SafeMeds

by

Samuel Curry and Vriti Patel

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Samuel Curry
Samuel Curry – Project Manager/Full Stack Developer — Date: 4/13/2020

Vriti Patel
Vriti Patel – Software Developer/QA — Date: 4/13/2020

Yahya Gilany
Technical Advisor — Date: 4/13/2020

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2020

~~~~~~

Prepared by:

Samuel Curry, Project Manager and Developer
Vriti Patel, Software Developer

Students of
University of Cincinnati
College of Education, Criminal Justice, and Human Services
School of Information Technology

April 2020
~~~~~~

TABLE OF CONTENTS

List of Illustrations

TABLES

FIGURES

ABSTRACT

According to NIH (National Institute of Drug Abuse), an estimated 52 million people have used prescription drugs for nonmedical reasons at least once in their lifetimes. There are many applications that allow healthcare organizations to view patient's history, but they lack the flexibility of allowing other organizations to view the same information. SafeMeds is a cloud-based web solution that allows healthcare providers to be connected, preventing patients going from office-to-office to acquire prescriptions. Our product allows pharmacists and their prescriber to be alerted to patient activity. SafeMeds allows pharmacists to view the prescription sent over by doctors or prescribers, while doctors can view the history and prescribing prescriptions page. SafeMeds now makes it easy for prescribers and pharmacists to view trends and identify abuses in prescriptions. Prescription drugs can be regulated and the use of medications for nonmedical reasons can decline gradually using SafeMeds.

PROBLEM STATEMENT

Introduction

With all of the technology advancements in the medical field, communication and monitoring medications is crucial between healthcare providers. Family care physicians and pharmacies can be in contact with one another for certain patients, but it does not mean that a person can go to several different doctors and pharmacies to get their medicine. Sometimes, the lack of communication can lead to unintentional sales of multiple medications. While pharmacy chains like Walgreens being in contact with other Walgreens or medical professionals are able to send over prescriptions, we believe there is a lack of communication on a national level. Our goal will be to create a cloud based web solution that allows medical professionals to keep track of medications sold to patients and be accountable for any abuse.

Project Description

SafeMeds will be an application that allows pharmacies and medical offices to communicate. Doctors and pharmacists will be signed up and each medical professional will have a role selected. For instance, a doctor will be given the role of a doctor and pharmacist will be given the role of a pharmacist. This will give each user a separate authorization page. A doctor will be able to enter patients, search for patients, see past prescription history and even prescribe medications. A pharmacist will be able to search up customers and dispense the medication. However, customers may be denied a sale of medication if they came in too recently for a refill. Pharmacists can also directly contact the prescribing doctor and resolve the issue or report any abuse. Our application will show a dashboard that shows which drug has been prescribed the most or which drug has been dispensed the most in each facility.

Problem

As medicine changes every year and becomes vital to everyone's health, people start to misuse medications and often overdose. According to Centers for Disease Control and Prevention (CDC), more than 702,000 people have died from a drug overdose between the years of 1999 and 2017 and just in 2017, more than 70,000 people died from drug overdoses which was the leading cause of injury-related death in the US. For instance, millions of people suffer from pain and to treat those conditions, opioids are most often prescribed. Of the many deaths that occurred from drug overdose, about 68% involved a prescription of illicit opioid. Based on our research, there is no application to allow healthcare organizations and pharmacies to monitor medications that are being prescribed to people on a national basis. People are able to obtain controlled substances from many organizations at a time because there is no way to keep track of who has been prescribed what on which day and when they are due for refills. There are people who misuse prescription drugs or use medications to make other drugs. Sometimes when people are asked where they got the medicine from, their response is that they bought or received it from either their friends or family members.

Solution

With an epidemic of opioid abuse, a solution is needed to help safeguard the distribution of these addictive drugs. By offering a cloud-based web solution, SafeMeds connects healthcare organizations from the primary care physician all the way down to the pharmacy/pharmacist. Through a browser-based solution, SafeMeds allows healthcare organizations across the board to not only prescribe medications, but also see a history of what has been prescribed, where it was prescribed, and by who. Not only does this prevent patients from going from office-to-office getting prescriptions by not allowing the prescription, but it allows healthcare professionals to be connected, and if there are any possible reactions between two drugs prescribed, the system will automatically warn the pharmacist and the prescriber. If there are any questions around the prescription, the pharmacy will be able to have direct contact with the prescriber and the history of prescriptions. This means that

prescribers will be held accountable, and if there is any abuse, this can be directly reported to the appropriate organization with the data reporting to support it. Through building an open API, we will be able to provide integrations with other healthcare software, such as EPIC, that is HIPPA compliant.

While hospitals often times will use software like Epic that lets patients be shared from hospital-to-hospital, there is a lack of something in the market that brings in the pharmacist and other healthcare providers. These solutions often provide a way for healthcare providers to manage care in that they act as databases for vitals, notes, and prescriptions, but this is often lackluster, and basic. Through a software-as-a-service model, SafeMeds will provide a modern interface for prescribing medicine. Furthermore, there is nothing on the market that will allow the analysis of prescribing trends across hospital networks, which would allow a targeted approach to solving epidemics. Once this solution is built, it could one day expand into an app for patients to manage their prescription schedules, and have a quick way to ask questions to their pharmacists.

USER PROFILE

Table 1. User Profiles 1 shows a user profile for pharmacists and pharmacy technicians. The users should be familiar with basic skills of logging in, navigating to searching a patient, opening up a patients prescription, dispensing medications and even opening up a patient's prescription history. Our application will allow health care professionals to be in contact with one another, view medication/prescription history, and manage the use of certain drugs. Pharmacists will only be able to view certain pages as their role will limit the application access. The application will show trends of which drug is being prescribed the most by the pharmacist and pharmacy itself.

Potential Users: · Pharmacists, pharmacy technicians
Software and Interface Experience: · All users should understand the basic skills about using software from logging in and performing basic tasks. · Pharmacists and pharmacy technicians will have the skills to look up customers and dispense the correct medications.
Experience with Similar Applications: · Pharmacists and pharmacy technicians will have experience with QS/1 which is a pharmacy management system that allows users to view all prescription information such as respective doctor, drugs and quantities prescribed.

Task Experience: · Pharmacists will be able to contact doctors at ease in case of asking further questions about a specific customers prescription and prescription history. · Trends will be shown as to what drug is sold most with each pharmacy · Patient prescription history can be retrieved
Frequency of Use: · This application will be used on a daily basis to prescribe and dispense medications. · Doctors in hospitals will use 24/7 · Doctors at clinics are open around 8am-7pm (depends on location and doctor).
Key Interface Design Requirements that the Profile Suggests: · Simple and intuitive UI · Easily accessible navigation tabs · Visually and aesthetically pleasing designs

Table 1. User Profile 1

Table 2. User Profiles 2 shows a user profile for doctors and nurses. The users should be familiar with basic skills of logging in, navigating to searching a patient, creating a profile for patients, and even opening up a patient's prescription history. A doctor will look up a patient and if they are not in the system, a doctor will register the patient with their details. They then can prescribe any medication along with their dosage, strength, name, etc. Our application will allow health care professionals to be in contact with one another, view medication/prescription history, and manage the use of certain drugs. Pharmacists will only be able to view certain pages as their role will limit the

application access. The application will show trends of which drug is being prescribed the most by a doctor.

Potential Users: · Doctors, nurses
Software and Interface Experience: · All users should understand the basic skills about using software from logging in and performing basic tasks of searching and saving. · Doctors and nurses will have know the basic skills of entering patient information and prescribing medications.
Experience with Similar Applications: · Doctors and nurses will understand the functionality of the application because it is similar to Epic, which is what is currently used in hospitals and health systems. They are able to access, organize, store and share electronic medical records.
Task Experience: · Doctors will be able to enter any prescriptions to their patients and be in contact with pharmacists. · Trends will be shown as to what drug is sold most with each doctor or pharmacy · History of patients can be retrieved as well as prescription history.
Frequency of Use: · This application will be used on a daily basis to prescribe. · Pharmacies in hospitals will use 24/7. · Doctors at clinics are open around 8am-7pm (depends on location and doctor).

Key Interface Design Requirements that the Profile Suggests:

- Simple and intuitive UI
- Easily accessible navigation tabs
- Visually and aesthetically pleasing designs

Table 2. User Profile 2

Use Case Diagram

Figure 3. Use Case Diagram presents the use case diagram. This represents the level of requirements for our cloud-based web solution.

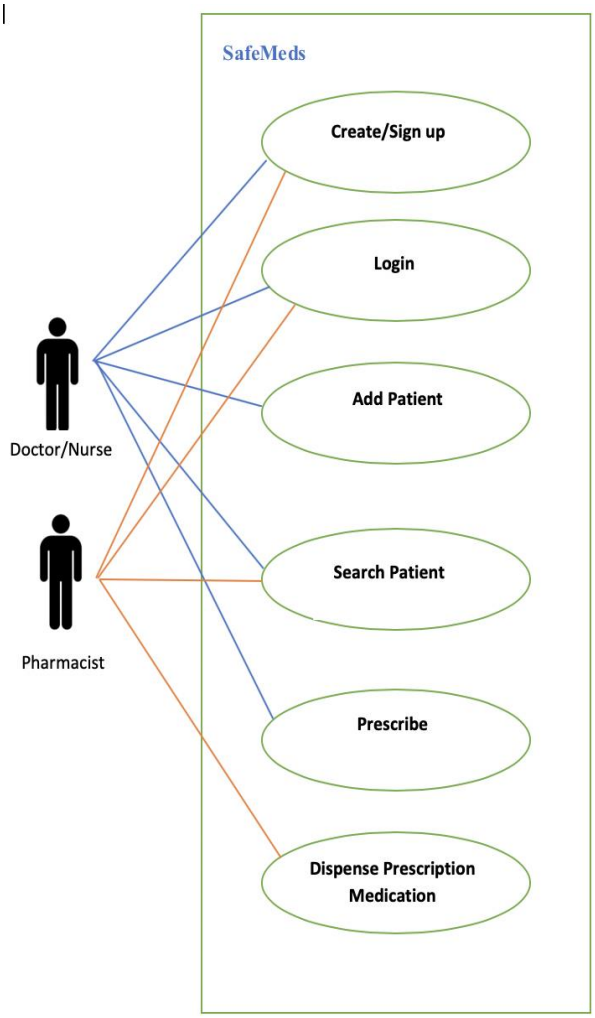

Figure 3. Use Case Diagram

PROJECT MANAGEMENT

Budget

Figure 4: Project Budget illustrates the projected real-world cost for the development of the SafeMeds application.

Project Asset Type							**Funding Source (if applicable)**				
Compliance/Regulatory		Comments: SafeMeds is a software designed to keep healthcare professionals in compliance by limiting how much of certain substances can be prescribed in a given interval. And it provides accountability by providing trend analytics.					Self		Comments: Chipped in money for azure credits to host the project fully on azure		
Risk Identification (See Risk Types tab)							**Project Stakeholder(s)**				
	Risk Rating 1-5 (5 is high)*	*Comments*	*Weight*	*Score*			New product, with no stakeholders at the moment besides ourselves.				
Work Effort (days)	3		40%	1.20							
Complexity	2		60%	1.20							
Project Risk Score:				**2.40**							
Estimate of Benefits											
If project will generate revenue, estimate 1 year here:		$ 25,000.00									
Select other benefits the project may bring a customer or user:											
Risk Avoidance		☑									
Improved customer satisfaction		☐									
Increased system availability		☐									
Productivity or process improvement		☑									
Reduced costs		☐									
Estimated Cost Rough Order of Magnitude:											
	Rate Per/Hr	**Work Effort (Hours)**	**1 X Costs**		**Ongoing Annual**			**Comments: Hardware costs are about $3,000 per year with Azure. This would obviously be annual, and would increase as our user base increases, as scalability will come along with a larger user base.**			
					Rate Per/Hr	**Work Effort (Hours)**	**1 X Support Cost**				
Labor - IT	20	800	$ 16,000.00		20	150	$ 3,000.00				
Labor - External			$ -			0	$ -				
Software - External			$ 3,000.00								
Hardware - External											
Misc.											
TOTAL			$ 19,000.00				$ 3,000.00				
5-Year ROI Analysis											
Description		**5- Year Expected**	**Conservative (1.5)**								
Total Costs		**$ 34,000.00**	**$ 51,000.00**								
Total Benefit		**$ 125,000.00**	$62,500								
Total Costs/Benefit Differential		**$ 91,000.00**									
Conservative Costs/Benefit Differential		**$ 11,500.00**									

Figure 4. Project Budget

Objectives/Deliverables

The project deliverables and deadlines are presented in **Table 5. Project Objectives/Deliverables Due Dates**.

MAJOR PROJECT MILESTONES (DELIVERABLES)			
FALL OF 2019 MILESTONES			
Initiation Phase	9/10/19	Team Contract Written	9/23/19
Research Phase	9/23/19	Project Abstract Drafted	10/14/19
API-Authentication	10/23/19	User Profile Drafted	10/21/19
API-Patient/Medication	11/25/19	Elevator Speech	10/21/19
Design	11/30/19	Fall Presentation	12/4/19
SPRING OF 2020 MILESTONES			
Alpha Test Phase	1/20/20	Spring Presentation	4/8/20
Deployment Phase	2/10/20	Tech Expo	4/14/20
Security Testing	2/28/20		
Beta Test Phase	3/7/20		
Redesign/Software Update Phase	3/15/20		

Table 5. Project Objectives/Deliverables Due Dates

Through a web-based application, we will allow healthcare organizations to stay connected. They will be able to prescribe, view prescription histories, and also view any possible counter effects of prescriptions based on a patients' prescription history. This will not only create a more connected and more aware healthcare industry, by stopping repeated prescriptions of addictive substances, but it will always provide the analytics to stop the problems at the source. Through a browser-based dashboarding system, SafeMeds will allow hospitals to track trends in how illnesses are treated (with what pharmaceutical), and see trends across doctors to more quickly identify any abuse. Not only will this stop the problem at the source, but it will provide the data to make society more aware of the problem we have.

The features of SafeMeds will include comprehensive prescription history, showing who, when, and why things were prescribed that connects organizations and stops the prescription of drugs before the patient is due for renewal. In addition, it will include an intelligent warning system that stops prescriptions of drugs that will counteract with each other, or harm people who cannot take

specific types of drugs for other reasons, such as allergies. A dashboarding system will track prescription trends across illness, across each doctor, and point out potential abusers to address them as quickly as possible

Through all of these features, we will enable healthcare organizations to make smart decisions when it comes to prescriptions, and create a safer environment for the patient by stopping possible prescription mistakes. Through this, we create a better quality-of-life for the patient, and communities, who are afflicted by epidemics such as the opioid epidemic.

Project Schedule

Table 6. WBS Chart is our project schedule with the task name, duration and respective dates.

Task Name	Duration (Days)	Start Date	End Date
1. 0 Project Management and Deliverables	193	08/20/19	02/29/20
1.1 Team Building	7	08/20/19	08/27/19
1.2 Ideas and Brainstorming	6	08/27/19	09/02/19
1.3 Fall Semester Assignment 0: Team Members & Project Name	6	08/27/19	09/02/19
1.3.1 Project Name	6	08/27/19	09/2/19
1.3.2 Project Logo and Branding	6	08/27/19	09/02/19
1.4 Fall Semester Assignment 1: Team Contract	21	09/02/19	09/23/19
1.4.1 Project Approval	10	09/02/19	09/12/19
1.4.2 Gantt Chart	21	09/02/19	09/23/19
1.4.3 Work Breakdown Structure	21	09/02/19	09/23/19

1.5 Fall Semester Assignment 2: Project Abstract for Tech Expo	21	09/23/19	10/14/19
1.6 Fall Semester Assignment 3: Team Contract Resubmission	7	10/07/19	10/14/19
1.7 Fall Semester Assignment 4: User Profile	14	10/07/19	10/21/19
1.8 Fall Semester Assignment 5: Use Case Diagram	14	10/07/19	10/21/19
1.9 Fall Semester Assignment 6: Draft Report	14	10/21/19	11/04/19
1.10 Fall Semester Assignment 7: Final Fall Semester Report	28	11/04/19	12/02/19
1.11 Fall Semester Oral Presentation	30	11/04/19	12/04/19
1.11.1 Presentation Practice	14	11/04/19	11/18/19
1.11.2 Presentation	1	12/03/19	12/04/19
1.12 Spring Semester Assignment 1: Testing plan/report	11	02/03/20	02/10/20
1.13 Spring Semester Assignment 2: Abstract	7	02/11/20	02/17/20
1.14 Spring Semester Assignment 3: Draft Tech Expo Poster	14	02/18/20	03/02/20
1.15 Spring Semester Assignment 4: Final Poster	14	03/03/20	03/16/20
1.16 Spring Semester Presentation	8	04/01/20	04/08/20
1.16.1 Presentation Practice	8	04/01/20	04/08/20
1.16.2 Presentation	1	04/08/20	04/08/20
1.17 Spring Semester Assignment 5: Final Report	28	03/17/20	04/13/20
1.18 Spring Semester Assignment 6: Safe Assign for Final Report	13	04/01/20	04/13/20

1.19 Spring Semester Tech Expo	1	04/14/20	04/14/20
1.19.1 Tech Expo Preparation	1	04/14/20	04/14/20
1.19.2 Tech Expo Event	1	04/14/20	04/14/20
1.20 Spring Semester Assignment 7: Final Library Copy	15	04/13/20	04/27/20
2.0 Research	7	09/16/19	09/23/19
2.1 Software Requirements	7	09/16/19	09/23/19
2.1.1 Determine Front End Language	7	09/16/19	09/23/19
2.1.2 Determine Back End Language	7	09/16/19	09/23/19
2.1.3 Determine Database Environment	7	09/16/19	09/23/19
2.2 Determine Features	7	09/16/19	09/23/19
3.0 Environment Set-Up	7	09/23/19	09/30/19
3.1 Setup GitHub	7	09/23/19	09/30/19
3.2 Setup Spring Tool Suite	7	09/23/19	09/30/19
3.2.1 Install Maven	7	09/23/19	09/30/19
3.2.1 Install Tomcat	7	09/23/19	09/30/19
3.3 Setup Database Server	7	10/7/19	10/14/19
3.3.1 Instal Microsoft SQL Server	7	10/7/19	10/14/19
3.3.2 Write Database Creation Scripts	7	10/7/19	10/14/19
3.3.3 Link Database Tables	3	10/14/19	10/17/19

4.0 Development (Front End)	141	09/23/19	03/07/20
4.1.1 Create react app (with create-react-app)	1	09/23/19	09/23/19
4.1.2 Create login page	8	9/30/19	10/07/19
4.1.2.1 Create UI tests	8	10/07/19	10/15/19
4.1.3 Create register page	8	10/15/19	10/23/19
4.1.3.1 Create UI tests	5	10/23/19	10/28/19
4.1.4 Create forgot password page	4	10/28/19	11/01/19
4.1.4.1 Create UI tests	2	11/01/19	11/05/19
4.1.5 Create patient profile page	10	11/05/19	11/15/19
4.1.5.1 Create UI tests	4	11/15/19	11/19/19
4.1.6 Create patient ‘prescribe’ page	20	11/20/19	12/10/19
4.1.6.1 Create UI tests	6	01/13/20	01/19/20
4.1.7 Regression Testing	15	01/20/20	02/04/20
4.1.8 Create prescribable page	4	03/12/20	03/14/20
4.1.9 Create drug page	4	03/12/20	03/14/20
4.1.10 Create patient analytics page	5	04/1/20	04/05/20
4.1.11 Create doctor analytics page	5	04/1/20	04/05/20
4.1.12 HIPPA Compliance Analysis	10	03/04/20	03/14/20
4.1.13 HIPPA Compliance Implementation	20	03/15/20	04/07/20

5.0 Development (Back End)	91	09/30/19	04/06/20
5.1.1 Create new java spring boot starter project	7	09/30/19	10/07/19
5.1.2 Setup folder structure Java EE	7	09/30/19	10/07/19
5.1.3 Add needed dependencies in pom.xml	7	09/30/19	10/07/19
5.1.4 Create User Database Schema	2	10/15/19	10/17/19
5.1.5 Create Role Database Schema	2	10/15/19	10/17/19
5.1.6 Create Privilege Database Schema	2	10/15/19	10/17/19
5.1.7 Create UserRole Database Schema	2	10/15/19	10/17/19
5.1.8 Integrate SQL with java project	2	10/17/19	10/19/19
5.1.9 Create interface and implementation classes	2	10/19/19	10/21/19
5.1.10 Create login or authentication models	31	10/21/19	11/20/19
5.1.11 Create forgot password	31	10/21/19	11/20/19
5.1.12 Generate add API and save in database	31	10/21/19	11/20/19
5.1.13 Generate search module	7	11/20/19	11/27/19
5.1.14 Generate search module	12	02/20/20	02/29/20
5.1.15 Generate search module	20	03/01/20	03/20/20
5.1.16 Integration, testing and fix defects	27	03/20/20	04/06/20

Table 6. WBS

Figure 7. Gantt Chart is our project schedule with the task and their respective dates.

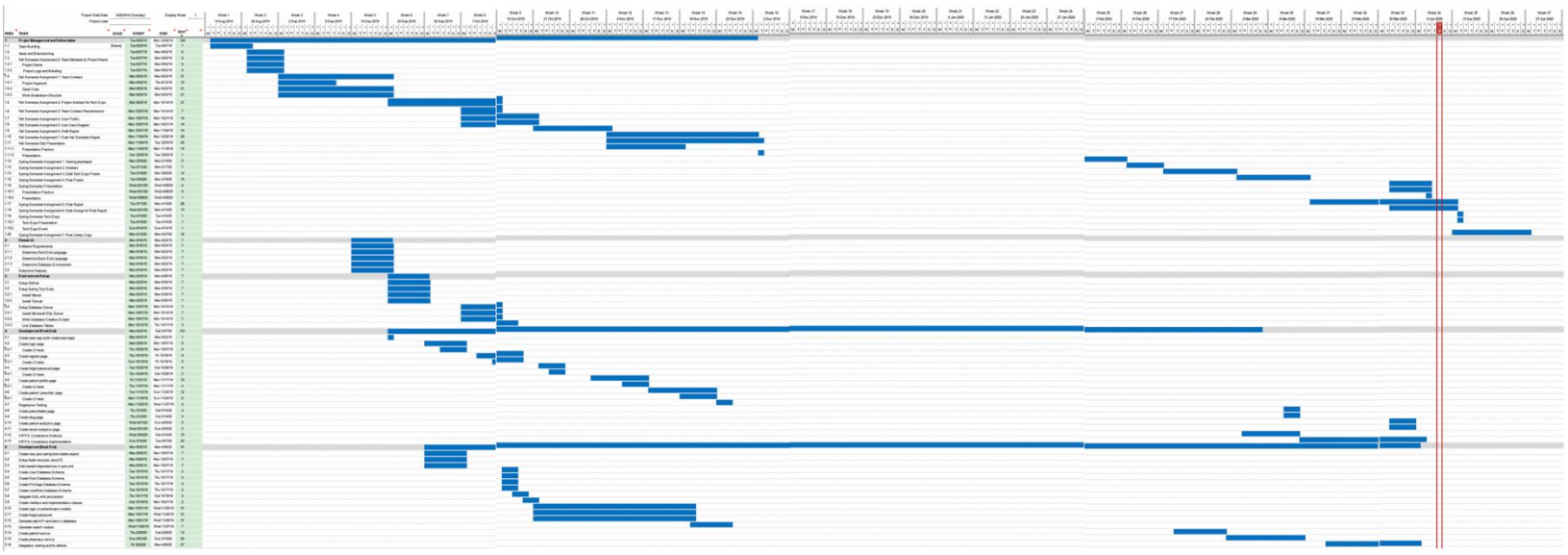

Figure 7. Gantt Chart

TECHNICAL ELEMENTS

Network (hardware/infrastructure)

Since SafeMeds follows a microservice paradigm, it uses distributed computing to serve requests. For the database, we chose SQL Server. To best host this architecture, we chose Azure SQL, which is a cloud-based SQL server, which can scale to any traffic demands. To host the services, SafeMeds uses Azure App Services, where each service gets its own app service, and uses a shared pool of resources from an app service plan. We chose NGINX as a reverse proxy for the app service plan, which proxies requests to the API Gateway. The API Gateway is also part of the app service plan, and it effectively aggregates data from the services to serve the user interface (UI). The UI is hosted on Azure Websites, which is able to serve static apps and allows for the quick serving of our application.

Application (software)

The application follows a microservices pattern, where each service can be written in any language, and communicate with one protocol, which is usually HTTP (hypertext transfer protocol) with JSON to format the data. This pattern allows for the effective parallelization of work, and it means that one member of the team can use the language they are most comfortable with, while the other members can do the same in their own services. Because of this, our services are a mixture of Java and Node.js.

The Node.js services are all written with Fastify as a framework, with Sequelize as their object-relationship mapper to communicate with SQL Server. Mocha is used as a test runner to automate running tests for quality assurance purposes.

In a microservice pattern, you typically have an API that acts as a gateway to the services and aggregates data from them when the UI or another API consumer needs data. The gateway is written in Node.js with Fastify as a framework. Mocha was used again as a test runner. The gateway secures the application using a technology called JSON (JavaScript Object Notation) web tokens, which

allows you to create a token from a hash, and send data to verify that the consumer is really who they claim to be, and that they have access to the system. It also provides an easy way to identify who the user is for auditing reasons.

The user interface is a static set of JavaScript and JSX files that create the user interface, and are served by a content distribution network (CDN) in Azure. The interface is completely data-driven, and retrieves data from the API gateway, which aggregates data from the services. The UI is written in React.

Security (how is application/project secured)

One of the most key conversations we had about the application was how a user can register. This application will be cloud-based, so anyone can get to the site, but we do not want anyone to register. So, in order to register for SafeMeds, you must receive an invite from an administrator of your institution. This keeps registration locked down to just the people who should have access to the application. In regards to the register page, one important piece of security here is that any password you enter is secured with hashing with the BCrypt hash algorithm, which hashes the password, and stores it with a salt, which makes the password hash more secure in case of data leaks. Salts are used with hashing to make it so that when you create a hash of the same value in repetition, each hash that gets created is unique (because hashes are usually one way mappings from a value to a hash). This means that if your database is leaked, the attacker will not be able to figure out the passwords, because they have random salts. This stops rainbow table attacks. This also means that us, as database administrators cannot unhash the passwords.

As previously stated, the application is secured using JSON web tokens (JWT), which act as a token to authenticate/identify users when they are requesting data from our APIs. Anytime a request is sent, before any data is retrieved, this token is checked. Since JWT are created using hash functions,

these tokens can easily be checked for malice by verifying the token is tampered with by checking the cryptographic hash.

On the UI layer, the application is secured using username and password, and will soon be protected by two factor authentication. Anytime you are accessing the application, you need to login, and if you spend too much time idle, you are forced to login again. But, aside from just logging in, the system is also safeguarded with a role and privilege based access control system. Roles grant specific privileges, so, for instance, if you are a doctor, you have the doctor role, and that grants you a specific privilege set. A user can have multiple roles, but the system never allows you to prescribe yourself medicine. So, if you are, for instance, a patient and a doctor at the same hospital, you cannot prescribe yourself medicine.

In order to access specific sections of the app, you need certain privileges (which are granted by roles), and even further than that, you need specific privileges to take certain actions. This is checked both on the UI layer, and the API layers, so anyone who tries to do something (like prescribe medicine) without permission will be immediately stopped. There are also specific functions that require you to verify your identity by typing in two factor authentication codes (besides login), like the prescription page. You will be prompted for a one-time access code to complete the prescription, just to make sure that no one can take control of your work station and prescribe whatever they want.

Since healthcare applications need to follow strict privacy rules under HIPPA (Health Insurance Portability and Accountability Act), we had to make sure SafeMeds followed these rules. In **Table 8. HIPPA Regulations** are shown, and how we met them.

Ensure the confidentiality, integrity, and availability of all e-PHI they create, receive, maintain or transmit;	Encryption at rest, JWT (JSON Web Token), Forced HTTPS, Username/Password Auth
Identify and protect against reasonably anticipated threats to the security or integrity of the information;	Hosted in Azure (built-in cybersecurity team), Logging, Access Control Logging,

Protect against reasonably anticipated, impermissible uses or disclosures; and	Role/Privilege Based Access Control, Only Healthcare Workers Can Access Data
Ensure compliance by their workforce	Possibility of Future Custom Role/Privilege Protection

Table 8. HIPPA Regulations

VISUAL

User interface

Figure 9. Login Page Visual shows the login page and will be the first page the user sees. It will allow for users to verify their identity, and gain access into the SafeMeds system securely. Whether the user is a doctor, patient, or pharmacist, they will login through this screen and be directed based on his or her individual role.

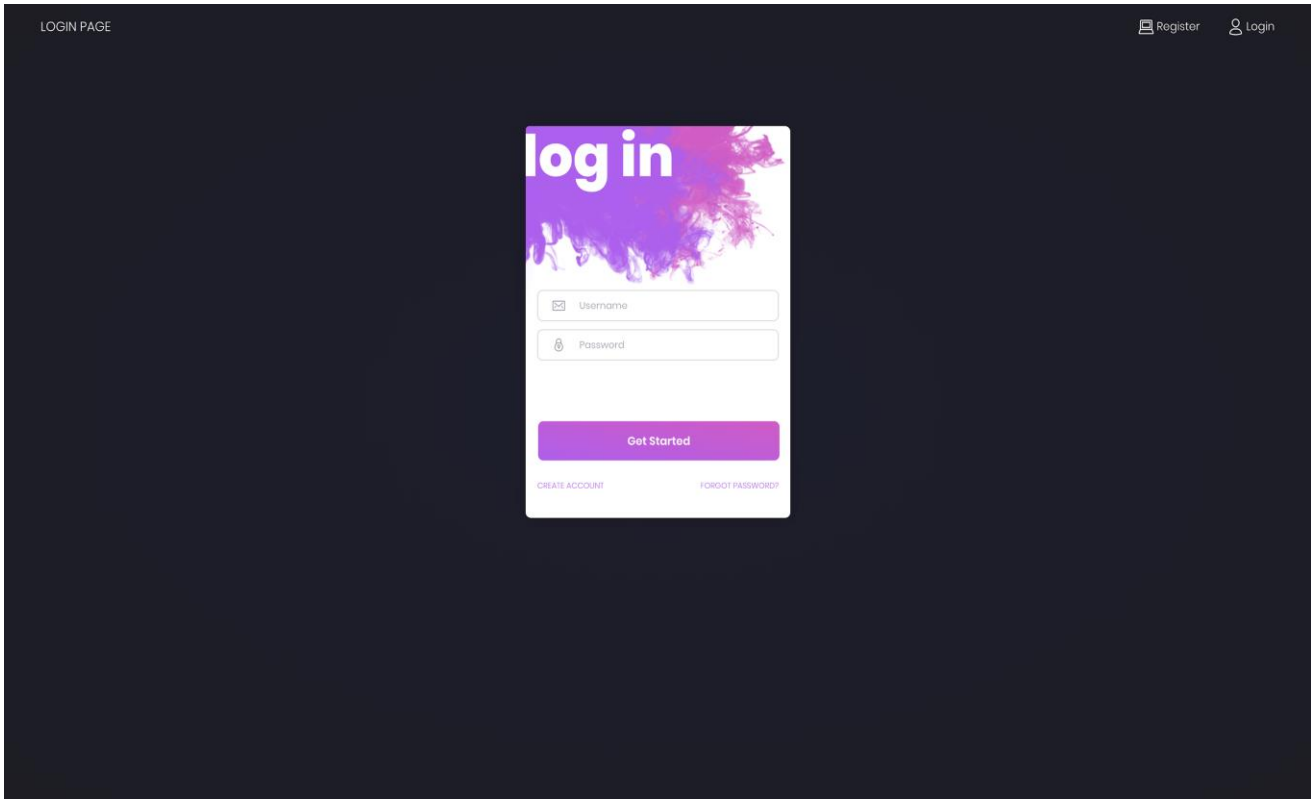

Figure 9. Login Page Visual

Figure 10. Adding Drugs/Prescribables shows that administrators and pharmacists can create new drugs. The way that the data is modeled in the system is that there is a drug, take Tylenol for example. Tylenol comes in many forms, and is not prescribable in itself, so that is where prescribables come in. Prescribables are the prescribable form of the drug. For example, Tylenol 200mg would be prescribable. All of this data feeds into the prescription process.

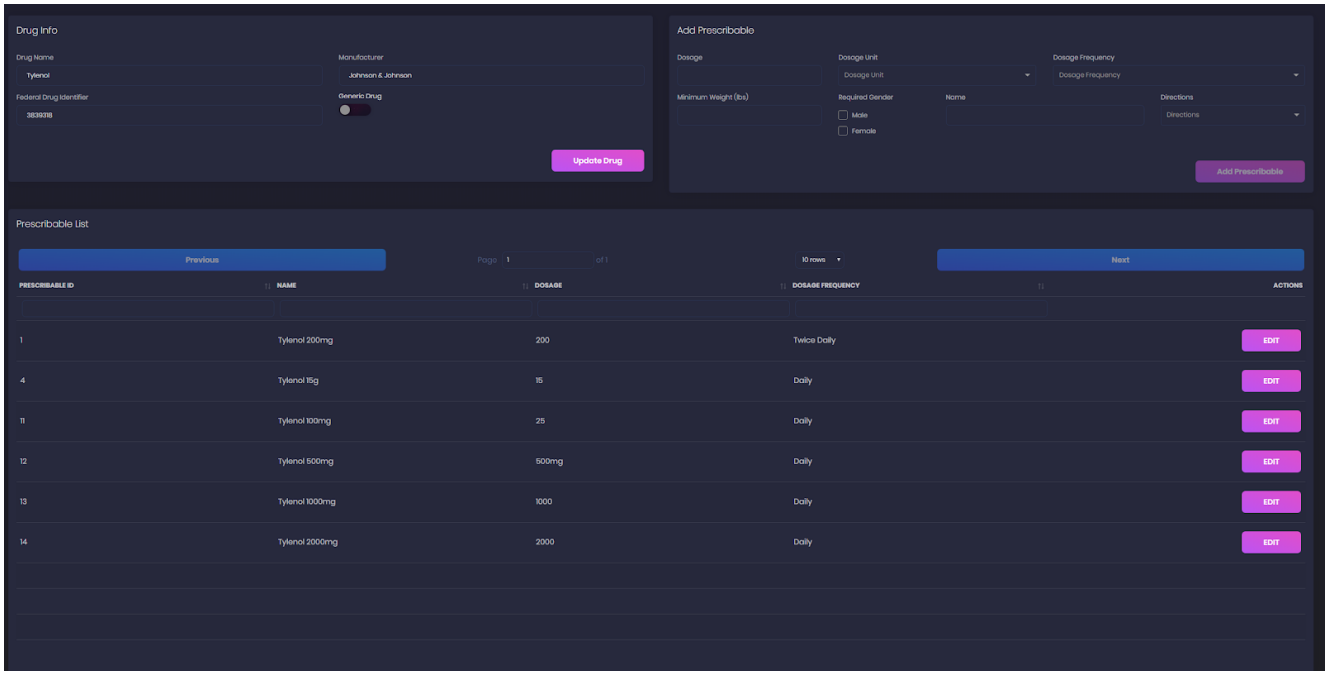

Figure 10. Adding Drugs/Prescribables

Figure 11. Prescription Process is where SafeMeds really comes into play. You can prescribe patients prescribables on this page.

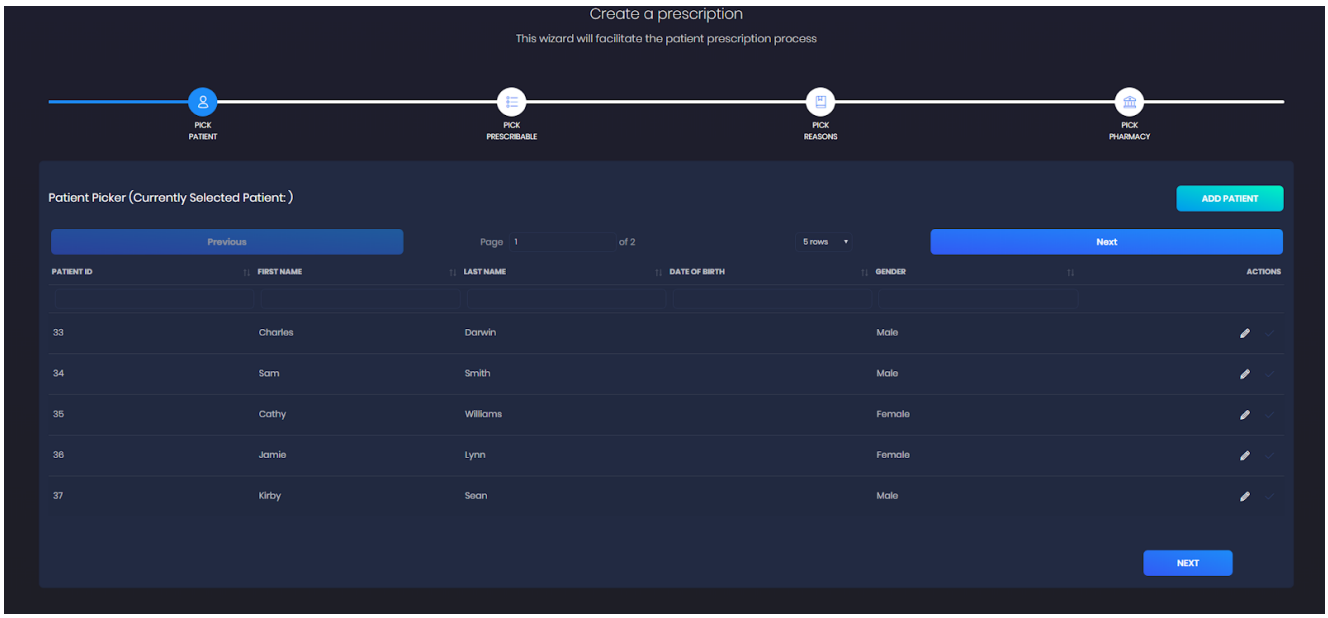

Figure 11. Prescription Process

Figure 12. Prescription Uniqueness shows how the uniqueness comes in when you actually prescribe the drug in that it does a check to see if that patient has already received that drug during that time period. This stops them in their tracks if they already have a script for it.

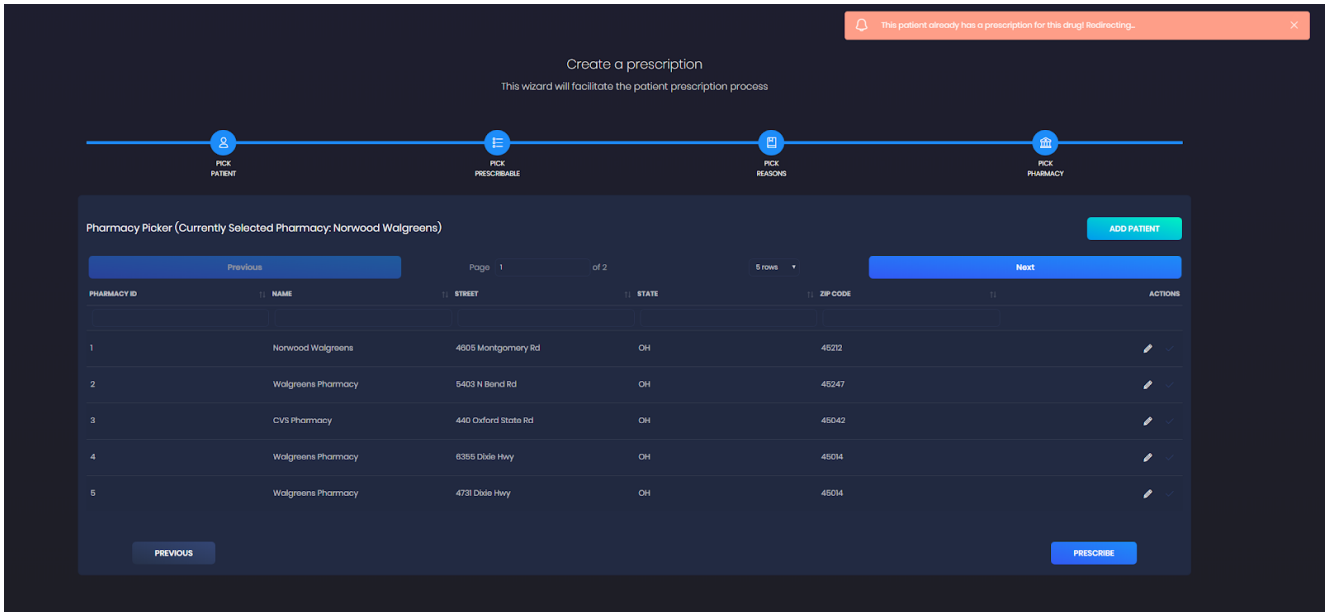

Figure 12. Prescription Uniqueness

Figure 13. Register Page Visual shows the register page and it is where users will register after they are invited by an administrative SafeMeds user. This will grant them access to the institution that they were invited to. This page features an eye-appealing design, and assures that no person registers with SafeMeds twice, so they can be tracked in one central place.

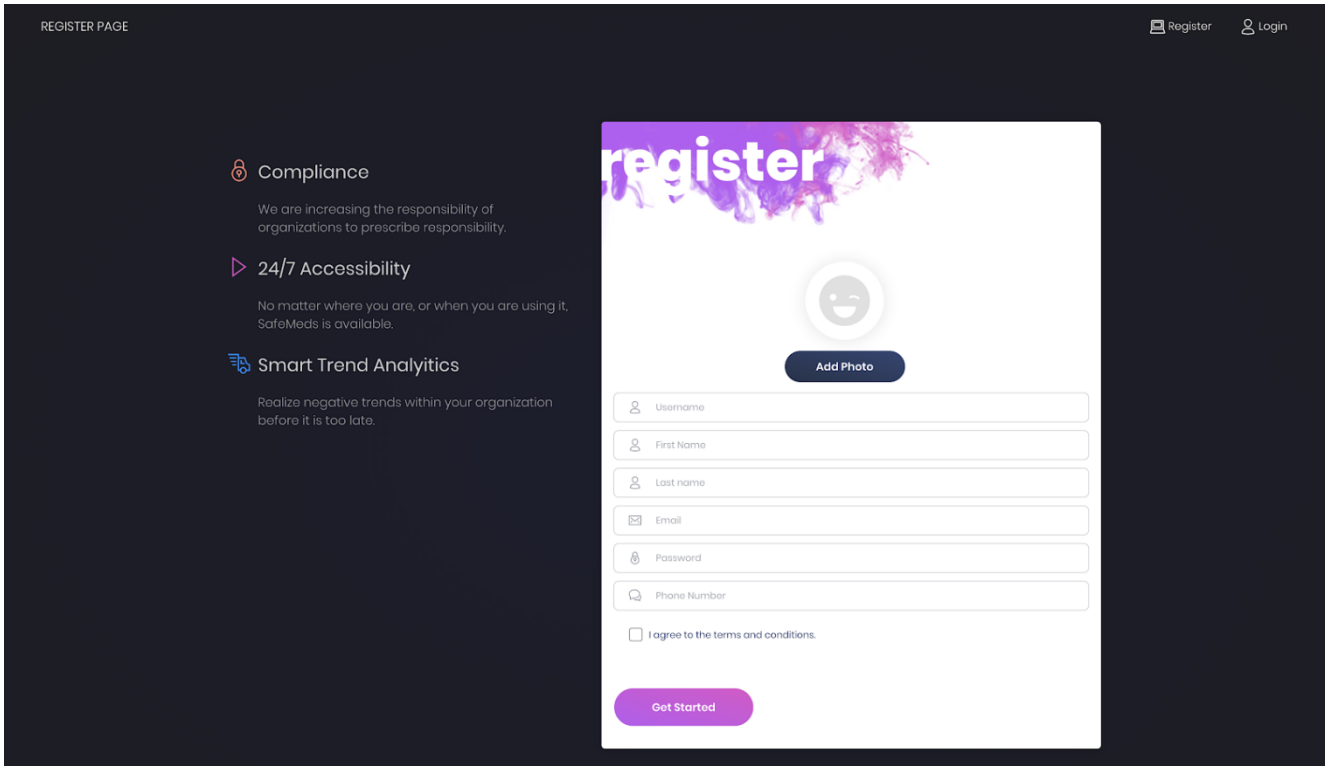

Figure 13. Register Page Visual

Figure 14. Doctor Analytics This page can be used to see the data trends across doctors to see if they are appropriately prescribing. This is available to both administrators and doctors. Organizations, or even federal organizations, can use this to really find the trends and intercept any issues before they become bigger.

Figure 14. Doctor Analytics

Figure 15. Patient Analytics shows a page that is similar to the doctor analytics, but you can actually see specific patients data. This means that you can find people who are potentially abusing very easily. You can see that this patient in the picture gets a lot of oxycontin, and this is where the conversation about abuse could begin!

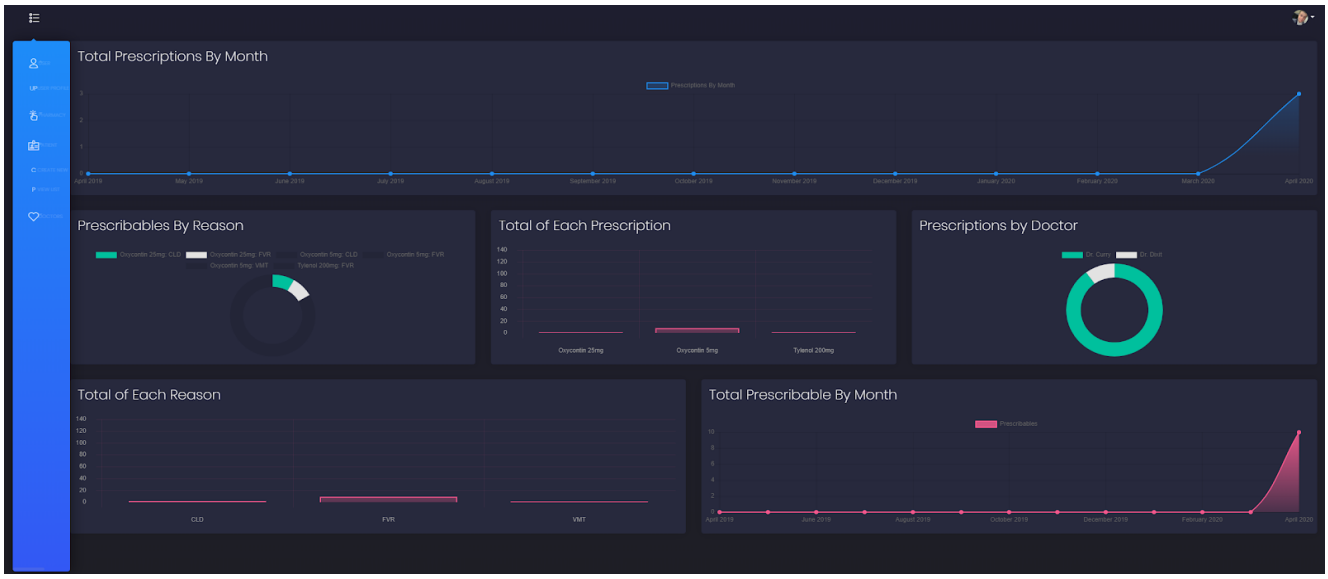

Figure 15. Patient Analytics

TEST PLAN

Overview and Methodology

For our testing methodology, we have three approaches:

- Automated UI testing
- Automated API testing
- Manual smoke/regression (depending on the change) testing

First Approach - UI Testing

For the first approach, some of the pages have automated testing using Mocha.js with Webdriver.io. When every feature is created in Github, we attach a task item to it to build automation testing for it. These tests run any time code is pushed for the project, so any time someone makes a change, all tests will run to make sure the changes had no breaks. It will stop them from merging the code into the project if all test cases did not pass. This makes sure that we never merge something with issues into the production branch, which could potentially release bugs. The good part about these automation tests is that someone does not have to test manually every single time, and it keeps the developer who made the feature from testing the feature.

These test cases are originally built out to be happy path cases, but if we come across other bugs, we will build those in as test cases so we do not release the same bug again.

Second Approach - API Testing

API testing can be critical to make sure that the consumers of your APIs are receiving what they are expecting. Thus, we implemented API tests with Mocha. API tests are a quick way of knowing whether or not your changes changed business logic. If it did, and the UI was not appropriately updated, this could cause breaks. So, this is a really easy way to find issues quickly. This is another set of tests that are automatically run any time code is pushed to the production branch of code, and will stop the changes if the tests do not pass.

Third Approach - Manual Testing

Our final step of quality assurance is manual testing. To manage test cases, we are using Test Quality, which integrates with Github (where our source code is stored). Any of the manual test cases that are run are stored in Test Quality, which means that we can keep track of who ran what when. Our manual quality assurance testing workflow goes as follows:

1. Developer runs basic test cases as feature is developed to make sure requirements are met
2. Developer pushes change to staging branch
3. Developer lets QA know a change was pushed
4. QA runs test cases
5. If any tests fail, a bug is created in the Github project with the info on how to recreate. The cycle then repeats until the bug is fixed

Figure 16. Sample Test Case shows a test case for testing the login page to make sure it succeeds. This is a happy path test, but it is just an example.

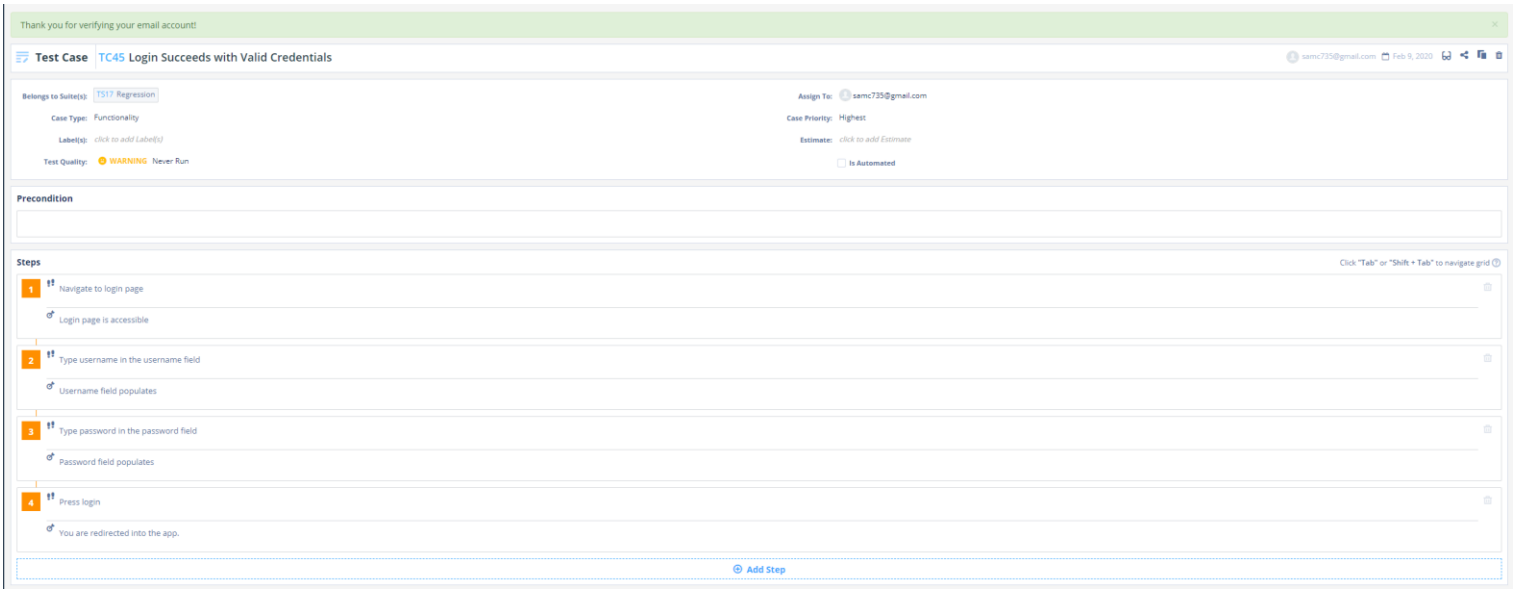

Figure 16. Sample Test Case

Scope of Testing

Every feature within the application has a test case that encapsulates it. This does not just include happy path tests, it includes any previous bugs we have found (testing to make sure they are not reintroduced), but it also includes any other cases that QA feels could break the system. Testing in the application, since it is role-based, means that you have to test it as every role to make sure no one gains access to a feature they are not supposed to have, and that everyone can use the features they are supposed to. **Figure 17. Test Plan** shows overall test cases for the application.

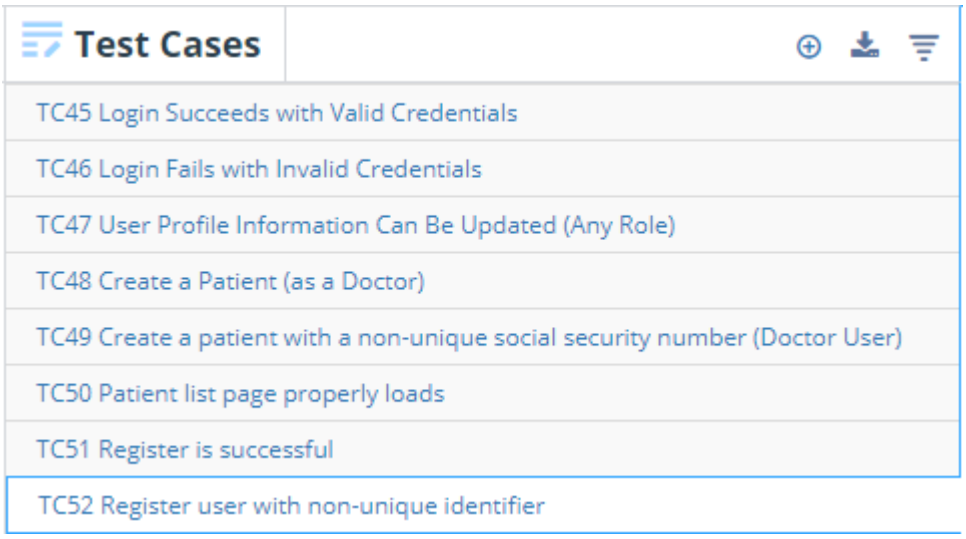

Figure 17. Test Plan

Objectives

The objective of our testing was the following items:

- All features must be covered by some sort of test that covers any permutation of user roles (doctor, patient, pharmacist, administrator) and any combination of data — not just happy path tests

- All bugs found must result in a Github bug being created and assigned to a developer with sufficient description to reproduce
- Any and all bugs are closed before IT expo
- The final release before IT expo must pass QA review

Logging Test and Procedures

In the three approaches we took, all resulted in a Github bug if an issue was found. These issues are bubbled up as issues in Github, and tagged as bugs so that the developers can pick them up and resolve them.

Our test cases remained in Test Quality because Github has no real way of managing test cases, and since Test Quality integrated nicely with Github, it was picked to document test cases. This helps us track the progress of testing, so we know how close to release we are, and also helps keep track of any prior bugs or defects that were found.

Pass-Fail Conditions

It is expected that SafeMeds must pass all listed test cases in order to be successful. If an unexpected result or bug occurs while performing any test case, it will be considered to be failed and the issues should be marked as a bug and the developer should be informed in order to resolve i

Test Cases

User Login

STEPS	EXPECTED OUTCOME
1. Enter User credentials	User will be redirected to the home page, according to user role.
2. Click Login	

User Logoff

STEPS	EXPECTED OUTCOME
1. Enter User credentials	User will be redirected to the login page after logging off.
2. Click Login	
3. Click Log off	

Add a Patient

STEPS	EXPECTED OUTCOME
1. Login as User	User will redirect to patient profile. User will be able to view patientid, name, etc. in patient list
2. Enter patient basic info	
3. Click Next	
4. Enter patient user info and insurance info	
5. Click Submit	
6. Verify patient was added in patient list	

Add a Pharmacy

STEPS	EXPECTED OUTCOME
1. Login as User	User will be redirected to same page to make update to pharmacy. User will be able to view pharmacy list and view pharmacy.
2. Click add pharmacy & fill form	
3. Click create pharmacy	
4. Verify pharmacy added in pharmacy list	

Add a Drug

STEPS	EXPECTED OUTCOME
1. Login as User	User will be redirected to page with drug info, prescribable and prescribable list.
2. Click add drug & fill form	
3. Click create drug	
4. Verify drug added in drug list	

Edit a Patient

STEPS	EXPECTED OUTCOME
1. Login as User	User will be able to view patient on patient list with any modifications.
2. Click patient-view list	
3. Click on Edit on selected patient	
4. Edit patient info	
5. Click Save Patient	
6. Verify patient edited in patient list	

Edit a Pharmacy

STEPS	EXPECTED OUTCOME
1. Login as User	User will be able to view pharmacy on pharmacy list with any modifications.
2. Select pharmacy list	
3. Click on Edit on selected pharmacy	
4. Edit pharmacy profile	
5. Click Update Pharmacy	
6. Verify pharmacy edited in pharmacy list	

Edit a Drug

STEPS	EXPECTED OUTCOME
1. Login as User	User will be able to view drug on drug list with any modifications.
2. Select drug list	
3. Click on Edit on selected drug	
4. Edit drug info	
5. Click Update Drug	
6. Verify drug edited in drug list	

Create Prescription

STEPS	EXPECTED OUTCOME
1. Login as User	User will get message at top "successfully prescribed the drug. Redirecting…"
2. Select patient and click next	
3. Click selected prescribable and give start and end date and click next	
4. Click prescribable	
5. Click reason and click next	
6. Select pharmacy and click prescribe	

Create Prescription for Same User, Same Drug, Within given timeframe, Within Same Pharmacy

STEPS	EXPECTED OUTCOME
1. Login as User	User will get message at top "Patient already has a prescription for this drug. Redirecting…"
2. Select patient and click next	
3. Click selected prescribable and give start and end date and click next	
4. Click prescribable	
5. Click reason and click next	
6. Select pharmacy and click prescribe	

Test Results

Test results are all tracked in Test Quality. However, for the automated tests, these are kept track of in the Github build process. **Figure 18. Result of Test** shows the results of automation tests:

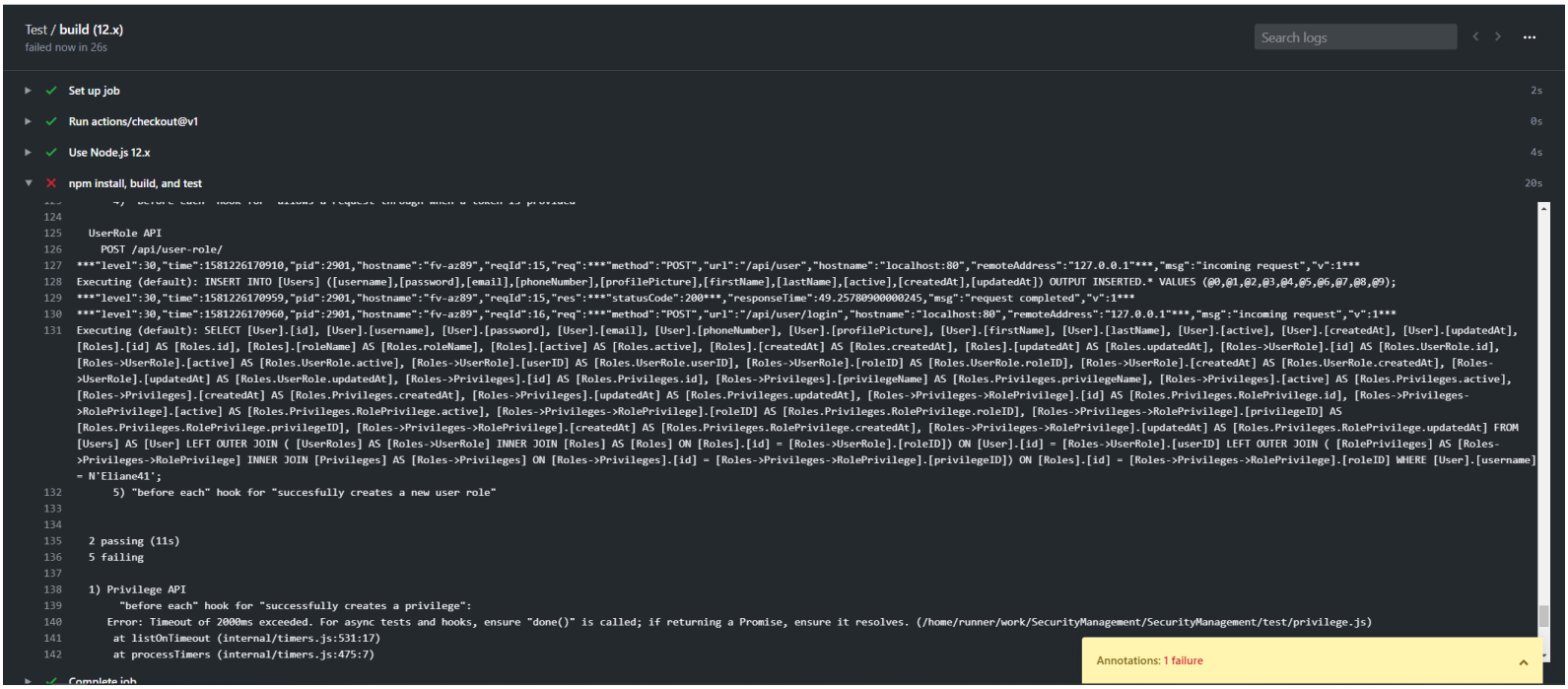

Figure 18. Result of Test

This is an example of when a test fails. It will stop the pull request from being able to be merged, and it will also send an email to the creator saying that something is broken.

Figure 19. Progress in Testing shows the dashboard for the results of any and all manual testing:

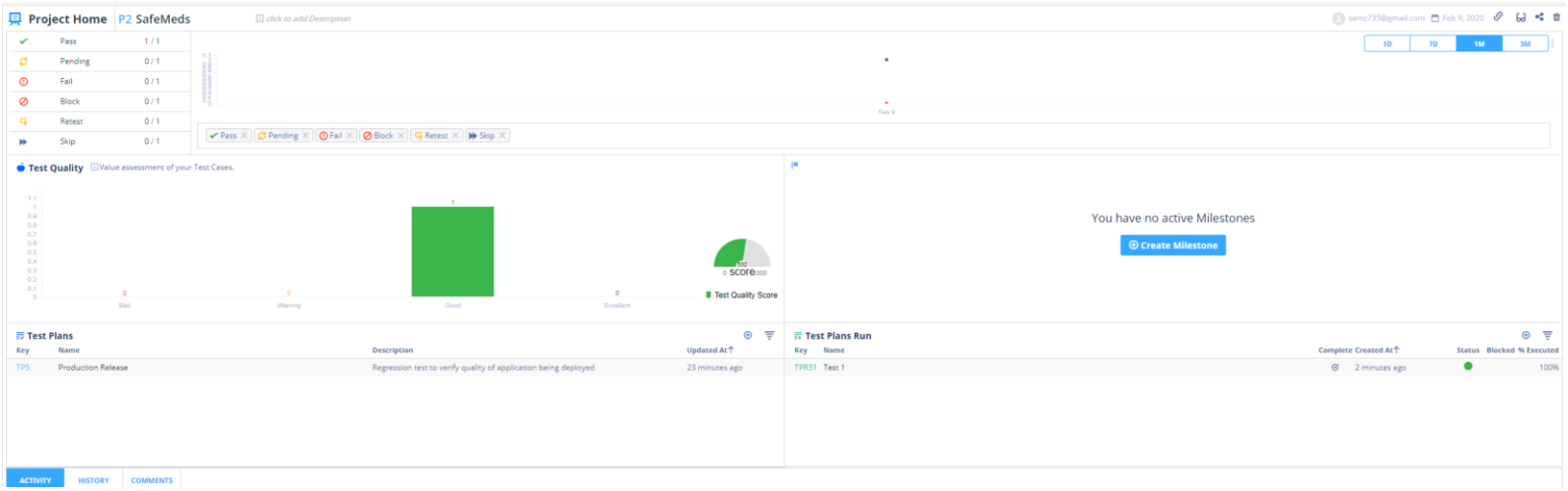

Figure 19. Progress in Testing

As you can see, it shows the progress of all tests that have been run, which indicates the status of the release. You can see each specific test case run, and the results.

Problems Encountered and How Solved It

We learned that it is really important to test every single role for a user. Testing just one role is going to result in issues down the line. We also learned that we need to do a lot more testing in general, and test more cases.

We learned how important continuous integration is. At some points, with the number of changes going in at once, constantly deploying manually would have been a huge waste of time. Thanks to Azure Devops, we were able to establish a good pipeline for this. **Figure 20.** Build shows how many pushes went out, and how much stuff was getting built.

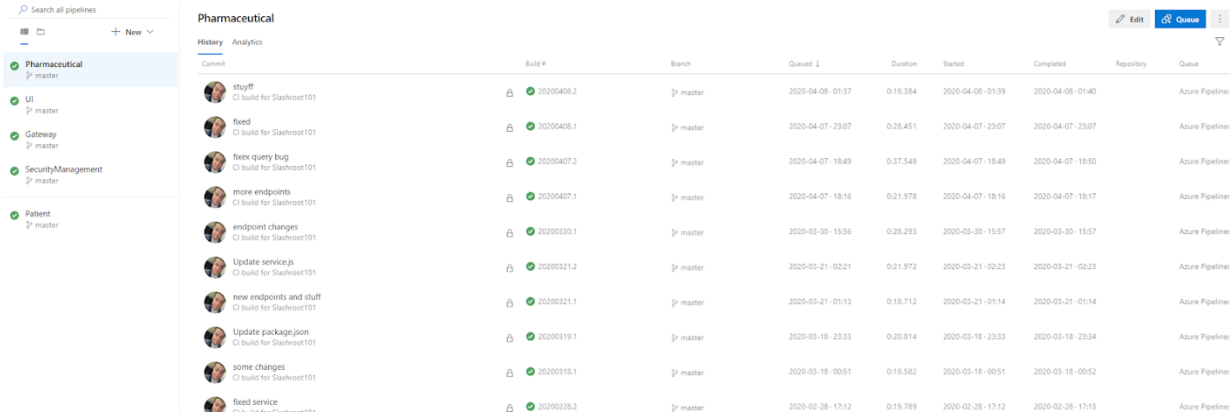

Figure 20. Build

We did not really consider how long manually testing takes. While you think it would not take that much time, you have to set up the tests, run the tests, run every case, and it is very time consuming. This helped us learn the importance of greater automation, as this can take your time to release way down.

The final result of what we learned is better testing procedures make a better product, and if we had more time, we would definitely invest in better testing or maybe even test-driven development.

CONCLUSION

Fall Semester 2019

During the Fall Semester, researching, designing and creating some of the APIs were accomplished. Samuel developed the front-end software with REACT. He also created the Security Management API using node.js as well as the API Gateway. Vriti developed the back-end software with java and created the Patient API. Vriti was new to docker and Azure so she was able to learn throughout the process.

Initially, we split the work in front-end and back-end, but because we were using a template on the UI side, there was less work on the front-end so we ended up splitting the backend work. Samuel picked up the login/register API and Vriti had already started the Patient API. However, there was some confusion on Vriti's end where she incorporated the login and creation on her API. After realizing that this was not needed and Samuel had already finished this, she restructured her database schema and proceeded to work on getting the endpoints for the Patient API. Patient API will allow doctors to retrieve all patients based on doctorid. Unfortunately, she had created just 1 API that had two different services which was not what was initially decided. She then broke it into two microservices, 1 for patient and 1 for prescription.

Another small issue that came up was finding a free server which we can deploy our application and have our database all in one. Samuel had one ready to go but neither had that communication of sharing that connection string. Vriti was able to get the connection string and make the connection of the server to make it running on the same page.

Prescription API was created in order to save prescription information along with which prescription belongs to which patient. Users are able to retrieve a list of prescriptions for a specific patient based on patientid. Some data is kept static such as drug, prescribable, etc. This allows prescriptions to be filled easier with already entered data.

Spring Semester 2020

Something that proved very beneficial to the project as a whole was developing a strong CI/CD (continuous integration/continuous deployment) pipeline. This was done with Azure Devops. Any time a change was made, this was automatically pushed up to the production environment with the appropriate tests run (if any). This helped expedite changes going out, and issues in production since the application was deployed the same way every time. Having this shared cloud environment for testing was very helpful.

We moved away from Java Spring during the Spring semester because we decided to split the roles up where Sam would take the full-stack development, and Vriti would do QA. This proved out to be a very good choice as Vriti found many bugs that helped contribute to the overall quality of the demo.

Most of the application was completed with a few bugs that were easy to fix while some took up time. As the application developed, new pages were added and functions were added. For instance, a prescription page was added towards the end which meant testing the prescription page a bit tightly to make sure a drug cannot be prescribed multiple times during the a given timeframe. This was not working as decided and it allowed all drugs to be saved. This was a major bug that needed to be fixed which was eventually and for some reason it went back to saving all the drugs. Samuel spent time to get it revised and made sure it was functioning.

One main bug was the prescribable dropdown in the prescribe page. For some reason the dropdown would not show anything despite choosing a drug and this took up many days for Sam to solve. It was crucial to our application so it had to be solved before our demo. Samuel spent time solving all the bugs and Vriti spent her time testing all scenarios and she found small bugs which were then handled by Sam. Both spent a majority of their making sure all functions were

working and all of the pages were showing up properly. Samuel added an analytics page which showed which doctor prescribed which medicine the most and which patient takes certain medicine and which doctor they get it prescribed to, etc. At the end, all of the pages were working and all of the functions we wanted were doing what they were supposed to at a given time.

APPENDIX A. REFERENCES

"Prescription Drug Abuse." *Drug Abuse*, 2011, drugabuse.gov.
https://www.drugabuse.gov/sites/default/files/rxreportfinalprint.pdf

"Opioid Overdose." *Centers for Disease Control and Prevention,* Centers for Disease Control and Prevention.
https://www.cdc.gov/drugoverdose/index.html

APPENDIX B. POSTER