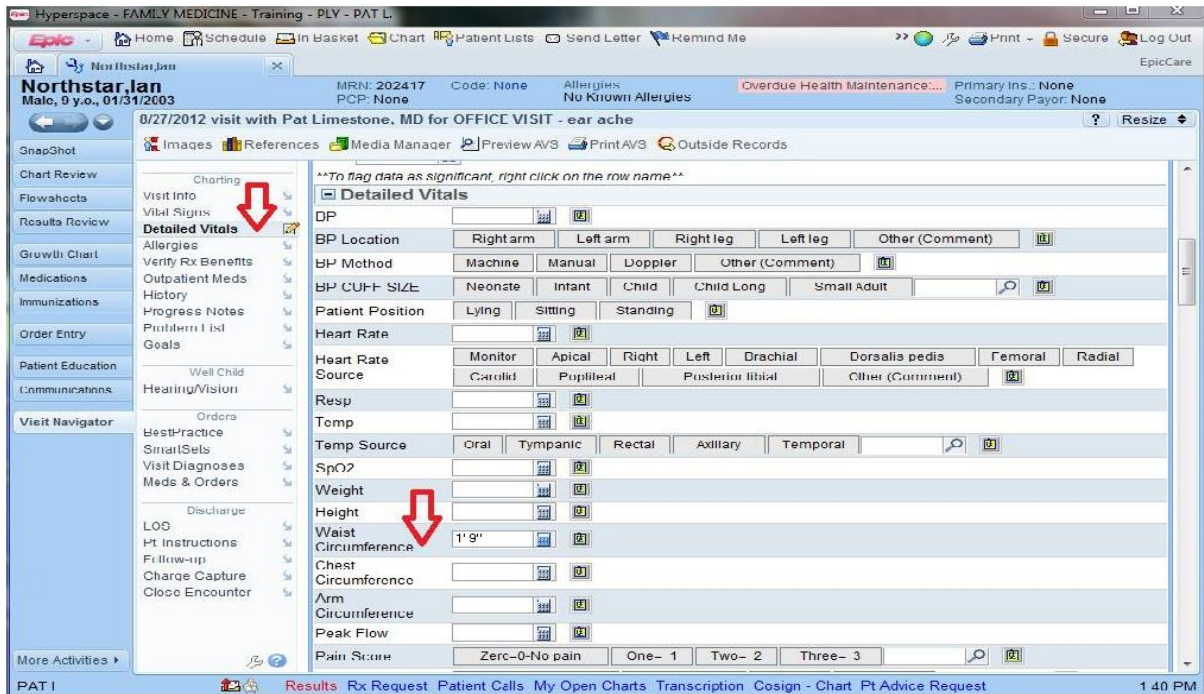
Figure 21 - EPIC EMR displaying where a user would fill the detailed vitals for a patient. Source: (A Medium Corporation, "Flatiron Health EMR Product Case Study – garyyauchan – Medium", 2016)
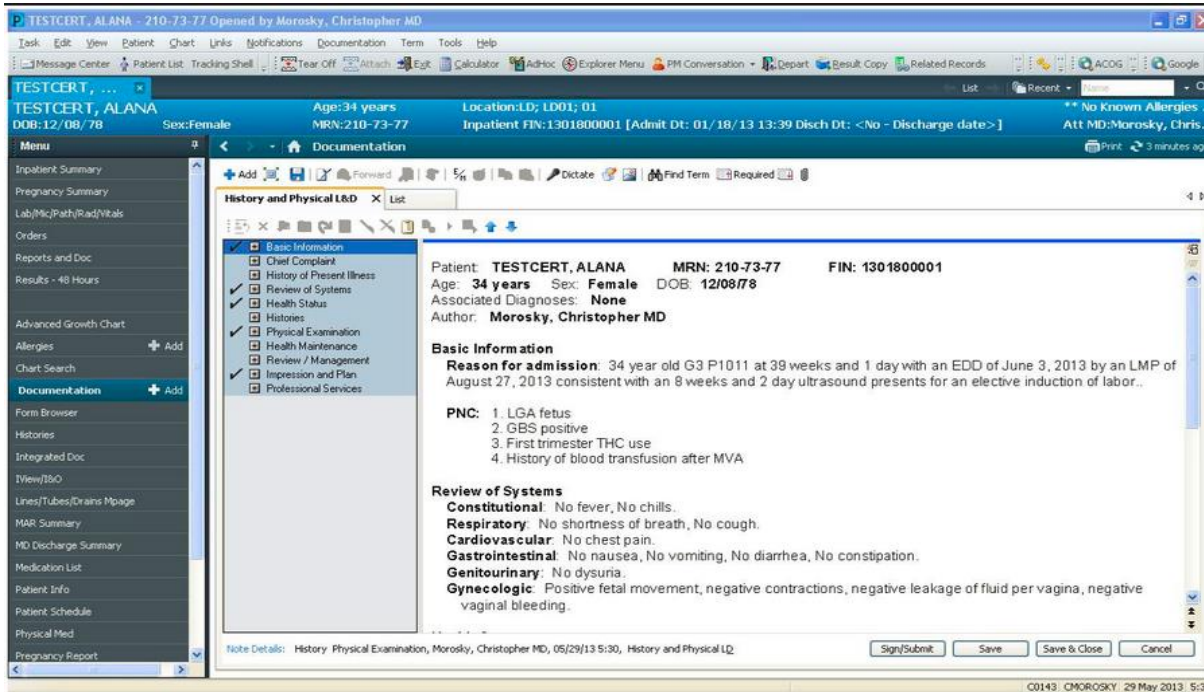
Figure 22 - Cerner EMR displaying the documentation of a diagnosis on a patient. Source: (Cerner, "Cerner | Maternity - EMR-Matrix.org")
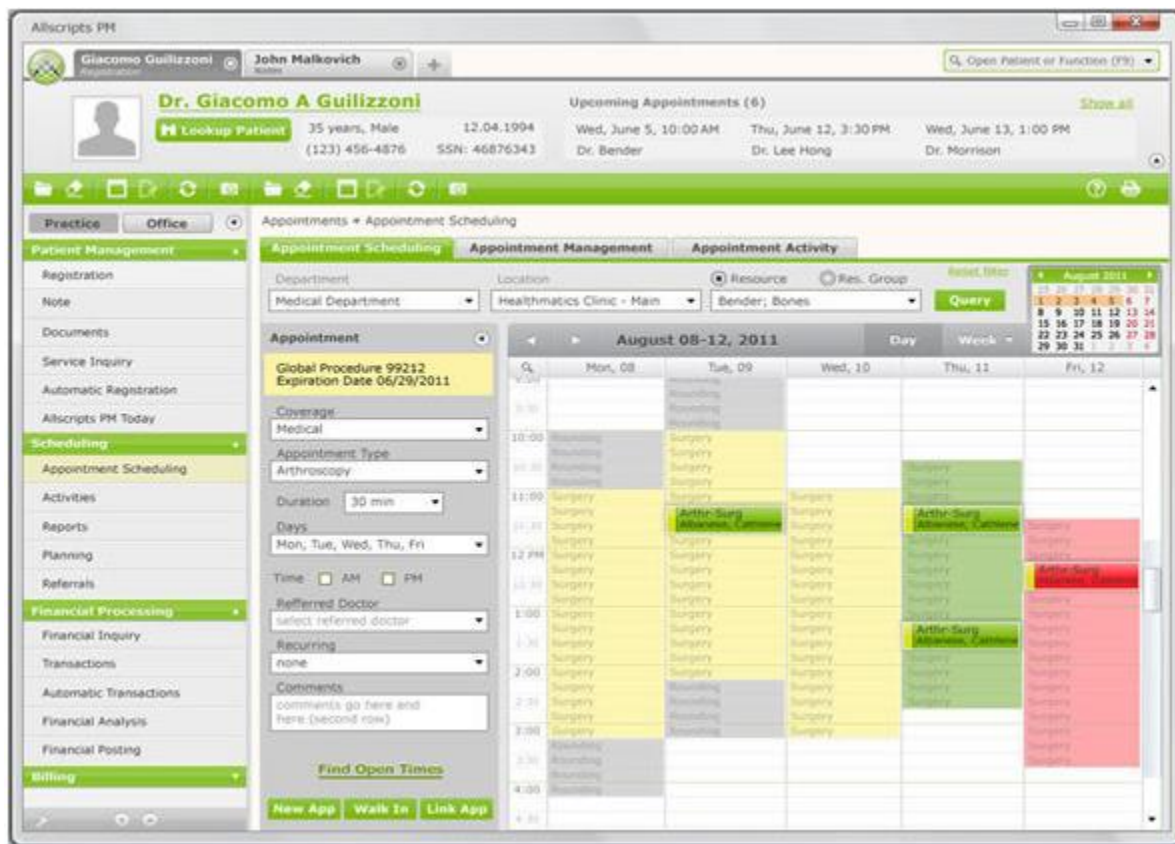
Figure 23 - Allscripts EMR displaying the appointment scheduling system. Source: (Allscripts, "Allscripts EMR, Medical transcript, practices, specialties and medical centers")

Although each EMR system has its own features and functionality, there are some noticeable similarities between the user interfaces. One of the similarities is each has a menu on the left hand side of the screen for navigation. EMRs are systems that contain large amounts of data on patients and being able to navigate to the right location quickly to perform a task is key for efficiency, especially within a medical facility. A navigation menu allows information throughout the software to be compressed and organized in a list-like fashion. This gives users a complete understanding of the capabilities of the software.

Another similarity across the images is a tab-based design. At the top of each screen, we noticed

that each tab was for a different patient. The addition of tabs adds the ability for the user to multitask. A user can focus on working on another task without having to stop the current one. For example, a physician may be in the middle of an examination with one patient but another visiting patient may need immediate diagnosis due to the severity of the symptoms. A tab system allows the physician to start a new examination on another patient without having to completely restart the current one.

The final similarity to notice is the amount of choice selection versus free form. Figure 22 represents this well in the column to the right of the navigation menu and in the menu above. There are many drop down menus that let users select pre-listed values and only a calendar to enter notes for an event. Figure 21 also shows the free form section including buttons within the text fields to indicate convenience with completion. From this observation, we can indicate that physicians do not prefer to fill in information frequently and would rather select options as it is much quicker to select an option from a list than having to take the time to fill each field individually. In the medical environment, time is valuable as medical staff have appointments scheduled with patients. Time spent filling many free form sections in an EMR application can cause delays and errors which results in a slower workflow and a decrease in patient healthcare.

6.2.2 Design Concept

In order to create a product that clients would want to use, a user and task analysis needed to be performed. A user analysis provides findings about a user's characteristics and preferences which are used to make decisions about development and design. Roles and characteristics are identified so that features are based on people who will use the product and not on the opinions

of the stakeholders (Cao, "User Analysis Before Diving Into Design (Part 1) - UXPin").

A task analysis is done to determine the design structure of software. It identifies the user goals and processes taken to achieve them in order to better understand the user. There are two types of task analyses, the hierarchical analysis and the cognitive task analysis. The hierarchical analysis was chosen for this project as it focuses on breaking high level tasks into subtasks (Affairs, 2013). By performing both user and task analysis, we follow the user-centered design process and effectively create interfaces that are explicitly made for the user.

User Analysis

Education Level

The targeted users of our application are physicians and other medical staff members. Tasks such as diagnoses of patients need to use terminology that is familiar with the users. The use of abbreviations for medical terms is a common practice within the medical field. For example, HEENT (Head, Eyes, Ears, Nose, Throat) is a frequently used abbreviation when a physical examination is conducted. Other examples of abbreviations include units for measuring vitals such as rpm. While rpm to non-medical professionals typically represents rotations per minute, a medical professional would understand it as respirations per minute.

Computer Experience

Many medical facilities in Ghana still rely on paper record systems due to EHR systems being recently adopted in 2014 ("Towards eHealth 2.0 in Ghana: A programme and opportunities for private and public ICT initiatives," 2014). With medical professionals accustomed to paper use,

introducing technology that manages the same processes can cause frustration if the learning curve is high. Novice users of technology need more guidance when completing tasks in order to increase usability. Generally in software, guidance is in the form of textual instructions for different objects or pre-filled data values in form sections. By giving guidance to novice users, learning becomes quicker and tasks are completed at more optimal speeds.

Ghanaian Culture

In Ghana, there are nine government sponsored languages. However, the official language of the country is English. This makes designing software for Ghana simpler since English can be used as a base language, seeing that everyone should understand it. Another cultural factor to consider is the units of measurement. Although our team is accustomed to the imperial system, the metric system is used in Ghana. As a result, height, weight, and temperature need to be measured centimeters, kilograms, and degrees Celsius, respectively.

Task Analysis

Patient Examination

The first task of examining a patient involves documenting the patient's symptoms. Patients will give physicians long descriptions of any changes in their physical condition. The physician then has to match this description to a single or group of symptoms. Additional comments are used to describe these symptoms in detail. Once the patient has finished reporting his or her symptoms, the physician continues the process by examining the patient.

The physician will start by inspecting the vitals of the patient. The vitals consist of blood

100

pressure, temperature, pulse, and respiration. Any abnormal readings for these values are noted as well. The physician will continue with the examination by inspecting the physical parts of the body. Common areas are examined first with more specific areas examined second. Specified areas will vary based on the patient's symptoms. Any findings will be documented for each specified area. The physician then continues to the next task of diagnosing a patient.

Patient Diagnosis

Based upon the findings of the examination, the physician has a list of suspected illnesses that the patient may have. In order to confirm these suspicions, various tests are ordered and carried out to determine the results. Test results will vary as they are not always positive or negative. Situations arise where there may not be enough evidence to support either side which produces in an inconclusive result. Additional notes are written for any tests that involve an analysis on data collected. Once all tests have been performed, the physician has a list of confirmed illnesses and can begin assigning a treatment.

Patient Treatment

Before assigning treatments, the physician checks if the patient has any drug allergies or is taking any current medication. The patient informs the physician if any of the information needs to be updated. The appropriate medication is assigned along with their respective dosages.

Examination Review

After treatment has been prescribed, the physician reviews all the information documented throughout the examination with the patient. Any errors in the data are fixed immediately. Once the examination has been reviewed, the prescribed treatment is added to the patient's current

medication. A report, summarizing the results of the examination, is given to patient for his or her own benefit.

6.2.3 Final Application

The following section will discuss all the components of our final project along with the design rationale behind each screen. The screens are discussed in order of appearance when using the application. Section 6.2.3 begins with the creation of our logo, then continues to discuss the login portion of the application including user registration and password recovery. From there, the tasks of adding and examining a patient are explained along with their respective subsections. The discussion ends with other functionality such as administrative requests, data analytics, and physician-patient interaction.

Logo

Figure 24 - The logo displays the name of the application and a slogan.

When designing our logo, we wanted to be able to symbolize our goal of improving healthcare within Ghana. One of the major outcomes of improved health care is a lower mortality rate. We used a heart as a symbol for life and the heartbeat line to indicate that the heart's still beating as seen in Figure 24. Our application name and slogan are between the two halves of the heart to

illustrate that our application is what keeps the heart persistently beating. Our application's slogan is an indicator of the beating heart as it expresses the purpose of the application. The application's name was used as a tribute for one of our colleague's friends who had recently died due to malaria. In order to identify the type of application, we added the "MS" portion to the end of the name which is short for "Medical System".

Login
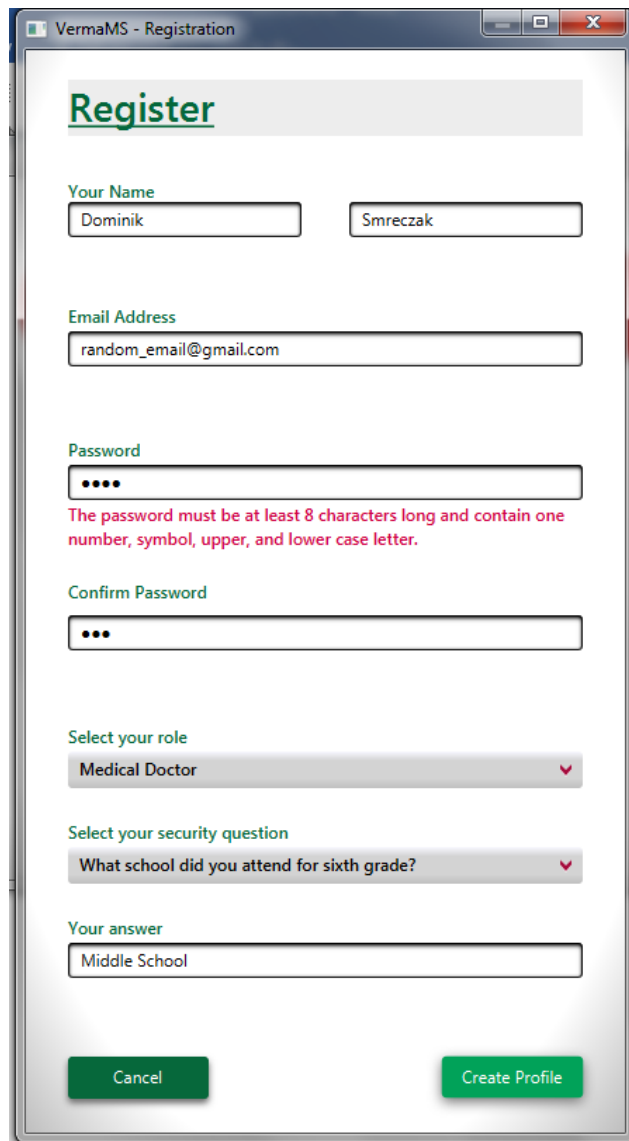
Figure 25 - Demonstrates a user logging into our application.

Figure 25 is the first screen that the user is shown when loading the application. The screen includes a standard login that most other applications use such as an email, password, and login button. The use of email was chosen over a username since typically usernames are used in social networks to provide some anonymity. In a medical environment, there is no need for this

as every staff member would have to be identifiable in order to ensure patients are seen by the

correct personnel. Additionally, email is the primary form of contact for almost any business

profession. As standard practice, we have also included a "forgot password" and registration

process to accommodate new users.

User Registration

Figure 26 - Demonstration of user registration with password error handling.

In order for a user to successfully register with our application, all of the fields in Figure 26 need to be filled. In addition to providing an email and password, the name, user role, and security questions are needed to create an account with our application. The name is required in order to identify the user that is currently logged into the application. A user's role identifies what level of access that user has within the application. All the user roles need to be approved by an administrator in order for that user to be able to successfully login. The user is given five different security questions to choose from that require an answer. This ensures that the user has a reliable option to recover their password if it is forgotten.

Since all the fields are required, there is no need for a required field indication such as an "*". Feedback for errors is displayed as red text underneath the respective fields. Furthermore, the errors are displayed one at a time each time the user clicks the "Create Profile" button. Although this creates a repetitive task of trial and error, the user is not overwhelmed with all the error messages displaying at once. The Registration Screen is also modal, meaning that other windows cannot be interacted with until the current one closes. A modal view forces the user to either complete the task or cancel it which reduces the number of potential errors for our application.

Password Recovery

Figure 27 - Demonstrates selecting the security question recovery option and answering it incorrectly.

The user has two options when choosing to recover his or her forgotten password. The user can

either choose to have a temporary code sent to the email provided in the text field, or answer his

or her security question correctly. The best way to represent a single choice between multiple

options is the use of radio buttons as shown in Figure 27. Before the user can select a radio

button, a valid email needs to be entered into the text field. By selecting either radio button, the

corresponding elements are disabled to prevent user interaction.

The security question option has two security measures to ensure user protection. The first is

handling an email that is not registered within the database. In order to prevent the user from

knowing whether the given email is already registered with the database, no error is thrown and

the first question is always displayed. The second security measure is disabling the option

completely after a certain amount of attempts. As seen in Figure 27 the maximum number of

attempts for answering the question is five. After reaching the limit, the option is then disabled

for 24 hours before it can be selected again. By disabling the option for a certain time limit,

attackers are slowed significantly.

Dashboard
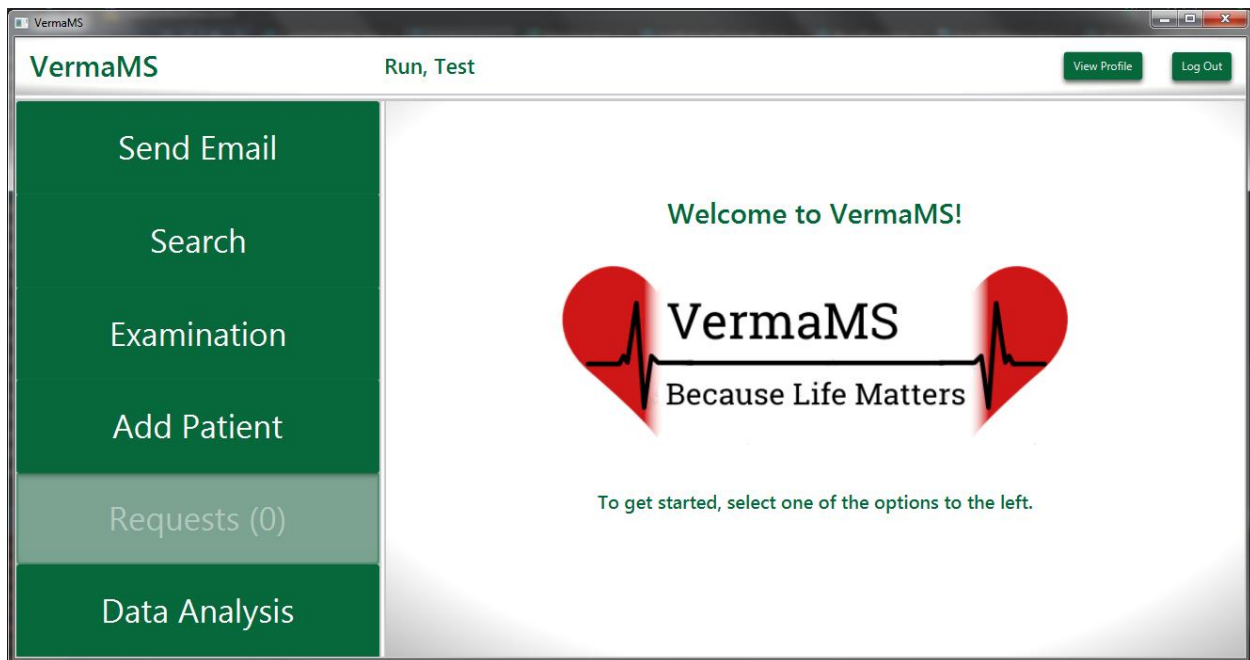
Figure 28 - The dashboard displays all the capable functionality for our application in a button
menu to the left of the screen.

Figure 28 is displayed first when the user logs in. All of the functionality of the application is

shown in a menu like fashion on the left. The buttons with more frequent tasks are oriented

towards the center of the menu with less frequent functionality towards the top and bottom. This

decision was made to minimize the distance that a cursor has to move. Users tend to have their cursors naturally gravitated towards the center of the screen and thus the distance to the frequent tasks is minimized when the cursor and the button are directly horizontal.

Some of these buttons will be disabled based on the user role to ensure functionality is only granted to users authorized to use that functionality. In the case of Figure 26 the current user role is a medical doctor, and therefore the request functionality has been disabled since physicians should not be able to delete patient data without the approval from a user of higher access level; in this case, a hospital administrator would need to approve the deletion of patient records.

The task bar at the top of Figure 28 is designed to contain functionality pertaining to the user such as modifying the user's profile. The user's name is also shown in order to give other users of the device information on who is currently logged in. A logout button is placed to the far right corner of the screen as it is the last task that any user would take when using our application. The logout button brings the user back to the Login Screen (Figure 25) preventing any other users of the same device from being able to access the functionality of our application. Furthermore, the top bar is visible throughout the application meaning that any tasks taken in our application will always have functionality for the user available.

Add Patient

Figure 29 - Demonstrates registering a new patient within the application

A user is brought to the screen shown in Figure 29 when a patient could not be found in either

the examination or search screen. The user is required to fill the fields marked with an "*" in

order to successfully register a patient. The use of asterisks is appropriate in this situation as

there are fields that can be left blank such as the cell phone and email fields. The reasoning for

this decision was that a patient must be able to be contacted in some manner. The best and most

common form of contact is a home phone which is why it was chosen as the required field for

contact. Once a patient is successfully registered, users are able to perform examinations on the

newly added patient.

Examination

Patient Selection

Figure 30 - Demonstrates selecting a patient for an examination.

When a user starts an examination, they are presented with a patient selection screen as shown in

Figure 30. Since an examination is done on patients, a patient needs to be selected from the

existing database or added if not registered. A search bar was used to accomplish the task of

selecting a patient from the database and a table was used to display the results. The search

provides the user flexibility with multiple criteria to search for such as patient ID, first name, and

last name. By providing any of the three pieces of information, the user is able to find a

registered patient. A table was used to display the results of the search query primarily for its

proficiency at sorting data. Each column can sort the information alphabetically in ascending or

descending order making it quicker to find the desired information.

The right hand section includes more detailed information on a patient. Additional information is provided for the user in case there are two patients with the same first and last name. Patient ID is the unique identifier for these situations however, typically patients are unaware of this number. By providing detailed information on a patient, a user can confirm that the current patient is the one selected in the application by using the other biographic information.

Symptom Selection

Figure 31 - Displays a user filling out the symptoms of a patient. The suspected diseases are selected as well with additional remarks about the choices made.

To match the physician-patient interview process, the first stage is to record the symptoms that the patient is feeling. The table in Figure 31 contains a list of all the common symptoms for Malaria, Meningitis, and Tuberculosis. It also contains a list of the most common symptoms experienced by patients in general. Users are able to select multiple symptoms by using the checkboxes and make comments by clicking on the rows to display a text field. There are two fundamental benefits to listing symptoms as opposed to free form filling. The first being that it reduces the amount of input needed from the user which results to faster documentation. The second is being able to perform data analysis on the recorded data. In the case of free form, each individual user may have a slightly different name for a symptom which causes data to be separated when technically they should be together. By creating a finite set, the user is required to map a patient's symptoms to the given list of symptoms.

After all of the patient's symptoms have been documented, the user then determines what diseases the patient may have. If the symptoms exhibited by the patient do not relate to the three targeted diseases, the user may choose to add a new disease. Additional notes may be added to explain the reasoning for the choices.

Physical Examination

Figure 32 - Demonstrates the information needed to collect for a physical examination.

Although symptoms provide a basic idea of what illnesses a patient may have, a physical examination is conducted by a physician in order to reveal any additional symptoms unaware to the patient. A physical examination is divided into three sections: Growth, Vitals, and Physical. Growth and Vitals involve numerical measurements while physical is more descriptive as it involves inspecting specific areas of a patient. The difference in input type is what determined the layout of the screen in Figure 32.

The left column consists of the Growth and Vitals sections as all fields require numerical input.

The units of measurement were given to keep the input consistent across users. A text box was provided to the user to make note of any abnormalities within the measurements. The right column of the screen involves examining the physical areas of the patient. The user is provided with a defined list of commonly inspected areas. Again, this limits the amount of time a user has to spend manually typing the name of every physical area examined. An "Other" option is provided in case the user does not find the desired area. Each physical area is given a description and a status rating ranging from poor to excellent. Although status ratings are uncommon, we decided to implement them so that users can quickly identify the condition of the physical areas by reading the status versus reading each description. The list view is used to store documented physical areas and makes it convenient to modify information by clicking on the area. A similar rating is given to a patient's overall health as it serves the same purpose as a summary.

Figure 33 - Demonstrates ordering tests for the suspected diseases and documenting their results.

After thoroughly examining a patient, tests are ordered based on the suspected diseases to either confirm or deny suspicions. The Test Selection screen in Figure 33 contains a table with the most commonly used tests for Malaria, Meningitis, and Tuberculosis as well as a generic test for other illnesses. This table functions similar to the table in the Symptoms Selection screen in Figure 31.

For the test results section, a pagination element was used to simplify switching between different test results. A pagination acts similar to a slider in which results are moved to the left if the user presses the right arrow button and vice versa. It also provides the user the convenience

to jump to whichever result they choose by selecting a numbered button in between the arrows. Few tests are similar in terms of input however, a majority require different numerical measurements and analyses in order to determine the final result. In order to accommodate for unanticipated data, each test contains a notes text box where any additional information can be documented. Each test can result in three outcomes: positive, negative, or inconclusive. Ideally results should either be positive or negative however, in cases of X-rays (Figure 33) where results are determined solely from observation, there may not be enough satisfactory data to confirm or deny a suspicion.

Treatment Selection

Figure 34 - Displays the patient's drug allergies and current medications as well as demonstrating the treatment selection process.

The final stage of the examination process involves assigning the patient treatment. The left hand column in Figure 34 provides the user with information that influences the final treatment such as drug allergies and the current medication that a patient is taking. In the right hand column, the user can select any of the predefined list of medication and add it to the list of treatments. For each treatment, a dosage can be provided with specific instructions if needed.

The user has the option of adding or removing drug allergies if they find that the list is not accurate. A user can add drug allergies without restrictions however, removing drug allergies requires approval from an administrator. A listview was used as opposed to a table since only the name of the drug is needed to prevent prescribing it as treatment. On the other hand, a table is used for current medications as specifics such as dosage can be used to determine treatment selection. The treatment selection process can be done in two forms. The user can either add all the treatments first and then fill the dosages later, or add the treatment and dosage one at a time. A save button was provided so that the user would have an understanding of when the dosage information was saved. We decided to have the dosage be a text box primarily because dosages do not provide beneficial information towards disease prevention therefore, the information need not be selectable or specified for data analysis.

Examination Review

Figure 35 - Displays an example of a final report screen which contains all saved information throughout the examination.

The screen in Figure 35 provides the user with a review of all the sections from the examination process. Each section is titled and all fields are non-editable. Although editing information in this screen would be convenient, it would destroy the purpose of the other examination screens. The goal was to break the examination process into sections to help reduce cognitive load on the user making a lengthy task, such as an examination, easier to complete in smaller steps. Having separate fillable sections creates a mental model of "building" a report and Figure 35 serves as a review for the final product. To enforce this idea, an edit information button was added to each section, bringing the user to the respective screen where changes can be made. Once the user is

satisfied with the report, they may choose to save it as a pdf file to print and hand to the user or

as an xml file to transfer data to other medical facilities. Otherwise the user will select the done

button which saves the report to the assigned patient and returns the user to the Dashboard

Screen.

Examination Progress

Figure 36 - Demonstrates the completion of sections of the examination. A check represents a
completed section; a dash represents a skipped or incomplete section; an empty box represents a
section that has not been viewed.

To the left of the examination screens, there is a panel that outlines the structure of the

examination process as shown in Figure 36. This screen allows a user to easily view the progress

of the examination, and each section's completion status. A check indicates a section has all the

required information completed, a dash indicates that a section has not completed all the required

information, and an empty box indicates that a section has not been viewed yet. A user may also jump to any section by simply clicking the name of the section.

The design approach for this screen was to imitate a checklist. Medical facilities that use a paper based system have to fill patient information in a certain order and make sure the necessary information is filled out in order to proceed to the next section. We wanted to bring the familiar use of a checklist on a clipboard and translate it to our screen design. The task of the checklist is displayed at the top with the patient being examined underneath to serve as a reminder to the user as to what the completion of the steps accomplishes and who the task is being accomplished for. By providing the user a visual representation of a checklist, there is a stronger inclination to complete all parts of the task due to the positive/negative feedback given from it.

<u>PDF Report</u>

Figure 37 - Demonstrates a sample report filled with information collected through the examination process.

A user has the option of saving a PDF report after an examination process has been completed as shown in Figure 37. The report is formatted in a table-like fashion where headers for the columns are displayed above at the top of each table as demonstrated in the vaccination section. Tables present information in a more organized fashion resulting in a report that is easy to navigate and read. Personal and vital information are displayed with the labels inside the cells in order to group the information by rows. The report is given in the same structure as the stages of the examination to maintain consistency however, not all information is printed. The intention of this report is to be printed and handed to the patient to inform them of significant findings from the examination. For example, the Gold In Tube lab test contains fields for physicians to input the number of antigens and mitogens which can be used to determine the result of the test. Those

numbers do not mean anything to the patient however, the end result of the test (positive/negative/inconclusive) and the physician's remarks for the test are of importance to the patient.

Search

Figure 38 - Displays patient's health profiles based on the selected patient.

A section of our application was dedicated to help users find all information on patients. The screen in Figure 38 is divided into two sections. The left half provides the user with the search functionality while the right half displays the selected patient's health profile. The health profile contains a table of examination records, assigned medical staff, vaccinations, etc. as well as when data was last modified and by whom. If the user double clicks a patient from the table, the left half of the screen is replaced with the patient's standard profile which consists of personal information.

The search functionality works similar to that of selecting a patient for an examination in Figure 30. The main difference is addition of a gender, date of birth, and phone number column to the patient table. Not all personal information could be displayed due to limited spacing therefore, the addition of those three columns was done to uniquely identify the desired patient. Because all information is presented in this section, the user also has the capabilities of modifying the information. For example, in the search portion of the screen, a user can add/delete patients if the patient isn't registered or has not been seen for some time. Again, certain actions such as deleting patients needs to be approved of by a higher level user to ensure data integrity.

Request

Active Users
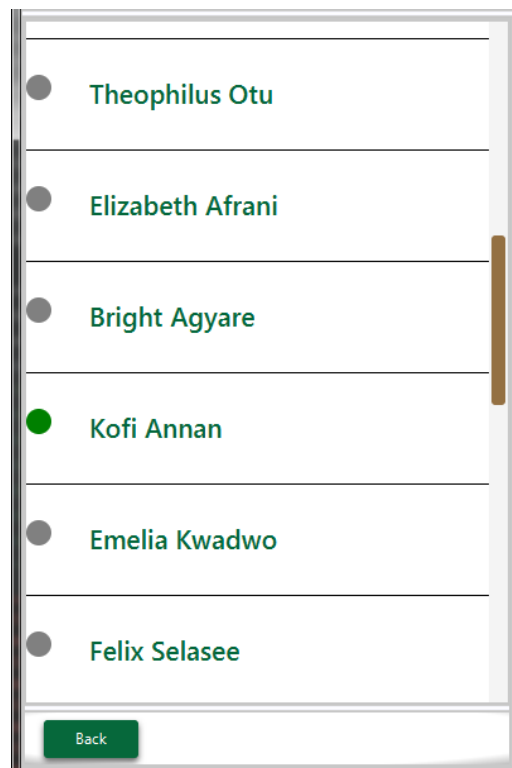
Figure 39 - Displays a list of admins registered with the application and their active status.

When a user opens the request portion of the application, a panel with a list of all the registered administrators is shown on the left side of the screen as seen in Figure 39. A circle is used to indicate the active status of the administrator. Green indicates that the user is currently signed into the application while gray indicates the opposite. This design style was chosen as it resembles the design of common social media websites thus making the interface more familiar with the user. An active status allows for a user to know how frequently requests are going to be handled as well as who should and should not be logged in. Ideally there would only be three to five administrators registered with our application however, for demonstration purposes, more were added.

Request Pool

Figure 40 - Demonstrates a request pool for administrators who can either approve or deny incoming requests from other users.

The screen shown in Figure 40 is where administrators approve or deny incoming requests. The requests are shown in the table with four columns: type, status, sender, and date received. There are several types of request such as deleting patient, user promotion, user registration, and other data modification. A registration request as shown in Figure 40 is received when a user registers a new account with the application. Hospital administrators approve, deny, and view request details by selecting a request from the table. Because requests can be sent at any time, the table request table refreshes every minute to receive the latest information. A refresh button was provided to the user to allow them to confirm their decision on a request without having to wait for the table to refresh on its own. A user can see their decision if the "Hide closed requests" checkbox is unchecked. The checkbox was added to prevent cluttering the table with handled and unhandled requests. If a request is approved, the appropriate action is taken and updated, otherwise it is removed from the pool.

Data Analysis

Parameter Selection

Figure 41 - Demonstrates the fields preselected for the pie graph in Figure 42.

In order for a user to be able to display a graph with the desired information, certain parameters need to be selected. A panel on the left side of the screen as shown in Figure 41 allows the user to select a chart type, a category, and several filters. Currently the analysis is limited to pie charts however, future implementation could include line graphs, scatter plots, and heat maps which would give a broader range for data analysis. A pie chart allows a user to view data as parts of a whole. In the example of suspected diseases, a user can identify which disease is suspected the most and the least.

The user was also given a selection of filters. Filters allow the user to hone in on a specific group of patients. In the case of suspected diseases, a user can identify which disease is more prevalent based on location, age, or gender. Once a user has finished selecting their parameters, the Show Graph button is pressed to display a graph on the screen in Figure 42. A Reset Fields button was also added for convenience when executing multiple queries so that the user does not need to change each field every time.

Graph Representation
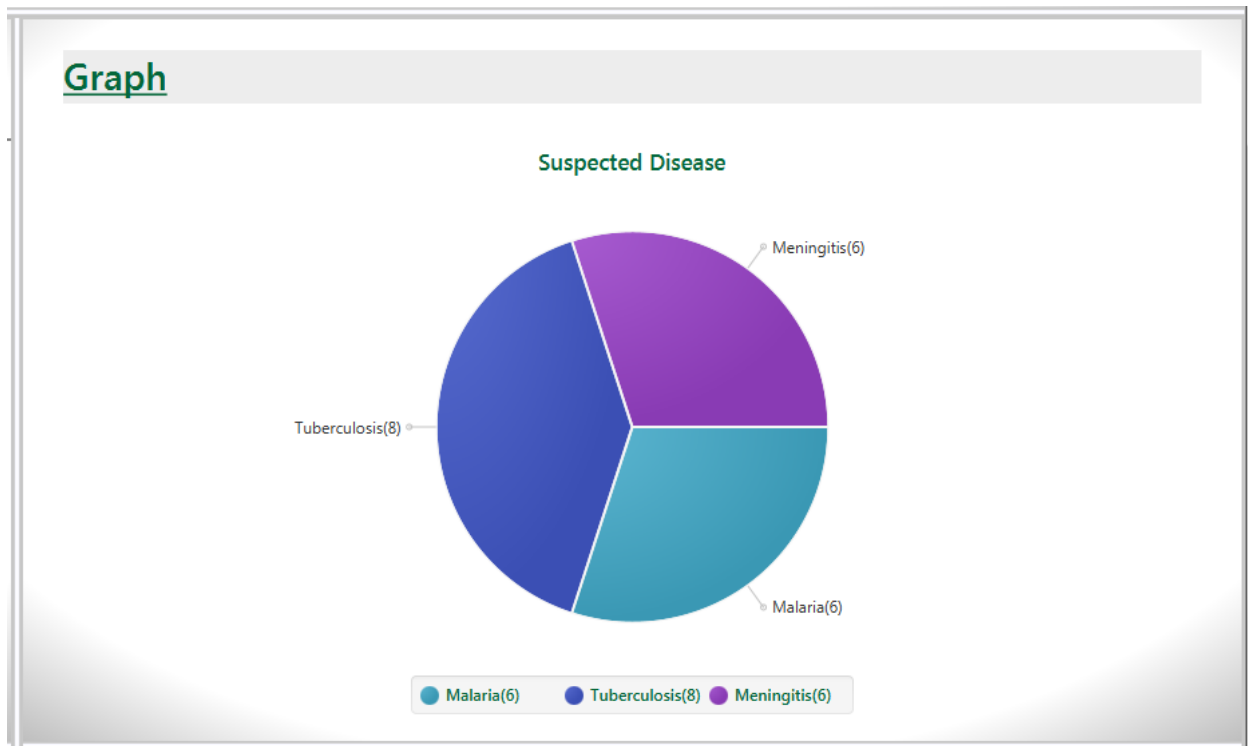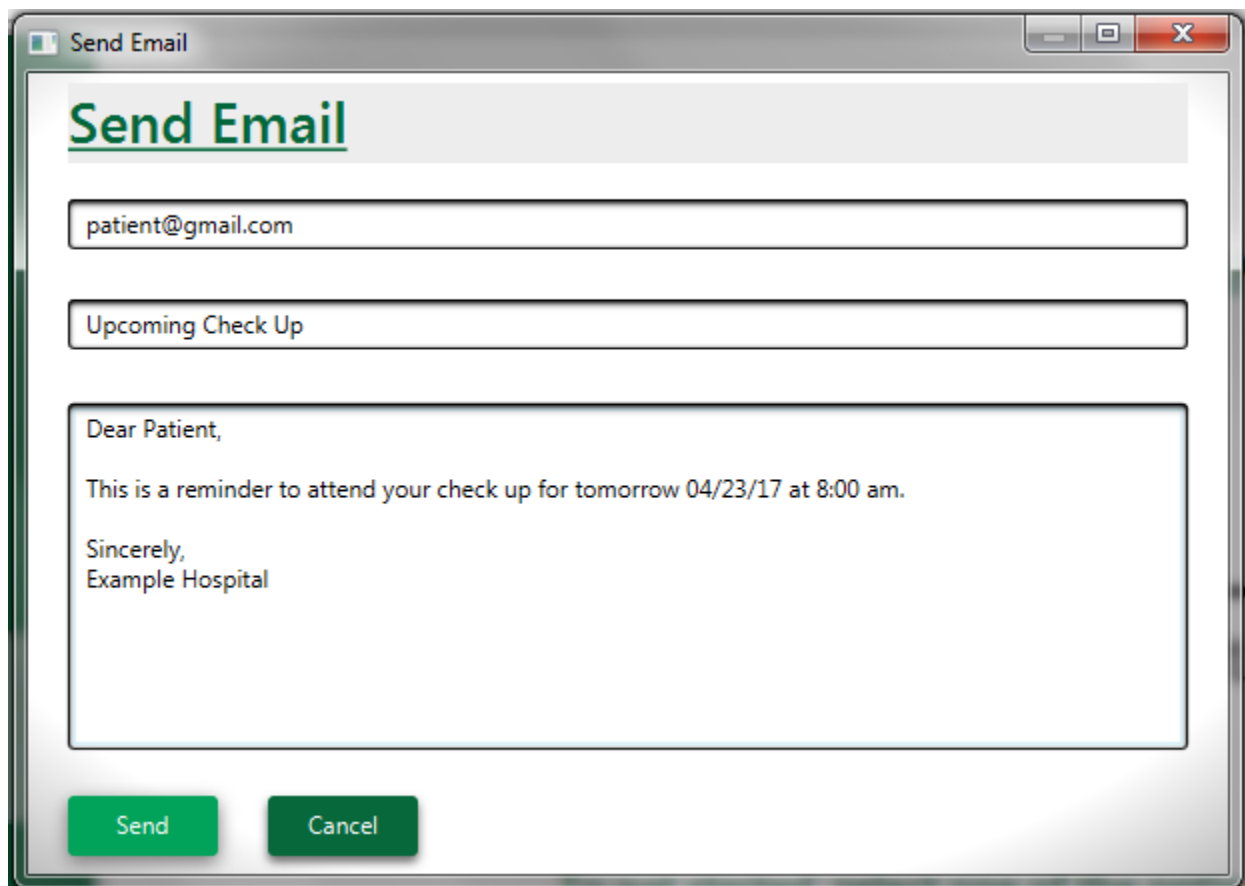
Figure 42 - Demonstrates queried data in the form of a pie graph. All data shown was randomly generated.

Figure 42 displays how a pie chart is displayed in our application. The title of the chart is set from the category parameter in Figure 41. The pie chart uses an analogous set of colors to clearly differentiate between pieces while indicating each piece's contribution to the same query. For

each new query, the primary color changes which helps the user to distinguish between pie charts. The number of patients for each piece is given next to the name to give the user precise information. With exact numbers, other calculations can be done such as percentages or averages. In addition to precision, the data shown was intended to be anonymous since all users have accessibility to the data analysis section of the application.

Send Email

Figure 43 - Demonstrates an example email being sent to a patient serving as a reminder for an upcoming checkup.

To enhance the physician-patient interaction, our application includes a simple email feature as shown in Figure 43. Each patient has a primary form of contact, one of which is by email.

Although each user is able to send emails through their own email service provider, we made it more convenient to send basic messages from a general sender name such as do-not-reply@vermams.com instead of having the physician use his or her personal email address. The feature requires the user to input the receiver's address with an optional subject line and message box. Once a user presses the Send button, a pop-up will indicate whether the email was sent successfully or not.

6.3 Architectural Design

We chose the Model-View-Controller for our design, as it provided optimum code structure, while providing some flexibility to further subdivide into feature-specific packages. MVC is technically a design pattern, which groups classes into packages, based on their purpose to the system. The Model refers to components that hold the data while the application is running, with the View responsible for displaying these to the user. Controllers provide an avenue for interacting with the application which could include making changes to the model. Using MVC encourages a separation of privileges by creating strict demarcations between the UI, the controllers that affect the UI, and the underlying data model.

6.3.1 Patterns
We employed a couple other design patterns in our application; the Factory and the Facade patterns.

Factory Pattern

The Factory pattern provides a class with static methods which facilitate the creation of objects on the fly. We use this design pattern in two areas: to spawn Lab Tests objects, and elsewhere for Request objects.

Facade Pattern

This is a package design pattern aimed at providing a simplified interface to access and perform actions on classes in a package. It greatly helped us in developing our examination feature, by abstracting the common functions of each examination section into an interface with much fewer methods.

6.3.2 Final Class Diagram

In order to accommodate the final designs described in this section, our initial class diagram had to be expanded significantly by the end of the project. Agile Kanban facilitated this incremental design with its continuous development approach. Additionally, active user involvement, another tenet of agile, provided a source of regular feedback from a domain expert, Dr. Mark Weber. These responses were then incorporated into our design in the above section 6.2, and necessitated the resulting final class diagrams.
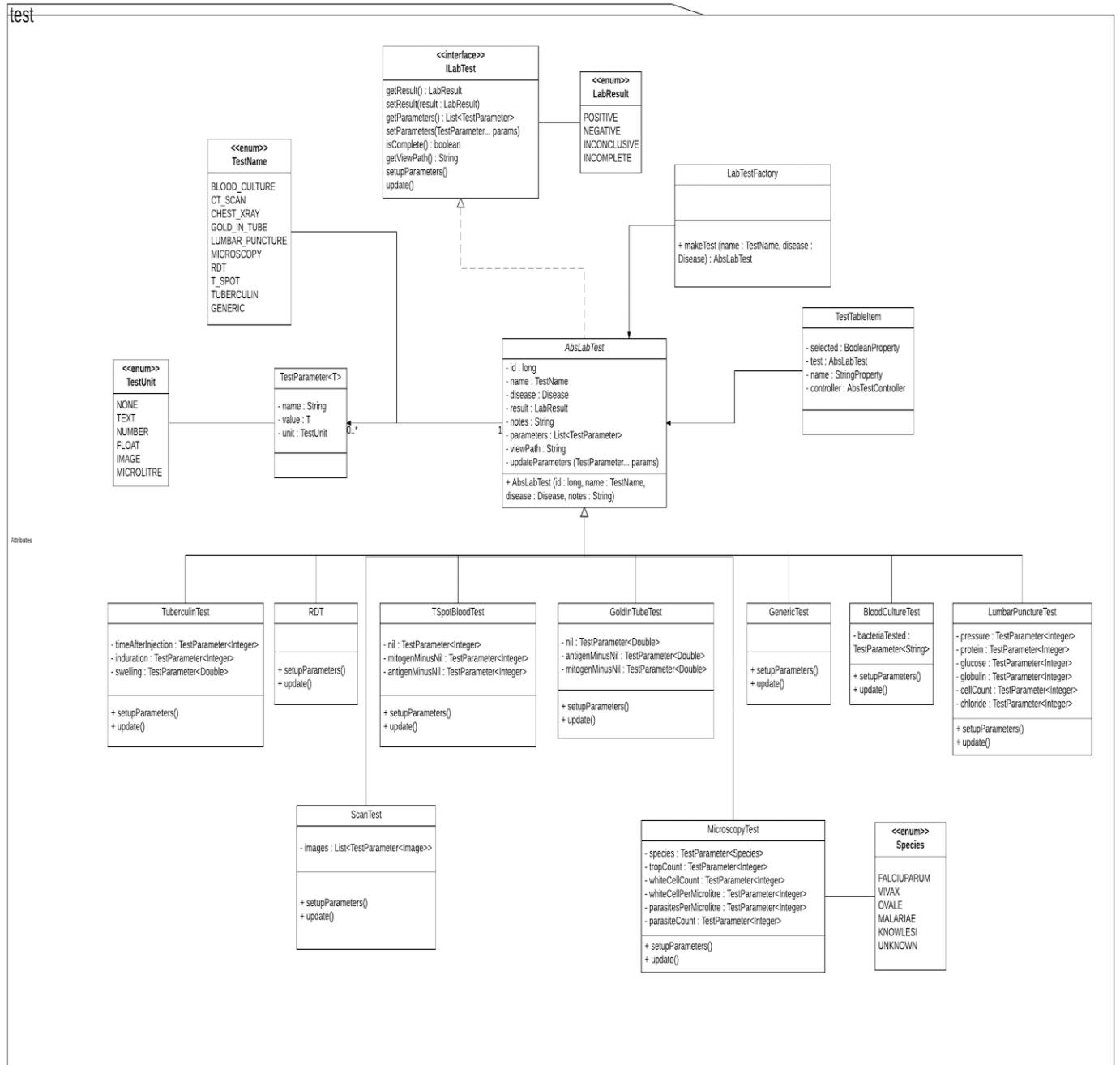
Figure 44 - Class Diagram (Entities)
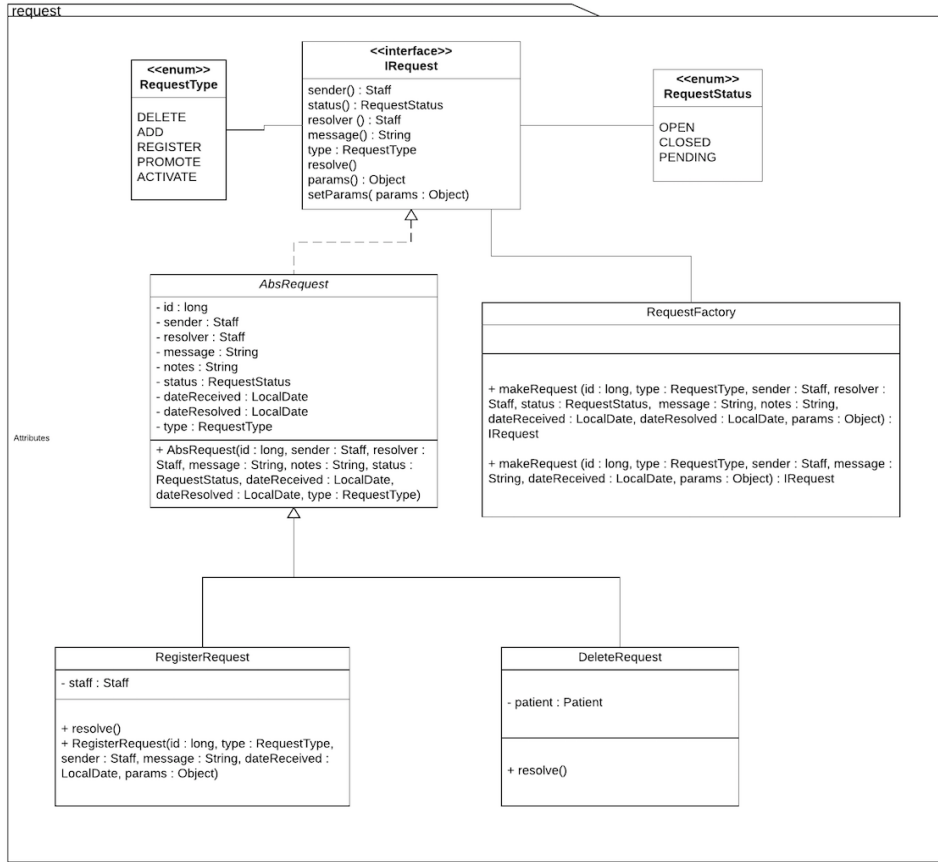
Figure 45 - Class Diagram (Entities)
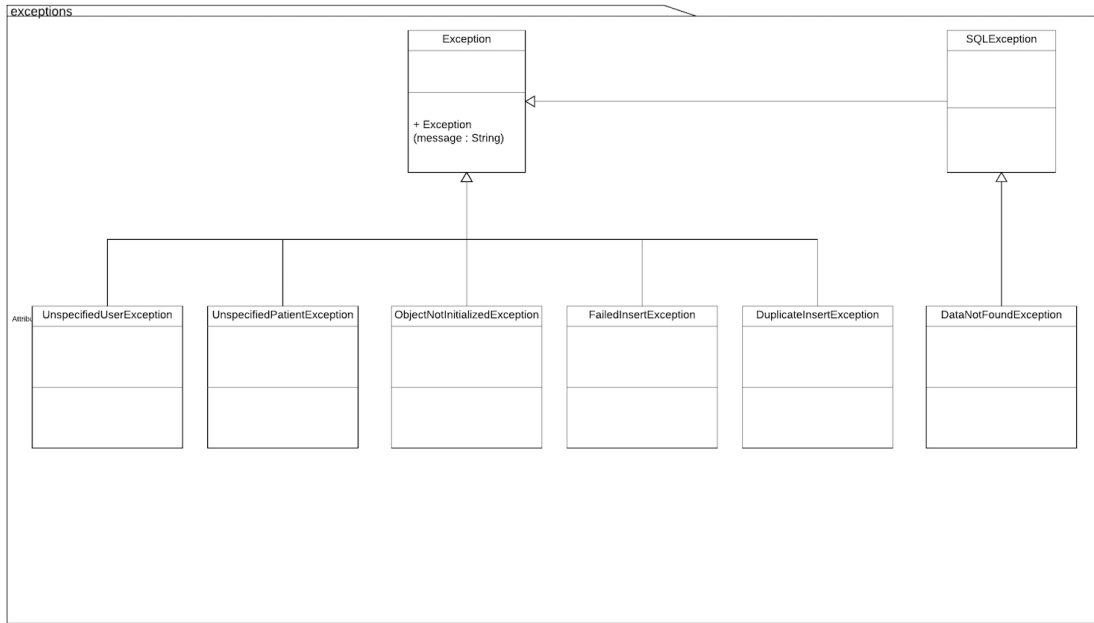
Figure 46 - Class Diagram (Entities)
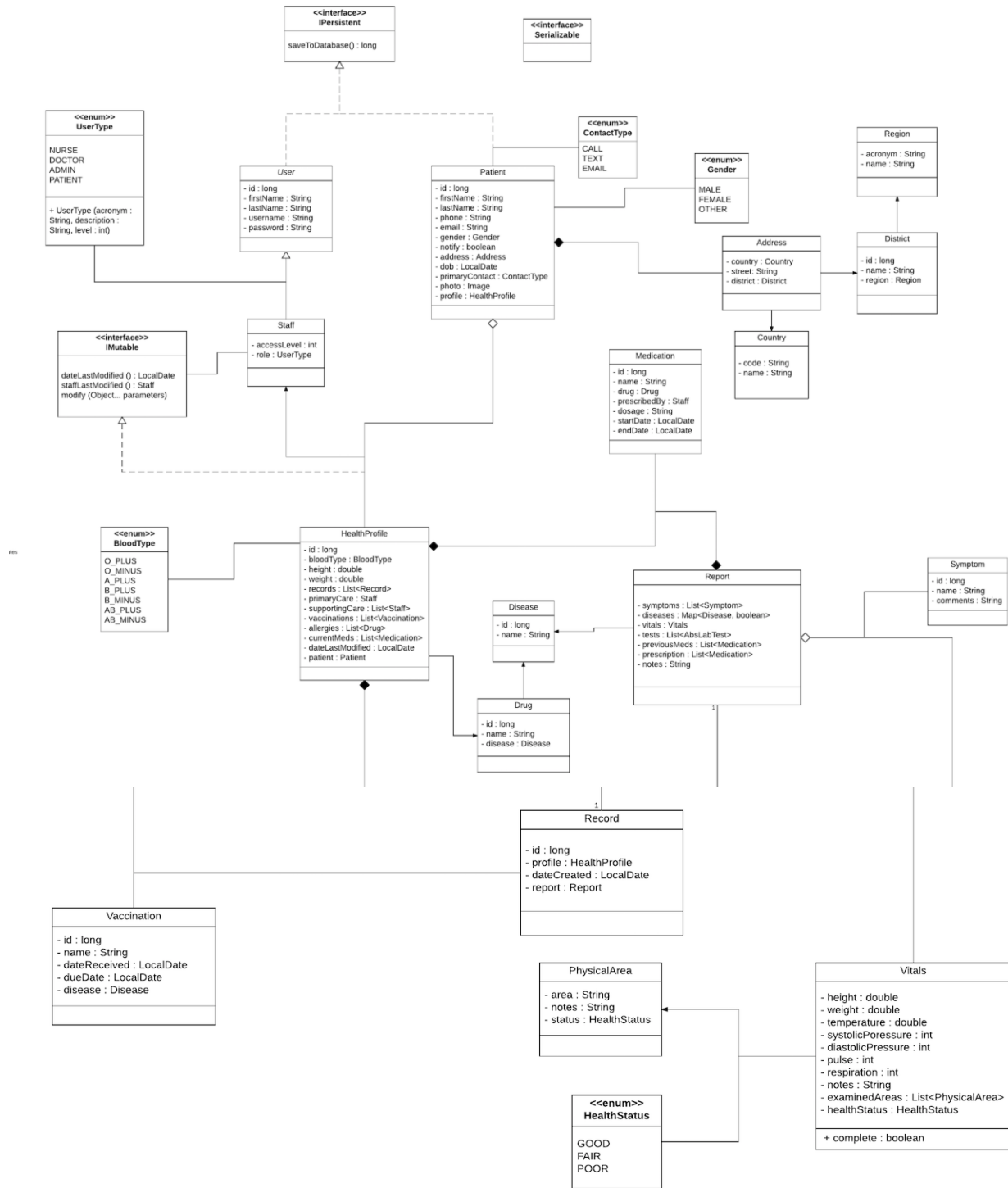
Figure 47 - Class Diagram (Entities)
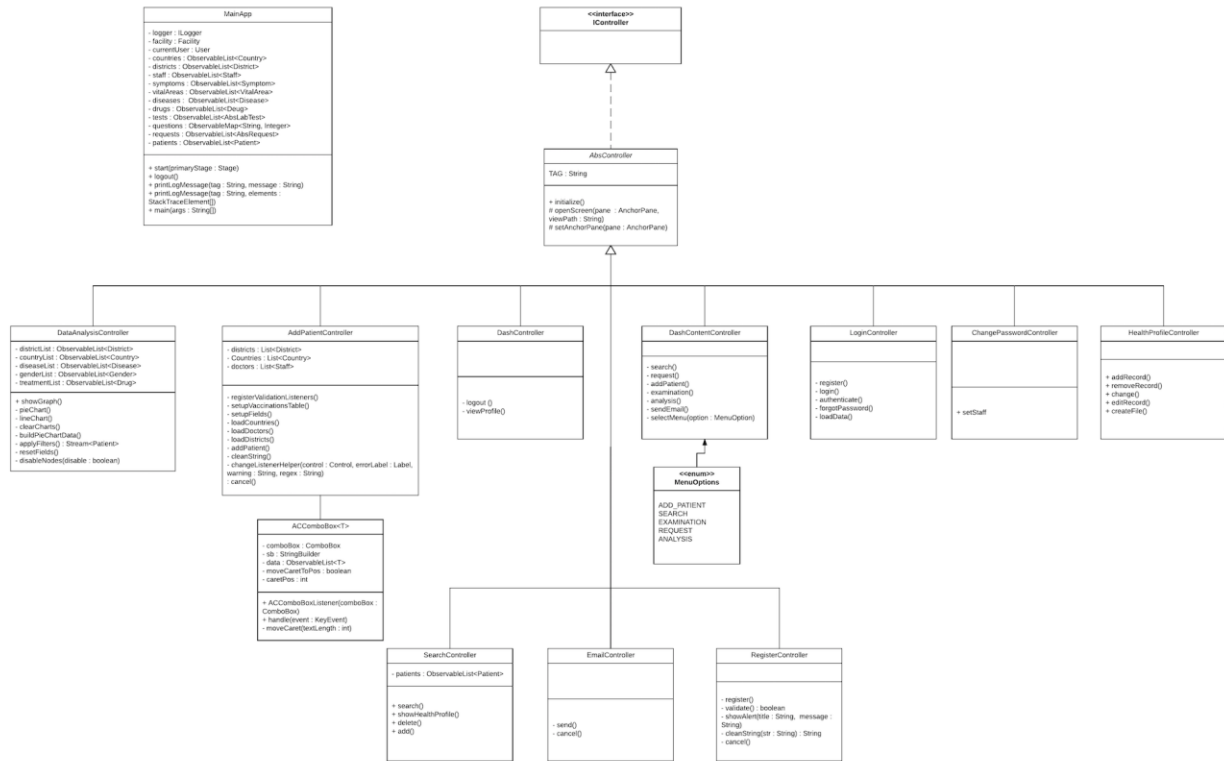
Figure 48 - Class Diagram (Controllers)
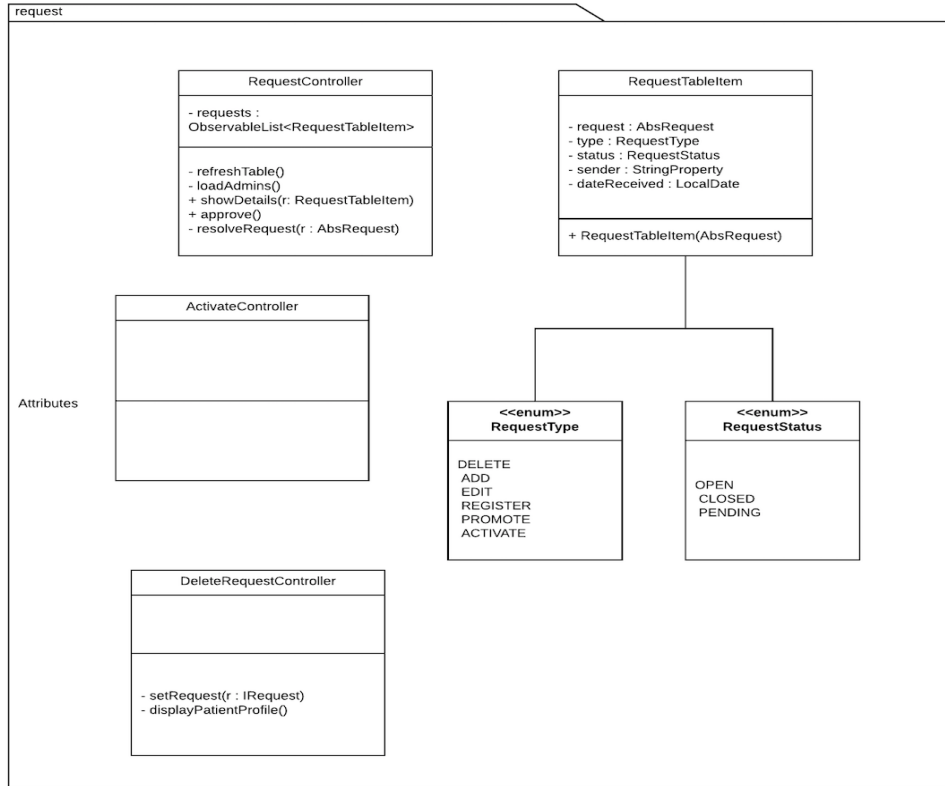
Figure 49 - Class Diagram (Controllers)
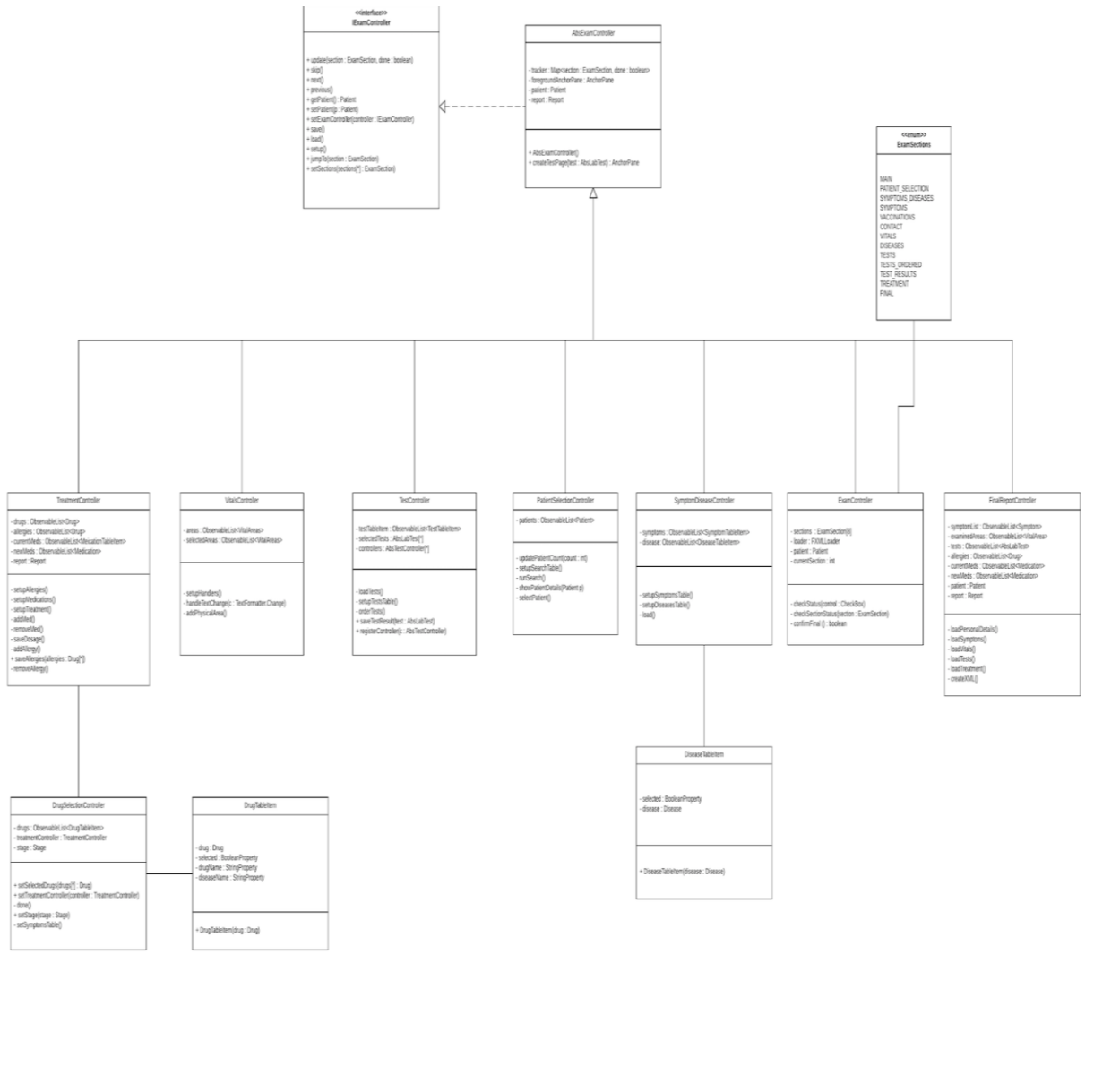
Figure 50 - Class Diagram (Controllers)

6.4 Security Design

6.4.1 Threat Assessment

In order to incorporate user experience along with security measures, a threat assessment was

written for the application. This helped us to document all possible threats to the application and

to determine our security solutions. The threat assessment is modeled after an example provided by the Open Web Application Security Project (OWASP) (Application Threat Modeling, 2015).

Threat Model Information
Application Version: 1.0
Description:
VermaMS is the first implementation of an application to assist healthcare providers in hospitals to manage patient records and visits. There will be three users of the application:

1. Registered Nurses

2. Doctors

3. Administrators

Nurses shall only be able to add a patient to the system and to update patient information during an examination. Doctors shall be able to add patient records, update patient records, and request that patient's records be deleted, search a list of his or her patients, and run data analysis. Each doctor shall only be able to have these privileges for patients that he or she is the primary medical care provider for. Administrators shall be able to view system logs, delete a patient's records after a request has been made by a medical staff member, and promote and demote other users via requests, and perform data analysis.

External Dependencies
The external dependencies of the application are entities outside of the application that end up threatening it by providing extra attack surface, which is the "reachable and exploitable vulnerabilities" (Northcutt, 2017) of a system, for potential attackers. These external dependencies are not under our direct control as the development team since we do not have the

ability to directly modify them (Application Threat Modeling, 2015).

ID	Description
1	The application will run locally on a personal computer.
2	The database server will be Oracle Database and it will run over WPI's Linux server. The server is hardened per WPI's policies. It is backed up on the cloud using Amazon AWS services, hardened per Amazon's policies.
3	The application will communicate with the database using an encrypted connection.

Table 6 - External dependencies of the application.

Entry Points

The entry points of VermaMS are the entry-level interfaces with which attackers can interact

with the application (Application Threat Modeling, 2015).

ID	Name	Description	Trust Levels
1	Login Screen	Medical staff, clerks, and administrators must login before having access to any other functionality.	(0) Anonymous user (1) User with invalid login credentials (2) User with valid login credentials

Table 7 - The entry points of the application.

For our application, the only entry points for attackers is the login page. Thus, it was necessary to

protect the passwords of legitimate users and to only allow hospital staff to create accounts with

the system.

Trust Levels

Trust levels are the roles that a user of the system can have. Different privileges are assigned to

users of different trust levels (Application Threat Modeling, 2015).

ID	Name	Description
0	Anonymous user	User with no valid login credentials and has no permission to do anything.
1	New Authorized user	User that is authorized to use the system, but cannot do anything until he or she has created an account and has that account approved by an administrator.
2	Registered Nurse	Registered nurses are able to add a patient to the system and to update patient information during an examination.
3	Physician	The physician can request that a patient's records be deleted, add a record for a patient, update his or her patients' existing records, search a list of his or her patients, and run data analysis.
4	Administrator	An administrator can confirm requests to delete patient records, view system message logs, promote and demote users, approve the creation of a new account, and run data analysis.

Table 8 - A list of each user role and its description.

As seen from the table, access control was a very important component for the application. User

roles are well-defined and their privileges should not be allowed to change. Anonymous users

and newly authorized users should not be allowed to enter the system at all, until their accounts

are created and approved by a user with the administrator role.

Assets

The assets of VermaMS are the data and actions that an attacker will be interested in obtaining or performing (Application Threat Modeling, 2015). These data and actions are especially sensitive, because they should only be performed by authorized users with the appropriate roles.

ID	Name	Description	Trust Levels
1	Administrators, Clerks, and Medical Staff	Assets related to users of the application.	Users that are authorized to perform this action
1.1	Medical staff login details	The credentials each medical staff member or clerk shall use to login to the application.	(2) Doctor (1) Nurse
1.2	Administrator Login details	The credentials for administrator-level users.	(3) Administrator
1.3	Patient data	Patient visit records	(2) Doctor (1) Nurse
2	System	Assets of the Underlying System	
2.1	Ability to Execute Oracle Database Read/Write Commands	This is the ability to execute read/write commands on the database, allowing one to retrieve information from or to write information to it.	(3) Administrator
3	VermaMS	Assets of the application	
3.1	Ability to promote users	Promoting a user within the system	(3) Administrator
3.2	Ability to demote users	Deleting a user from the system	(3) Administrator
3.3	Ability to approve request to delete a patient's records	Approval for a patient's records to be permanently deleted from the system.	(3) Administrator

3.4	Make Request to delete Patient record	Ability to make a request to have a patient's records deleted	(2) Doctor
3.5	Request addition to system as user	Ability to request an account within the system	(0) Anonymous User
3.6	Approve addition of user to the system	Ability to approve that a user with the given account credentials be added to the system.	(3) Administrator

Table 9 - A list of the assets of the system, along with a description and list of trusted users for each.

Many features of our application are sensitive, since they deal with private patient data. Therefore, it is important to protect these assets with proper authentication and authorization measures.

Threat Categorization

For each specific threat, we categorized it using Microsoft's STRIDE Model, since it is easy to understand and quick to implement. The STRIDE model is described below (The STRIDE Threat Model, 2005):

STRIDE Component	Description of STRIDE Component	VermaMS relevance
Spoofing Identity	An unauthorized user gaining access to a user's login credentials.	VermaMS should not allow an unauthorized user to gain access to an administrator's, clerk's, or physician's username and password.
Tampering with Data	Modification of data that should not change.	VermaMS should prevent tampering with private patient records except by the patient's primary physician.

142

Repudiation	Users can perform unauthorized actions and there will be no proof that they have performed such actions.	VermaMS should log all sensitive operations that are performed to ensure only authorized users are performing such operations.
Information Disclosure	Exposure of private information to users who are not authorized to view it.	VermaMS should prevent unauthorized users from viewing confidential patient records.
Denial of Service	A Denial of Service (DoS) attack is one in which services are made unavailable to authorized users.	VermaMS should implement necessary database defense measures to ensure that it does not go down.
Elevation of Privilege	An unauthorized user gains elevated privileges and can perform unwanted and harmful actions within the system.	VermaMS should prevent unauthorized users from gaining access to the system and prevent authorized users from elevating their access levels without proper prior approval.

Table 10 - A list and description of each element of the Microsoft STRIDE model, as well as how the application addresses each potential threat type.

Security Controls

VermaMS has been designed with the following security measures to mitigate threats as

described in the Threat Categorization:

Security Model Component	Definition	Threats Addressed	Implementation Solutions
Authentication	The "process of confirming the correctness of the claimed identity" (*SANS Glossary of Security Terms,* 2017).	Brute force, rainbow table, dictionary attacks	Passwords will never be transmitted as clear text, and will also never be stored as clear text in the database. Each user must answer a set of security questions in the event that he or she needs to change his or her password.
Authorization	The "the approval, permission, or empowerment for someone or something to do something" (*SANS Glossary of Security Terms,* 2017).	Privilege elevation	VermaMS operates on the principle of least privilege, granting users the ability to perform only those actions that are necessary for that user to perform.
Data Integrity	The prevention of unintended or malicious data modification while it is in transit or on the database.	SQL injection, improper input	All data that is entered will be validated and scrubbed (to prevent SQL injection) before being stored in the database. All connections to the database shall be encrypted. All default accounts on the database shall be removed.

			VermaMS shall use leading cryptographic standards to ensure that passwords stay secure.
Non-Repudiation	The "ability for a system to prove that a specific user and only that specific user [performed an action] and that it hasn't been modified" (*SANS Glossary of Security Terms,* 2017).	An attacker covering his or her tracks after successfully exploiting a vulnerability	Whenever a sensitive operation is completed, authorized hospital staff shall be able to view a log of it later to ensure that no unauthorized users have gained access to the system.

Table 11 - A list of the main security model components, a description of each, and how VermaMS implements each.

For our application, authentication, authorization, and data integrity were of special concern. Our focus was to protect private patient information from being viewed by anyone except that patient's health care providers.

DREAD Risk Ranking

We decided to use Microsoft's DREAD model for ranking risks, since it covers all of the aspects of a vulnerability and is easy to understand. The DREAD model is broken down as follows (Threat Modeling, 2003):

DREAD Model Component	Description
Damage Potential	How much damage can be done if an attacker discovers this vulnerability?

Reproducibility	Would it be easy for someone else or the same attacker to perform the exploitation again?
Exploitability	How easy would it be for a potential attacker to attack the system to exploit this vulnerability?
Affected Users	How many users are affected?
Discoverability	How easy is a vulnerability to find? Is it trivial, that someone with basic knowledge could discover it, or is it so obscure that someone would need a deep understanding of the system to find it?

Table 12 - A list of the components of the Microsoft DREAD model, along with a description of each.

For each threat, we used a simple 3-point scale to rate the risk of each: 1 for low-risk, 2 for medium-risk, and 3 for high-risk. For the overall rating of the potential threat, we took a total of all of the components of the DREAD assessment (for that specific threat) and then assign it a marker of low (1-4), medium (5-9), and high (10-12). The DREAD assessment for VermaMS is shown below:

Threat	D	R	E	A	D	Total	Overall
Attacker can inject SQL commands into the application.	3	3	2	3	3	11	High
Attacker cracks a user's password by brute force or other method.	2	1	1	3	2	9	Medium
Attacker gains direct access to the database.	3	3	3	3	3	12	High

User or attacker enters in invalid input and causes undefined behavior.	1	3	3	1	3	11	High
Attacker extends class to input malicious functionality.	3	1	1	2	2	9	Medium

Table 13 - A list of each threat to the system with a DREAD evaluation for each. Each DREAD evaluation is totaled, and a risk rating (low, medium, high) is assigned to each threat.

From our DREAD analysis, it was clear that SQL injection, unauthorized database access, and entering invalid input were the main concerns. Additional concerns were the cracking of passwords and the malicious extension of application classes.

Documented Threats

For each threat that we did a DREAD analysis for, we documented it. Each threat has its threat target, risk rating, attack technique(s) used to carry out the attack, and countermeasures explained in more detail.

Threat Description	Attacker injects SQL commands into the application.
Threat Target	Application database
Risk Rating	High
Attack Techniques	SQL Injection
Countermeasures	Sanitize input, use prepared statements

Table 14 - SQL Injection threat description and countermeasures.

Threat Description	Attacker cracks a user's password by brute force or other method.
Threat Target	Application user authentication
Risk Rating	Medium
Attack Techniques	Brute force, rainbow table attack
Countermeasures	Never store clear-text password or expose stored password in clear-text, enforce good password composition, employ security questions when a user requests to change his or her password, use good password encryption practices (salt, hash, algorithm)

Table 15 - Password cracking threat description and countermeasures.

Threat Description	Attacker gains direct access to the database.
Threat Target	Application database
Risk Rating	High
Attack Techniques	Unauthorized privilege elevation, authentication bypass
Countermeasures	Always encrypt connection to database and database traffic (SID), apply database updates regularly, lock down PUBLIC role

Table 16 - Unauthorized database access threat description and countermeasures.

Threat Description	User or attacker enters in invalid input and causes undefined behavior
Threat Target	Application front-end
Risk Rating	High
Attack Techniques	Entering in data outside of reasonable ranges
Countermeasures	Validate input, use BigInteger class when doing arithmetic on large or small integers, handle underflow or overflow exceptions in an acceptable way.

Table 17 - Invalid input threat description and countermeasures.

Threat Description	Attacker extends class to input malicious functionality.
Threat Target	Application front-end
Risk Rating	Medium
Attack Techniques	Extending functionality maliciously and uploading the cracked version to a computer, or distributing it freely.
Countermeasures	Declare important classes as final, such as the class that handles password validation. We accept the risks from the other classes, because we want other project students or developers to be able to extend upon our software.

Table 18 - Malicious class extension threat description and countermeasures.

By doing this analysis, we were able to focus our efforts in implementing robust security

measures to help prevent attackers from gaining access to the sensitive areas of the application.

6.4.2 Authorization

Authorization - Access Control

One of the most significant threats to the application is adversaries gaining any access privileges,

and legitimate users gaining excess access privileges. To combat this, VermaMS was designed to

follow the principle of least privilege which means giving users only the permissions they need and nothing more. The application enforces role-based authorization in which user privileges are assigned based on the user's role. Three user types exist in VermaMS: nurse, doctor, and admin. Nurses are authorized to add a patient to the system and to update patient information during an examination. Doctors are authorized to add a patient to the system, update patient information during an examination, request that a patient's file be deleted, and to conduct data analysis. Admins are authorized to respond to requests (such as to delete patient records) and to conduct data analysis. In order to streamline the process of assigning roles and session management, VermaMS incorporates Apache Shiro. Shiro is a security-minded API developed by Apache that makes securing applications easy and effective.

6.4.3 Password Security and Storage

VermaMS follows best practices when it comes to password security and storage. When a user creates a password, it is required to be at least 8 characters long, and to contain at least one of: a special character, a number, and an uppercase letter. This allows for a strong password, but does not burden users of the system with long passwords that they must memorize.

After the user creates his or her password, it is hashed using 40,000 iterations of PBKDF2 (Password-Based Key Derivation Function). The function uses HMAC-SHA-512 (SHA-512 is part of the SHA-2 family) for its core hashing operation, which includes using an 8-byte (64-bit) long securely and randomly generated salt value. The HMAC-SHA-512 function hashes the password with the salt, and then runs again with the plain-text password and the hash computed during the previous iteration (Ducklin, 2013). This process is repeated for the rest of the

iterations. The PBKDF2 function keeps a secure running accumulation (using a bitwise exclusive

or, XOR) of the results of the HMAC-SHA-512 iteration outputs (Ducklin, 2013). 32 bytes of the

output from this function are stored on the database as the hashed version of the password

(Ducklin, 2013).

Using 40,000 iterations of the function ensures that the time it takes to hash many passwords is

extremely slow, and it will help protect passwords (if they are compromised and extracted from

the database) from dictionary attacks. A dictionary attack occurs when an attacker builds a

"dictionary" of common passwords, and pre-computes hashes for those plain-text passwords. He

then writes a simple program to compare all of his pre-computed hashes against the hashes

stored in a victim's password file. If there is a match, the attacker has discovered a correct

password.

Using a salt value ensures that there is less of an incentive for an attacker to perform a dictionary

attack (M., 2013). With a salt value, an attacker cannot generate a dictionary or rainbow table of

hashes once for all of the passwords of the dictionary, he must generate separate dictionaries of

hashes for all of the users in the database, since salt values are unique for each user (M., 2013).

For example, a plaintext password "Hello" with a salt of "1" hashes to:

6A68902DA1F60F0F96FB1693BCE159F6BC531EB88EECEF3A159CEC522A

E451B84221FBDF84AEA61F9643F8909290E0ED7BBF392476D3A888BE580

D44E15D2F0E

Using the SHA-512 algorithm. However, "Hello" with a salt of "2" hashes to:

B656B9EECB465D2501BA21FB54EA8E801B616A81756D730EB46113A8B6

2BC31499F78181483EBAC2995B9A383B62E35BCD4ADD000DD4FDF060D

C84CE54C28B51

After a user creates his or her password, the hashed password and salt are stored on the database and is associated with that user. The password is never stored in plain-text, because an attacker could get all of the passwords for all users if he or she is able to successfully compromise the database. When a user logs in, the password that is entered is hashed in the same way and the hashes are compared. This way, even if the passwords (in hashed form) are compromised, the attacker must perform time-consuming operations to crack the passwords by hashing them and comparing the hashes.

6.4.4 Database Hardening

SQL Injection Prevention
SQL Injection attacks allow someone to extract sensitive information from a database. In order to help protect the database from SQL injections, all input fields in VermaMS that lead to queries from the databases are made with PreparedStatements, which are Java's version of parameterized queries (SQL Injection Prevention Cheat Sheet, 2016). This way, the database can "distinguish between code and data, regardless of what user input is supplied" (SQL Injection Prevention Cheat Sheet, 2016).

Connection Encryption

To prevent an attacker from intercepting network traffic to view information being sent to or retrieved from the database, all connections to the database are encrypted. In order to encrypt database connections, Oracle recommends enforcing the use of SSL for every query to the database (Jeloka et al., 2012). This helps prevent an attacker from intercepting and copying or modifying any data and getting user passwords or private patient information, since all transmitted data is encrypted.

Removing Default Accounts

With every installation of an Oracle database, there are default user accounts that come pre-configured. If the passwords of these default accounts are not changed, then an attacker can easily gain access to the database through one of these default accounts. Oracle recommends changing default account passwords (Jeloka et al., 2012). We took this one step further, however, and removed all PUBLIC roles, since there is no reason for anyone but a database administrator to be looking directly at the database.

Database Credentials

Securely storing the password for the database is vital. If it is compromised, then the whole database is compromised: passwords, user information, and patient records. Since Java classes can be easily decompiled and viewed in readable form, the database credentials are stored on a .bat file that is set to only be executable and neither readable nor writable. This helps to ensure that only the application can access the database.

6.4.5 Java-Specific Security Measures

Since we decided to program VermaMS in Java, there are some measures that we took to thwart some Java-specific vulnerabilities.

Malicious Subclassing

One of the most useful features of Java is the ability to write subclasses for a class, so the subclass can share common functionality with its parent class while adding any necessary custom functionality. This feature can become dangerous, however, if an attacker is able to extend a class that performs sensitive operations and override those operations to do something malicious. In order to protect these sensitive operations (such as verifying a password entry or creating a password), either the class is declared final, or, if the class is not declared final (due to extensibility considerations), all sensitive methods are declared final or private. This prevents an attacker from maliciously extending these methods and classes to perform harmful actions.

Finalizer Attacks

In Java, a finalizer attack is a malicious use of a class's finalize() method. A finalize method is usually run by the garbage collector once it determines that the object is no longer being used, but before that memory housing the object is reclaimed (MET12-J. Do not use finalizers, 2017). If an attacker wants to gain access to a partially-initialized object, he can extend a class and override its finalize method. The attacker can then gain a reference to the object and use it. The zombie object can be used maliciously and can also bypass security checks (Mohindra & Snavely, 2017). In order to prevent the code from being vulnerable to this type of attack, classes are either declared final (preventing the extension of the class and overriding of the finalize method), or the class defines a final finalize method that does nothing. This prevents an attacker

154

from extending the class and writing a custom finalize method (Mohindra & Snavely, 2017).

6.4.6 Miscellaneous

Email

VermaMS allows doctors to send patients notification emails for things such as reminders to take

medication. When sending an email, VermaMS uses Google's Gmail SMTP server, and sends

the email over a secure TLS connection running on port 587.

7. Implementation

7.1 Kanban Implementation

The bulk of our implementation was done in the second term of our project (C-term), after enough requirements had been gathered. During these months, we relied on regular meetings to ensure the entire team was up to speed at any given moment, as well as software tools Slack and Trello. Slack was an extension of our meetings, giving us virtually perpetual communication channels. Figure 51 is a screenshot of an active Slack session. Trello, on the other hand, served as a means of distributing tasks among team members. We employed Trello's flexible card-based approach to make edits on the fly, as necessitated by our respective schedules. Additionally, a Trello extension supplied us a detailed burndown chart on-demand, keeping us aware of any timing constraints with the project. Figure 52 is a screenshot of our Trello board for C-term, accompanied by a burndown chart in Figure 53.
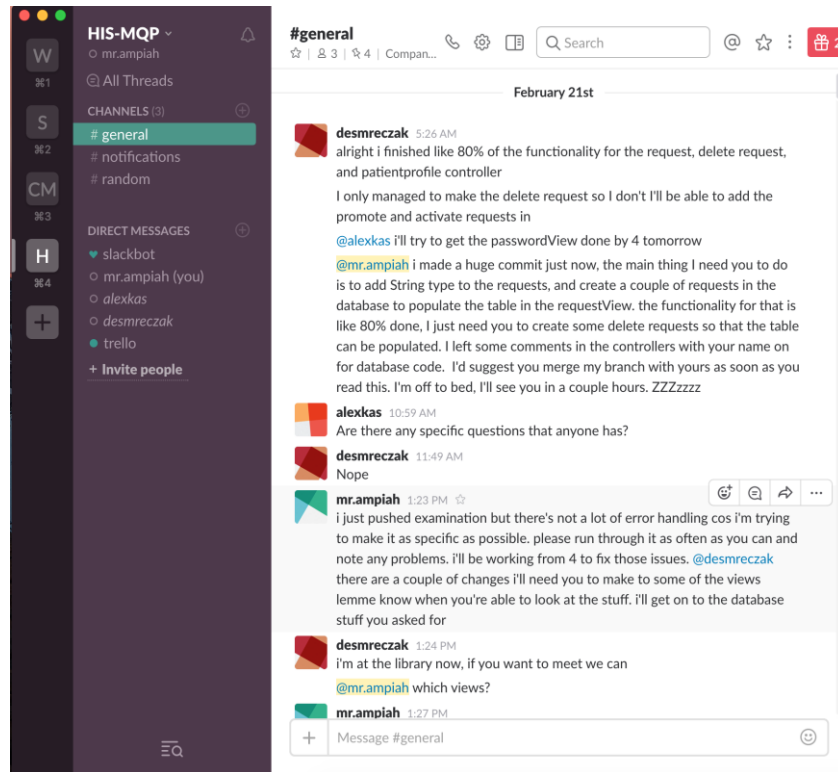
Figure 51 - Snapshot of Slack communication
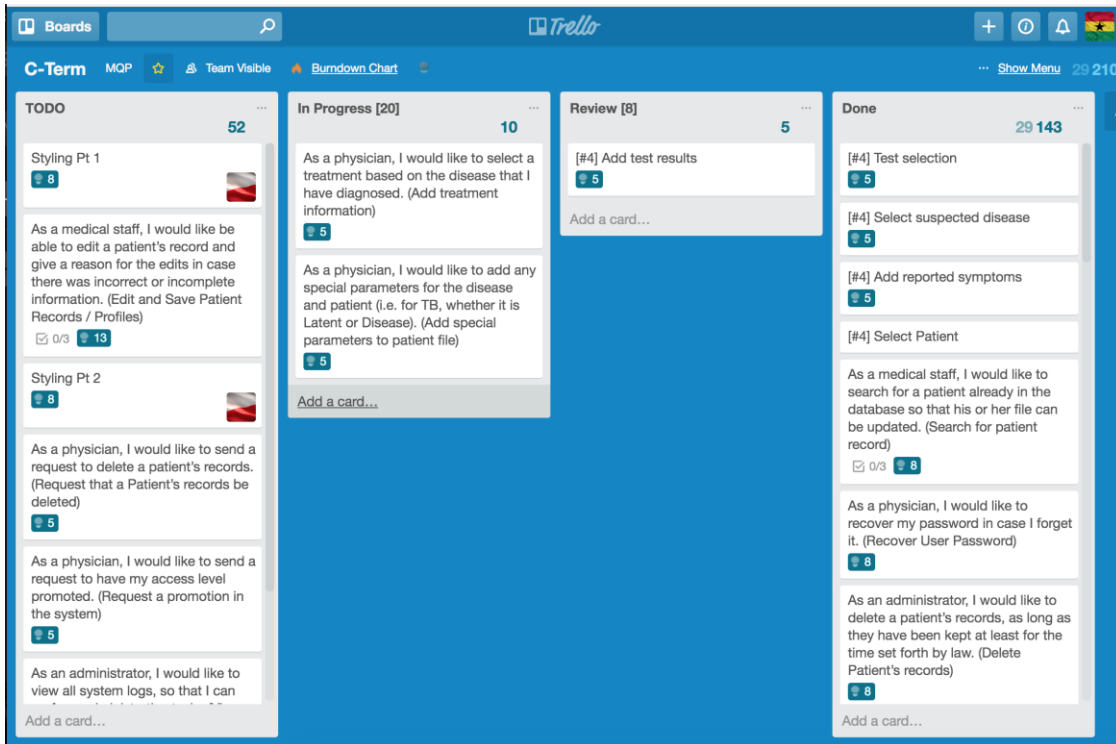
Figure 52 - Screenshot of Trello board
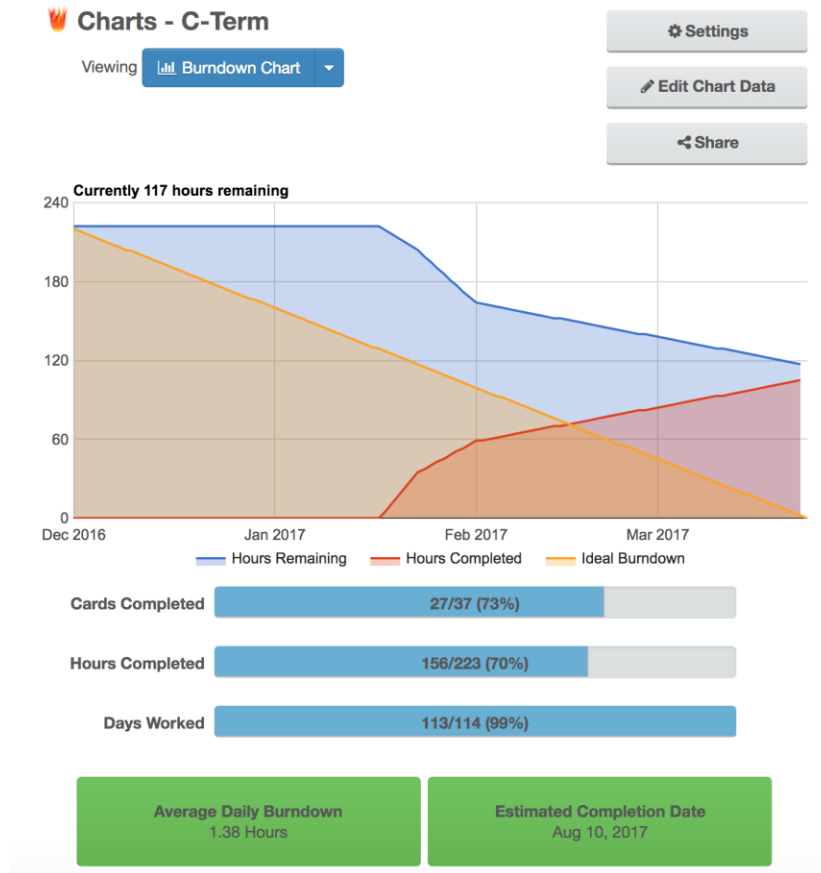
Figure 53 - Burndown chart

7.2 Development

During our development phase, we made several incremental changes. A majority of the changes were directly from Dr. Weber, our consulting physician. We also had to make a number of refactoring changes as the code base grew larger and systems grew more complex. The following sections highlight the major code adjustments made.

7.2.1 Examination
The two sections in our patient examination subsystem that were modified were symptom and

disease selection, and physical examination. Initially, we had thought to accommodate free-form

symptom input, to capture any possible symptoms a patient reported. However, after meeting

with Dr. Weber, we realized this could potentially dilute our sample data, and limit our data

analysis functionality. He suggested that a predetermined list of common symptoms, which the

physician could choose from, would be considerably more user-friendly and easier to perform

data analytics on by preventing user errors such as misspellings. We also altered disease

selection, to make provisions for cases where patients suffered from more than one ailment.

After implementing the aforementioned changes, Figure 55 displays our final version of the

symptom and disease selection page where Figure 54 was the initial version.

Figure 54 - Symptom and Disease Selection (Before)

Figure 55 - Symptom and Disease Selection (After)

7.2.1 Physical Examination

In this section of the patient interview process, we provided a simple interface to record patient vitals. We included text boxes for comments only on a select few body parts during the initial iterations of this section. Dr. Weber immediately warned us that there are more areas examined during the physical than the three we presented in Figure 56. After further research and analysis, a new design was created to accommodate more areas along with comments on the status of each individual area as shown in Figure 57.

Figure 56 - Physical Examination (Before)

Figure 57 - Physical Examination (After)

7.2.3 Refactoring Changes

A number of changes occurred as the project grew progressively. First, the examination controllers were heavily refactored in order to share several fields and functions, as well as some references to UI elements. The IExamController interface and AbsExamController abstract class were created because of the shared fields and functions. With these additions, a progress tracker for the examination section was developed to provide the physician with information on the sections that are yet to be seen, in-progress or completed. Lab test controllers were also refactored. By using a class with a generic parameter to represent test results, we easily abstracted a large number of common functions in the lab test classes. Similar abstractions were made amongst all controllers since nearly every controller had to include functionality that would switch the application's screens.

In addition to controllers, database access patterns endured significant change as well. Originally, every persistent object, such as patient and physician data, was obtained directly from the database. As we refined our patient model with the associated health profile, these tasks took more time to complete, and caused lags as long as half a minute in certain areas of the application. Furthermore, certain areas also began to exhibit similar lag after the implementation of our lab test models. To solve the lag issues, a preloader was implemented at the login screen to store data in memory which resulted in exponential advancement in speed. The use of a preloader was approved by Dr. Weber and enhanced with the addition of a manual refresh data button.

8. Assessment

8.1 Requirements

8.1.1 Functional

We successfully achieved our functional requirement goals by implementing a back-end database, using Apache Shiro, and designing our application to allow it to accept user input by following the MVC model. Implementing our application with an Oracle database back-end helped achieve our goals of being able to generate and save reports as well as enter, retrieve, and save patient data. By using Apache Shiro, we were able to achieve our goals of implementing authorization levels and restricting access to administrative functions. By following the MVC model, we implemented a way for doctors to send reminder emails to patients.

Medical Data - We were able to successfully implement this requirement by constructing our application in a way that follows a physician's workflow and patient interview process. We constructed the UI of the application to break down the process into the following components: patient biographical information, symptoms, physical exam, treatment, and final report. We also included an interface for administrators to approve and deny certain requests. The approval or denial of each request is logged.

Authorization levels - By incorporating Apache Shiro, we were able to successfully enforce authorization levels in the application. The three main levels are: administrator, doctor, and registered nurse. When a user is logged in, the role associated with the username is applied.

Features of the application are blocked by disabling their buttons if the user does not have permission to use them.

Administrative functions - One of the authorization levels in the application is administrator. VermaMS successfully assigns these users with the permission to approve requests to delete records after one has been made.

Notifications - We were not able to fully meet this goal. While a doctor has the ability to send a patient an email to remind him or her of an upcoming appointment, this process is not automated. Also, there is no appointment reminder system in place for physicians.

Generating Reports and Saving Data - VermaMS, throughout the patient interview process, builds a final report that contains all of the information recorded during the entire visit. Whenever a visit is concluded, a physician can generate either an XML or PDF version of this report and print it out for record-keeping purposes or to give to the patient upon request.

Retrieve Data - All data about current or past visits is displayed as it is requested by physicians. A physician can look at a patient's previous records, or go back to a previous section of a visit that is in progress to see information that has been previously entered.

Enter Data - We successfully fulfilled this requirement by incorporating input fields into

VermaMS. When that data is entered, it is saved to the backend database.

8.1.2 Non-Functional

We successfully achieved our non-functional requirement goals by following best security practices, implementing a back-end database, and designing our application to be efficient and built upon with future work. Using a well-tested password hashing algorithm and following Ghana's Data Protection Act helped achieve our goals of security and conforming to legal standards. Implementing our application with an Oracle database back-end helped achieve our goals of data integrity and scalability. The way we designed the application helped us achieve our goals of reliability, extensibility, usability, performance, and supportability.

Security - We achieved our goal of making VermaMS secure by following best security practices. Whenever a user creates or changes his or her password, we require that it be at least **8** characters with at least one of the following: one capital letter, one lowercase letter, one number, and one symbol. When a user signs in, he or she only has five attempts to log in before the account associated with the username is locked and must be unlocked by an administrator. We store passwords only in hashed form using the PBKDF2 function with SHA-512. The database credentials are stored in a secure .bat file that is used by the application on startup, all default accounts have been removed, all queues are passed as parameterized queries, and all connections are encrypted. The three types of users of the application (doctor, administrator, and registered nurse) are given specific permissions to do only the tasks each one needs to do.

Reliability - We achieved our goal of reliability. During our thorough testing of the application, it ran without crashes or fatal errors.

Data Integrity - We achieved our goal of ensuring data integrity. In order to ensure data integrity, the database is protected as mentioned earlier. Also, the database is stored on a remote server, and it is also backed up to the cloud using Amazon AWS. If the remote server hosting the database were to go down or be compromised in any way, there is a backup on the cloud that can be restored.

Scalability - We achieved a limited goal of scalability. We tested it to ensure that multiple users can use the application at the same time without experiencing system problems.

Extensibility - We achieved our goal of scalability. All classes that are similar have been abstracted into base classes and subclasses (for example, each specific type of test inherits form an abstract test class).

Usability - We achieved our goal of ensuring that VermaMS is usable. In order to accomplish this goal, we worked closely with Dr. Mark Weber and designed our interface according to his suggestions. By doing this, we have ensured that doctors will be able to easily adjust to using our application, since how our application is laid out will match their typical workflow. The interface is simple and easy to understand.

Performance - We were able to successfully make VermaMS perform well. The application preloads information whenever possible (i.e. it will load all types of tests as soon as it is opened so they do not have to be fetched later). Also, the application listens for user input to fields that are saved to the database, such as a patient's weight. As soon as this information is entered, it is saved to the database. By doing this, the application will not have to load all information from a visit at once when the visit is completed.

Supportability - We successfully ensured that VermaMS is supportable. Its simple interface allows system maintainers to easily navigate it and to make sure that it is functioning correctly. Due to its structure, it is also easy for developers that build upon the application to write test cases for those new features.

Legal - We have successfully incorporate all legal requirements into VermaMS. Data integrity and security is ensured as discussed earlier. Also, the application allows a doctor to print out or send patient records in PDF or XML formats to a patient upon request. Also, in order to delete patient records, a doctor must first make a request. This request then goes to an administrator who has ultimate authority in deciding whether or not to delete that patient's records from the system. By having this request system in place, this ensures that records have been kept for the time specified by the Ghana medical data laws.

9. Future Work

Our final application is capable of performing essential tasks as required by any complete EHR system on the market. It allows users to register and examine patients, generate reports on any patient examination records, and perform minor data analytics. Although our application achieves fundamental functionality, it is only a small portion of what a full EHR system is capable of. In order for VermaMS to succeed as a full EHR system, additional functionality is needed. Such additional functionality includes: a wider selection of graphs and parameters for the data analytics, the ability to monitor any disease entered into the system, and a web portal to recommend patients to the nearest drugstore to obtain their medication.

9.1 Heat Map

Currently, VermaMS provides the option to display data as a pie chart. This is useful to hospitals attempting to track different information, such as which disease is being diagnosed the most within the vicinity of the hospital. This data can also be useful for central public health agencies. Another visualization that would be very useful for VermaMS users is a heat map, which could display the number of cases of a disease in an area of Ghana. Figure 9 demonstrates how a heat map could help with concentrating disease eradication efforts by showing where occurrences are most concentrated.
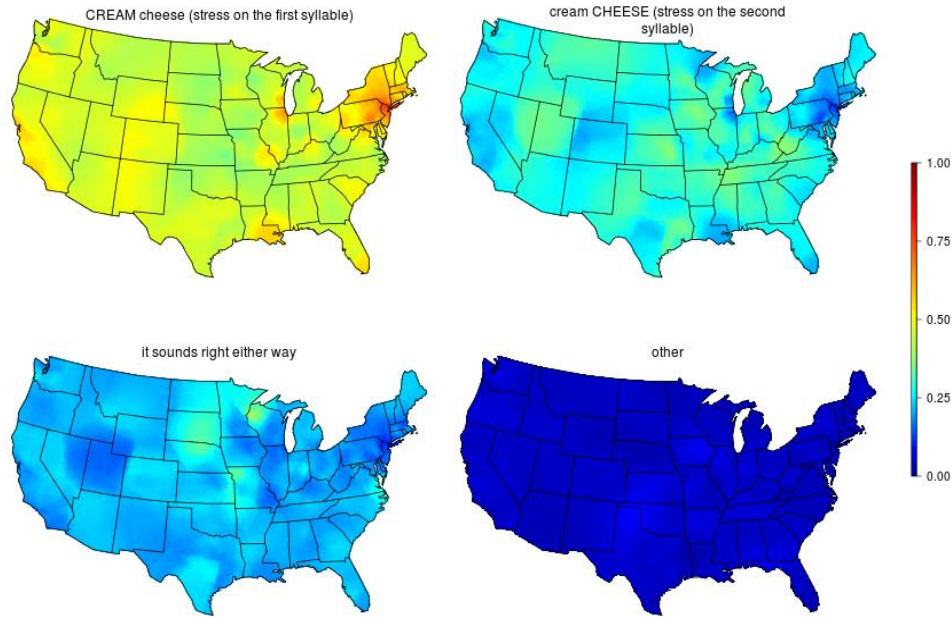
Figure 58 - A heat map displays a geographical area with different heat signatures on a color scale depending on the portion of a population that fits some criterion. The more red the color is, the more people fit the criterion, and the closer to blue, the less people fit the criterion (Katz).

9.2 Patient Interaction

VermaMS currently allows for doctors to send their patients generic emails. This functionality could be expanded to send automatic reminders to patients about appointment times and reminders to take their medication. This functionality could also be expanded to include a full-fledged inbox system for each doctor, so that there is one place within the application where doctors can keep all work-related emails. The inbox system would facilitate physician-patient interaction and also communication between hospital employees.

9.3 Extensibility

VermaMS was designed with extensibility in mind. The application can be expanded upon and

169

new features can be implemented using the present APIs. Four big features that could be added

are: the ability to add diseases to the system, an appointment scheduler, pharmaceutical stock

monitoring, and automated patient billing. The application is not a full-fledged EHR, but it can

become one with future additions.

Adapting the design of the application could allow for VermaMS to track more diseases besides

meningitis, malaria, and tuberculosis, such as cancer and diabetes. The current design allows for

medical staff to enter information about other diseases, but it must be done using free-text forms

rather than selecting from predefined lists. Medical facilities can add diseases to the database and

also their diagnostic tests and treatments. Those facilities can then track relevant information on

those diseases.

Another addition that could be added to the application is a billing system. Each billing statement

can be automated to be sent out to a patient via email as soon as a visit is finished. The billing

portion of the application could be accessed via the dashboard by members of the hospital billing

team. The staff member could then see a list of visits and their associated costs, and see if the bill

has been paid. This could allow for hospitals to easily send bills to patients and to keep a record

of transactions.

In Ghana, each medical clinic has its own pharmacy. When prescribing medication, it is

important to know how much of it is in stock. With a stock monitoring feature, a doctor can

check how much medication is in stock before prescribing it. A pharmacy staff member can

manually place orders through the application if supply is getting low, or the ordering system can

be automated to place orders if the supply reaches a certain threshold.

VermaMS does not currently have a way to schedule appointments with patients. With a future

addition, a receptionist could log in to the application and be presented with the option to

schedule an appointment. A list of the doctor's openings would appear for easier scheduling.

After the time is booked, the doctor would be notified and an entry would appear in his calendar

in the application, and the patient would be emailed with a confirmation to remind him or her of

the date and time of the appointment.

9.4 Drug Inventory Management

Given our initial finding on the overwhelming pressure on certain drugs, we thought it would be

convenient for hospitals with or without internal pharmacies to be able to recommend the closest

drugstore with stock of the desired drug. To achieve this, we envision a web portal through

which pharmacies could regularly update their stock figures. If implemented effectively, this

could grow into national database of drug availability, which could be crucial in some

circumstances.

10. Conclusion

An application for tracking the three most common diseases in Ghana (malaria, meningitis, and tuberculosis) was successfully created, reaching all of the goals and requirements set for the project. Through background research, we found common features among successful EHRs and incorporated them. Our application was implemented using the Kanban methodology, and it was designed with extensibility in mind. Although the application is not a full-fledged EHR system, it is in a state where it could be used by medical facilities in Ghana to help reduce reliance on paper records and help disease prevention efforts.

11. References

A CALL TO ACTION: Safeguard Drug Administration Within 2 Years! (2002, November). Retrieved December 13, 2016, from https://www.ismp.org/Newsletters/acutecare/articles/WhitepaperBarCodding.asp

A Medium Corporation. (2016, September 19). Flatiron Health EMR Product Case Study – garyyauchan – Medium [After visiting a Bronx hospital, speaking with staffs, and using an EHR myself, I learned the frustration end-users have to deal with on a daily basis. Their workplace software is riddled with…]. Retrieved April 4, 2017, from https://medium.com/garyyauchan/flatiron-health-emr-product-case-study-edd85049d19

Achampong, E. K. (2012, July 30). Electronic Health Record System: A Survey in Ghanaian Hospitals. Retrieved December 12, 2016, from https://www.omicsonline.org/scientific-reports/2157-7420-SR164.pdf

Acquah - Swanzy, M. (2015, May). The Faculty of Health Sciences Evaluating Electronic ... Retrieved December 15, 2016, from http://munin.uit.no/bitstream/handle/10037/8080/thesis.pdf;sequence=2

Adu, B. (2013, February 07). Accra hospitals reeling from power and water shortages ... Retrieved December 15, 2016, from http://edition.myjoyonline.com/pages/news/201302/101106.php

Affairs, A. S. (2013, September 06). Task Analysis. Retrieved May 03, 2017, from https://www.usability.gov/how-to-and-tools/methods/task-analysis.html

Agile & Waterfall Methodologies – A Side-By-Side Comparison. (n.d.). Retrieved May 18, 2017, from http://www.base36.com/2012/12/agile-waterfall-methodologies-a-side-by-side-comparison/

Agile vs. Waterfall Methods: Which Should You Choose? (2012, July 12). Retrieved May 18, 2017, from http://www.brighthubpm.com/agile/50473-agile-vs-waterfall-is-there-a-real-winner/

Allscripts. (n.d.). Allscripts EMR, Medical transcript, practices, specialties and medical centers [Allscripts centered medical transcription services that can help you scale down on costs. We provide transcribing solutions that are the perfect fit for your practice needs. Call 1888 571

9069]. Retrieved April 4, 2017, from http://www.isourcetranscription.com/EMR-EHR-Intergration-solutions/the-perfect-script-for-your-practice!.html

Al-Shorbaji, N. (2008). E-health in the Eastern Mediterranean Region: A decade of challenges and achievements. Retrieved December 15, 2016, from http://apps.who.int/iris/bitstream/10665/117599/1/14_s1_s157.pdf

Antibiotics for Bacterial Meningitis. (2014, November 14). Retrieved December 11, 2016, from http://www.webmd.com/brain/antibiotics-for-bacterial-meningitis

Atlassian (2012) *What is version control: Centralized vs. DVCS - Atlassian Blogs*. Available at: http://blogs.atlassian.com/2012/02/version-control-centralized-dvcs/ (Accessed: 18 December 2016).

Atlassian (2016) *The agile coach*. Available at: https://www.atlassian.com/agile/kanban (Accessed: 18 December 2016).

Authentication [Def. 1]. (2017). *SANS Glossary of Security Terms*. Retrieved April 26, 2017, from https://www.sans.org/security-resources/glossary-of-terms/

Aveda, S. (2015, August 25). What is the importance of a database management system ... Retrieved May 16, 2017, from https://www.linkedin.com/pulse/what-importance-database-management-system-scott-aveda

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved May 17, 2017, from http://agilemanifesto.org/

Bedeley, R. T., & Palvia, P. (2014). A Study of the Issues of E-Health Care in Developing Countries: The Case of Ghana. Retrieved December 15, 2016, from http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1112&context=amcis2014

Cao, J. (n.d.). User Analysis Before Diving Into Design (Part 1) - UXPin. Retrieved May 3, 2017, from https://www.uxpin.com/studio/blog/user-analysis-diving-design-part-1/

CDC. (2016, March 21). TB terms. Retrieved December 16, 2016, from Centers for Disease Control and Prevention, https://www.cdc.gov/tb/topic/basics/glossary.htm

CDC. (2016, September 1). Testing for tuberculosis (TB). Retrieved December 16, 2016, from

Centers for Disease Control and Prevention,
http://www.cdc.gov/tb/publications/factsheets/testing/tb_testing.htm

Cerner. (n.d.). Cerner | Maternity - EMR-Matrix.org. Retrieved April 4, 2017, from http://emr-matrix.org/2010/11/cerner-powerchart-maternity-cpoe-ob-triageldpostpartumnewborn/

(2003, June). Chapter 3 Threat Modeling. *Microsoft Developer Network.* Retrieved April 18, 2017, from https://msdn.microsoft.com/en-us/library/ff648644.aspx.

Cohn, M. (n.d.). The Daily Scrum Meeting. Retrieved May 18, 2017, from https://www.mountaingoatsoftware.com/agile/scrum/meetings/daily-scrum

DB-Engines Ranking. (2016, November). Retrieved November 28, 2016, from http://db-engines.com/en/ranking

DesRoches, C. M., Campbell, E. G., Rao, S. R., Donelan, K., Ferris, T. G., Jha, A., . . . Blumenthal, D. (2008, July 03). Electronic Health Records in Ambulatory Care - A National Survey of Physicians — NEJM. Retrieved December 15, 2016, from http://www.nejm.org/doi/full/10.1056/NEJMsa0802005#t=article

DocuTAP Implements Electronic Medical Records System for Sanford World Clinics in Ghana. (2012, August 26). Retrieved December 16, 2016, from DocuTAP, https://docutap.com/blog/docutap-implements-electronic-medical-records-system-for-sanford-world-clin

DocuTAP. (2016, September 07). Retrieved December 15, 2016, from https://www.youtube.com/watch?v=pfD9YM78ITs

Ducklin, P. (2013, November 20). Serious Security: How to store your users' passwords safely. Retrieved April 24, 2017 from https://nakedsecurity.sophos.com/2013/11/20/serious-security-how-to-store-your-users-passwords-safely/

Durrani, H., & Khoja, S. (2009). A systematic review of the use of telehealth in Asian countries. Retrieved December 15, 2016, from https://www.ncbi.nlm.nih.gov/pubmedhealth/PMH0027296/

Eliminate Handwritten Prescriptions Within 3 Years. (2000). Retrieved December 13, 2016, from https://www.ismp.org/Newsletters/acutecare/articles/Whitepaper.asp

EMR: The Progress to 100% Electronic Medical Records. (2015, January 30). Retrieved May 11,

2017, from http://elearning.scranton.edu/resource/health-human-services/emr_the-progress-to-100-percent-electronic-medical-records

Eriksson, U. (2012, April 5). Functional Requirements vs Non Functional Requirements. Retrieved December 17, 2016, from http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

Erstad, T. L. (2003). Analyzing Computer based Patient Records: A Review of Literature. Retrieved December 13, 2016, from http://www.providersedge.com/ehdocs/ehr_articles/Analyzing_CPRs-A_Review_of_Literature.pdf

Ewing, T., & Cusick, D. (2004). Knowing what to measure. *Healthcare financial management: Journal of the Healthcare Financial Management Association, 58*(6), 60 - 63

Garrett, P., & Seidman, J. (2011, August 26). EMR vs EHR – What is the Difference? Retrieved May 11, 2017, from https://www.healthit.gov/buzz-blog/electronic-health-and-medical-records/emr-vs-ehr-difference/

Gegick, M., & Barnum, S. (2013, May 10). Least Privilege. Retrieved December 17, 2016, from https://www.us-cert.gov/bsi/articles/knowledge/principles/least-privilege

Ghana adopts electronic health records for patients. (2014, February 09). Retrieved December 12, 2016, from http://www.ghananewsagency.org/health/ghana-adopts-electronic-health-records-for-patients-70537

Ghana: Tuberculosis Profile. (2016, December). Retrieved from https://extranet.who.int/sree/Reports?op=Replet&name=/WHO_HQ_Reports/G2/PROD/EXT/TBCountryProfile&ISO2=GH&outtype=pdf

Ghana: WHO statistical profile. (2015, January). Retrieved November 28, 2016, from http://www.who.int/gho/countries/gha.pdf?ua=1

GitHub (2016) *Build software better, together*. Available at: http://github.com (Accessed: 18 December 2016).

Guidelines for Treatment of Malaria in the United States (2013, July 01). Retrieved November 28, 2016, from https://www.cdc.gov/malaria/resources/pdf/treatmenttable.pdf

Gupta, S. (2014) *Key discussions (a-k-a meetings) that happen in AGILE – Scrum.* Available at: http://www.quotium.com/performance/key-discussions-a-k-a-meetings-that-happen-in-agile-scrum/ (Accessed: 18 December 2016).

Highly contagious meningitis outbreaks continue in African countries. (n.d.). Retrieved December 11, 2016, from http://www.afro.who.int/en/media-centre/afro-feature/item/7579-highly-contagious-meningitis-outbreaks-continue-in-african-countries.html

Hillsberg, A. (2015) *Wrike tour: An overview of features, pricing and awards.* Available at: https://financesonline.com/wrike-tour-overview-of-features-pricing-awards/ (Accessed: 18 December 2016).

HIV/AIDS CDC in Ghana. (2013, August). Retrieved December 11, 2016, from http://www.cdc.gov/globalhealth/countries/ghana/pdf/ghana-2013.pdf

How TB spreads. (2016, July 26). Retrieved December 16, 2016, from https://www.cdc.gov/tb/topic/basics/howtbspreads.htm

Hussung, T. (2016, March 10). What is the software development cycle? Retrieved from http://online.husson.edu/software-development-cycle/

Introduction: As medical care gets more and - nasbhc.org. (2009, August 05). Retrieved December 12, 2016, from http://www.nasbhc.org/atf/cf/{CD9949F2-2761-42FB-BC7A-CEE165C701D9}/TA_HIT_history of EMR.pdf

Jeloka, S., D. Gosselin, R. Smith, G. Mulagund, N. Lewis, J. Narasinghanallur, S. Tata, and N. Manappa. (2012, July). 2. Security Checklists and Recommendations. Retrieved April 18, 2017 from https://docs.oracle.com/cd/B19306_01/network.102/b14266/checklis.htm#CHDBEJBJ.

Just what is 'A big database'? (2009, March 15). Retrieved November 28, 2016, from http://stackoverflow.com/questions/647210/just-what-is-a-big-database

Justin (2013) *What is the software development life cycle (SDLC)?* Available at: https://airbrake.io/blog/sdlc/what-is-the-software-development-life-cycle (Accessed: 18 December 2016).

Katz, J. (n.d.). Beyond "Soda, Pop, or Coke": Regional Dialect Variation in the Continental US [Digital image]. Retrieved May 8, 2017, from http://www4.ncsu.edu/~jakatz2/project-dialect.html

Khalid, A. (2016) *The pros and cons of Facebook messenger and WhatsApp*. Available at: http://www.dailydot.com/debug/facebook-messenger-vs-whatsapp/ (Accessed: 18 December 2016).

Lardinois, F. (2014, March 13). Google Drive Gets A Big Price Drop, 100GB Now Costs $1.99 A Month. Retrieved May 18, 2017, from https://techcrunch.com/2014/03/13/google-drive-gets-a-big-price-drop-100gb-now-costs-1-99-a-month/

Mack, J. (2014, May 28). Five Advantages & Disadvantages Of MySQL. Retrieved November 28, 2016, from https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/

Malaria Diagnosis (United States). (2015). Retrieved November 28, 2016, from https://www.cdc.gov/malaria/diagnosis_treatment/diagnosis.html

Malaria Information and Prophylaxis, by Country [G]. (2016, September 30). Retrieved May 18, 2017, from https://www.cdc.gov/malaria/travelers/country_table/g.html#four

Manuswath, K. (2015, February 11). Problem oriented medical record. Retrieved December 17, 2016, from SlideShare, http://www.slideshare.net/manuswath/problem-oriented-medical-record

Menachemi, N., & Brooks, R. G. (2006, June). Reviewing the benefits and costs of electronic health records and associated patient safety technologies. Retrieved December 15, 2016, from http://www.ncbi.nlm.nih.gov/pubmed/16848129

Meningococcal Disease in Other Countries. (2016, October 13). Retrieved December 12, 2016, from http://www.cdc.gov/meningococcal/global.html

Mohindra, D. (2017, January 5). MET12-J. Do not use finalizers. W. Snavely (Ed.), Retrieved April 18, 2017 from https://www.securecoding.cert.org/confluence/display/java/MET12-J.+Do+not+use+finalizers

Mohindra, D. (2017, January 5). OBJ11-J. Be wary of letting constructors throw exceptions (W. Snavely, Ed.). Retrieved April 18, 2017, from https://www.securecoding.cert.org/confluence/display/java/OBJ11-J. Be wary of letting constructors throw exceptions

M. (2013, March 2). Passwords, Hashing, and Salt. Retrieved April 19, 2017, from http://goodmath.scientopia.org/2013/03/02/passwords-hashing-and-salt/

Mulcahy, R. (2007) *Business intelligence definition and solutions*. Available at:
http://www.cio.com/article/2439504/business-intelligence/business-intelligence-definition-and-
solutions.html (Accessed: 18 December 2016).

(2014, April 4). The NetBeans IDE: Pros and Cons. Retrieved May 18, 2017, from
http://infotech101.com/the-netbeans-ide-pros-cons/

Non-Repudiation [Def. 1]. (2017). *SANS Glossary of Security Terms*. Retrieved April 26, 2017,
from https://www.sans.org/security-resources/glossary-of-terms/

Northcutt, S. (2017). The Attack Surface Problem. Retrieved May 22, 2017, from
https://www.sans.edu/cyber-research/security-laboratory/article/did-attack-surface

(2015, March 8). OWASP. Application Threat Modeling. Retrieved April 18, 2017, from
http://www.owasp.org/index.php/Application_Threat_Modeling

Patkar, M., Price, D., & Lee, J. (2014, September 11). The Effects Power Outages Can Have On
Your Computer. Retrieved May 11, 2017, from http://www.makeuseof.com/tag/effects-power-
outages-can-computer/

PCMag (2016) *Trello*. Available at: http://www.pcmag.com/business/directory/project-
management/951-trello (Accessed: 18 December 2016).

Pressman, R. S. (n.d.). Chapter 6 – Requirements Modeling: Scenarios and Information. In
Software Engineering: A Practitioner's Approach (7th ed.). McGraw-Hill Education.

Plutis, D. (2016) *GroupMe vs WhatsApp | messaging app comparison*. Available at:
https://versus.com/en/groupme-vs-whatsapp (Accessed: 18 December 2016).

Rindfleisch, T. C. (1997). Privacy, information technology, and health care. *Communications of
the Association for Computing Machinery, 40*(8), 92 - 100

SDLC - overview. (2016). Retrieved December 18, 2016, from TutorialsPoint,
https://www.tutorialspoint.com/sdlc/sdlc_overview.htm

Security of Computerized Patient Record. (n.d.). Retrieved May 11, 2017, from
http://seeri.etsu.edu/SECodeCases/ethicsC/Computerized%20Patient%20Records.htm

Shelly, G.B., B, G., Rosenblatt, H.J. and Cashman, T.J. (2009) *Systems analysis and design*. 8th
edn. Boston, MA: Thomson Course Technology.

Shiotsu, Y. (2017, January 15). Agile vs. Waterfall. Retrieved May 18, 2017, from https://www.upwork.com/hiring/development/agile-vs-waterfall/

Skype (2016) *Free calls to friends and family*. Available at: http://skype.com (Accessed: 18 December 2016).

Slack (n.d.) *Slack: Be less busy*. Available at: http://slack.com (Accessed: 18 December 2016).

Smith, P. (2005, February 2). Missing clinical information during primary care visits. Retrieved December 13, 2016, from https://www.ncbi.nlm.nih.gov/pubmed/15687311

Sood, S. P., Nwabueze, S. N., Mbarika, V. W., Prakash, N., Chatterjee, S., Ray, P., & Mishra, S. (2008). Electronic Medical Records: A Review Comparing the Challenges in Developed and Developing Countries. Retrieved December 15, 2016, from https://www.computer.org/csdl/proceedings/hicss/2008/3075/00/30750248.pdf

SQL Injection Prevention Cheat Sheet. (2016, August 7). In *OWASP*. Retrieved April 19, 2017, from https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

Stansfield, J. (2014, March 13). Microsoft SQL Server vs. Oracle: The Same, But Different? Retrieved November 28, 2016, from http://www.seguetech.com/microsoft-sql-server-vs-oracle-same-different/

Stuchfield, B. M., Jagilly, R., & Tulloh, B. R. (2007, November). Second opinions in remote surgical practice using email and digital photography. Retrieved December 15, 2016, from http://www.ncbi.nlm.nih.gov/pubmed/17931268

T.B. cases increasing (2016, March 22). *The Ghanaian Times*. Retrieved from http://www.ghanaiantimes.com.gh/t-b-cases-increasing/

Tee, J. (2012, July). What's the big IDE? Comparing Eclipse and NetBeans. Retrieved May 22, 2017 from http://www.theserverside.com/feature/Whats-the-Big-IDE-Comparing-Eclipse-vs-NetBeans

Teeling, M. (2012, May 14). What is Data Integrity? Learn How to Ensure Database Data Integrity via Checks, Tests, & Best Practices. Retrieved December 17, 2016, from https://www.veracode.com/blog/2012/05/what-is-data-integrity

Teviu, E., Aikins, M., Abdulai, T. I., Sackey, S., Boni, P., Afari, E., & Wurapa, F. (2012,

September). Improving Medical Records Filing in a Municipal Hospital in Ghana. Retrieved December 13, 2016, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3645163/

The Casebooks Project. (2015, February 22). Retrieved May 11, 2017, from http://www.magicandmedicine.hps.cam.ac.uk/on-astrological-medicine/further-reading/history-of-medical-record-keeping

Theobald, M. (2014, July 18). How Meningitis Is Diagnosed in Its Early Stages. Retrieved December 11, 2016, from http://www.everydayhealth.com/hs/understanding-meningitis/how-meningitis-is-diagnosed-in-its-early-stages/

Theobald, M. (2014, July 18). Understanding the 5 Types of Meningitis. Retrieved December 11, 2016, from http://www.everydayhealth.com/hs/understanding-meningitis/types-of-meningitis/

(2005). The STRIDE Threat Model. *Microsoft Developer Network.* Retrieved April 18, 2017, https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx

Towards eHealth 2.0 in Ghana: A programme and opportunities for private and public ICT initiatives. (2014, April 30). Retrieved December 15, 2016, from IICD, http://www.rvo.nl/sites/default/files/2014/06/IICD%20Rapport%20Towards%20eHealth%202%200%20in%20Ghana.pdf

Tuberculosis (TB). (2016, November 15). Retrieved December 15, 2016, from NHS, http://www.nhs.uk/conditions/tuberculosis/pages/introduction.aspx#

Valentine, C. (2012) *A Comparison of System Development Methodologies and the Affect Corporate Culture has on Choice of Methodology*. Available at: http://cevsdc.blogspot.com/ (Accessed: 18 December 2016).

Varga, J. (2011). Managing Paper Patient Records in a Clinical Practice. Retrieved December 16, 2016, from http://www.nuance.com/ucmprod/groups/imaging/@web-enus/documents/collateral/nc_019331.pdf

Viswanathan, P. (2017, February 23). Comparing Java IDEs: Eclipse vs. NetBeans vs. IntelliJ. Retrieved May 18, 2017, from https://www.lifewire.com/comparing-java-ides-eclipse-vs-netbeans-vs-intellij-2373152

What are some advantages and disadvantages of Oracle database? (n.d.). Retrieved November 28, 2016, from https://www.reference.com/technology/advantages-disadvantages-oracle-

database-dbec0904f3b40425

What are the Software Development Life Cycle (SDLC) phases? (n.d.) Available at: http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/ (Accessed: 18 December 2016).

What is agile? What is Scrum? (2013) Available at: https://www.cprime.com/resources/what-is-agile-what-is-scrum/ (Accessed: 18 December 2016).

WHO | Meningitis vaccine provides hope to people in Ghana. (2012, November). Retrieved December 12, 2016, from http://www.who.int/features/2012/meningitis_ghana/en/

WHO. (2014, November 20). EHealth. Retrieved December 17, 2016, from World Health Organization, http://www.who.int/topics/ehealth/en/

Yellow Fever & Malaria Information, by Country. (2015). Retrieved November 28, 2016, from http://wwwnc.cdc.gov/travel/yellowbook/2016/infectious-diseases-related-to-travel/yellow-fever-malaria-information-by-country/Ghana