# Day Health Planner


## Team 4YourHealth

Zachary Nielson | Daniel Hudy | Peter Butler

Karen Snavely | Eric Majchrzak | Calvin D. Rosario

## Trillium Health

AJ

## Faculty Coach

Professor Hawker


## Project Overview

Trillium Health currently uses a combination of sharepoint and paper documents to keep track of all of their data. As their company grows they are starting to deal with the overhead that comes with paper documentation as well as the limitations of sharepoint. With this project Trillium is looking to move all of their documentation to one shared system, a system that not only merges the two old systems but also improves and expands on the features they already have.

### Desired Solution and Project Scope

The scope for this project includes everything Trillium needs to replace their current system with our system in addition to a couple of extra features. The scope primarily consists of three parts, the database, the clinician portal, and the session application. The three of those parts each have a distinct scope.

The database must be able to track all of the data that Trillium is looking to store. A huge part of this involves scanning and storing documents on a per patient basis. Each document may have different pieces of data that need to be tracked in the database. The database will be in charge of keeping track of what pieces of data each document is associated with as well as storing each instance of that data that has been input. The database will also store session data from each of Trillium's patient sessions. This includes everything from departure and arrival times, session notes, and individual session notes. More specifics on data to be stored can be found in the product features section.

The clinician portal is the primary user interface for the system. This will be a web application accessible from any compatible web browser. The clinician portal has four major parts: patients, sessions, documents, and statistics/billing. The patients section will be where the user can add, modify, or check any patient data stored in the system. This includes basic functionality such as adding a new patient, adding a new scanned document to a patient, or checking on a patient's basic info. The session section is where a user can go to add, modify, or check on any session data. This includes functionality such as scheduling a new session, modifying a session leader, or checking which classes are scheduled on a specific day. The third piece of functionality desired by Trillium is documents. Trillium expects to be able to add any type of document to the system as well as information from these documents stored as data in the database. The final basic part is statistics/billing. The statistics/billing section will be where the user can find tools that are able to run tracking associated with patient data and billing. This may involve a tool that can get the patients that are billable for the week or similar tools.

The session application is used by session leaders to standardize attendance tracking and session note-keeping. The session application will primarily be used to keep track of patients signing in to sessions. The session application will interface with a fingerprint scanner that the user will need to use to sign into a session. This gives Trillium a unique and precise timestamp for each individual that will allow them to determine how long a patient has been at each session.

## Basic Requirements

The Day Health Planner is an Enterprise application, and as such has many requirements. When gathering requirements we broke them into three main parts, database requirements, web application requirements, and the session application requirements.

Data was the center of the design at the start of our project. Everything in the project revolves around the database and with Trillium having such a complex system already in place there was a great need to gather requirements for the database. After many meetings with AJ there were two major requirements that stood out for the database. The first requirement was that the database needed to be able to keep track of new types of documents and the fields on those documents without having to change any code. The second major requirement for the database was that all changes to the database must be kept track of in case of an audit. Both of these requirements were architecturally significant when we were creating the database.

The web application was the main view in which the end users would be using on a day to day basis. This was the portal through which non-technological users would be able to view and manipulate the database. One of the major requirements that was made clear to use right at the start of the project was that Trillium expected users with a range of technological skill levels to be able to use the system. This was a major usability requirement that we had to keep in our minds throughout the project's lifecycle. Another major requirement for the web application was modifiability. In the future it is very likely that another software team will end up working on this

project and when they do the system should be easy to understand and update. A lot of our other requirements for the web application were related to the ability to modify data or have specific workflows. These requirements were similar to one another and were not as architecturally as significant as some of our other requirements but did composed the bulk of them.

The final piece of the system was the session application. The session application had one major requirement and that was a working fingerprint scanner that could verify a patient's identity. That requirement was huge to us because none of us had ever worked with a fingerprint scanner before so we knew that would be a technological gap for us to jump. With the session application came another requirement that would be difficult for us to meet and that was getting the application to work outside of Trillium's office. With all the fingerprint data being on the Trillium servers and HIPPA laws being strict about transfer patient data over the internet this was a requirement that we believe we underestimated at the start of the project. Overall the session application had a lot of requirements that were difficult to meet.

If you would like more information on requirements and have access to our SRS and use case document please see those as they are much more detailed than the information you will find here.

## Constraints

With six people on our team constraints were the last thing on our mind when starting this project. By the end of the project we had many constraints ranging from the database not integrating with certain technologies well to not having enough time to get everything done that we wanted to get done.

Looking back on the project it is very likely that our biggest constraint on was the lack of technical experience when it came to Microsoft SQL Server. SQL Server was the database we were required to use because it is already in use at Trillium and it is what their people know how to work with. At almost every stage of the project there was some form of database problem due to our lack of technological experience with databases. No one on the team had set up a project with SQL Server before and it turns out that SQL Server is one of the most difficult SQL databases to get to play nice with other applications. Had we had fewer database problems this project likely would have progressed at a much faster rate.

Another major constraint we had as a team was the session application's fingerprint scanner. Coming into this project we all identified the fingerprint scanner as a major concern and spent a lot of time trying to figure out a solution. Because none of us knew how to get the technology working we spent a lot of time online trying to figure out what the best option was. In the long run we ended up outsourcing the session application which brought with it a whole different batch of problems in the form of working with another development team and trying to integrate a non-ideal product into ours. Overall the session application and the fingerprint scanner ended up being a huge constraint for us.

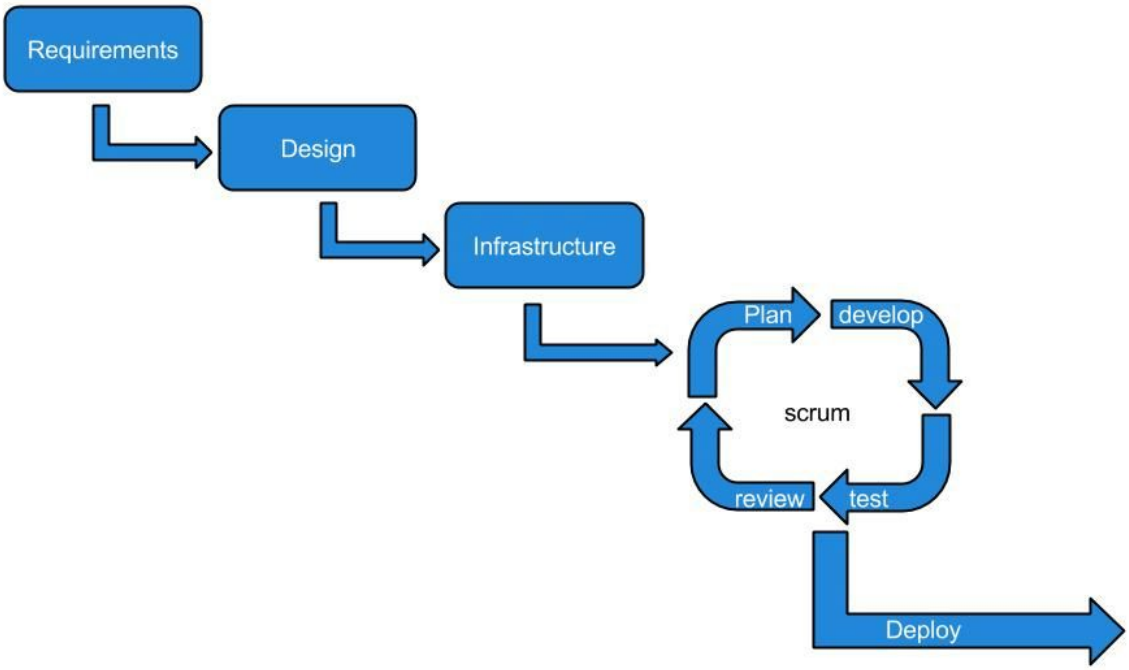
Our final major constraint for this project was definitely time. Throughout the life of this project we've all had an inconsistent amount of time to spend working on it. From classes to work to social lives getting six people to sync up for anything is remarkably difficult. By the time we finally hit our stride using GitHub's issue tracker and Slack for communication the project was almost done. Had this project had an additional five weeks we would have likely gotten a significant amount of additional work done as our process was refined and we had selected tools that were finally working well for us.

## Development Process

Throughout the life of this project our process has changed many times quite significantly. We started with Scrum and changed the process to what we needed it to be at different times in the project. Each iteration of our process brought us closer and closer to a process that worked for our team and the end result is something that we are all now proud of and know works for us.

At the start of the project our team decided to use Scrum as our process methodology. For a team of six it seemed like a good idea so we could use the backlog to communicate what needed to be done rather than figuring out what people needed to do. Whenever someone had time they should grab a task and do it. As the first semester went on we started to realize that while we were going through the process of Scrum, we were actually following a pattern closer to a waterfall pattern. We had done a heavy requirements gathering phase and a relatively design heavy phase. After analysing the backlog we realized that we spent so much time in these steps because the customer had asked for quite a bit of documentation. This led us to rework our development process and focus on doing a better job sticking to Scrum in the future.

In the coming weeks we began to focus on implementation, which was much easier for us to wrap Scrum around. As the weeks went on we realized that we were using Scrum the way it was meant to be used in development. With Scrum in active use we noticed an increase in working code as well as a spike in customer satisfaction. When we started to deploy to Trillium's servers we knew the process had produced some results. The end result was a waterfall process that led into an iterative implementation cycle which we carried through to the end of the project (See Development Figure 1).




*Development Figure 1: The end result of what our process looked like.*

## Project Schedule: Planned and Actual

Our original project plan was much different than what actually occurred. Because of various delays from the database not integrating correctly to the fingerprint scanner not working the way we thought it would our original estimated dates were pretty far off. Below is a table of our major projected milestones, when we estimated them to be complete, when they were actually complete, and why they were late. Many of the tasks were only late because another task was late, but if one had a reason other than that it will be listed in the reason for delay column.

| **Task Planned** | **Planned Completion** | **Actual Completion** | **Reason for Delay** |
|---|---|---|---|
| Database Setup | 3/29 | 4/12 | Underestimated task |
| Spring Models Hooked into the Database | 4/12 | Did not happen | Technology problems |

| | | | |
|---|---|---|---|
| Barebone Views | 4/26 | 5/31 | |
| Patient View Done | 6/7 | 7/26 | Underestimated Task |
| Session View Done | 6/14 | 6/7 | |
| Documents Done | 7/5 | 7/12 | |
| Billing Done | 7/12 | 7/19 | |
| Statistics Done | 7/12 | Did not happen | Underestimated task |
| Integrate Session Application | 7/12 | 8/2 | Negotiating with 360 Biometrics for changes |

## System Design

Designing an enterprise level system as a team from the ground up is something that most teams would struggle with. As a team of students we knew going into this project that we all had little to no experience with enterprise development, let alone enterprise design. We went through several iterations of design but in looking back on it the four largest design decisions we made were the database schema, whether to build a web application or a desktop application, our tech stack, and outsourcing the fingerprint scanner application.

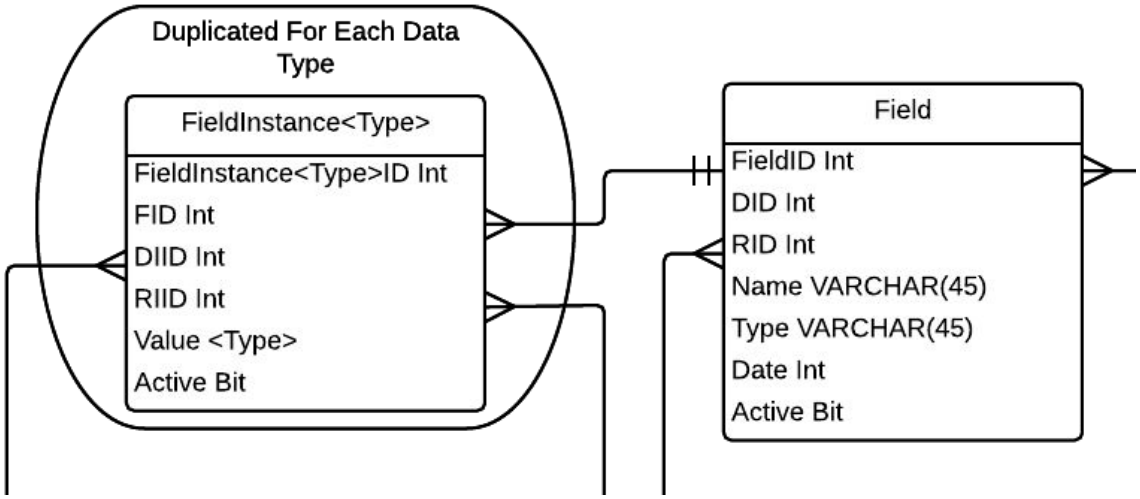
### Data Driven Design

From the start we knew that the product we were asked to create was all about data. Recording, viewing, and modifying data is what the product was all about. So with that in mind our goal was to get the database designed and approved as quickly as possible since that was the foundation for a lot of the other pieces of the project. We ended up spending more time on the design of the database than originally intended, but doing so really helped us learn a lot of the hidden requirements that this project had.

The design of the database was handled by half the team as to keep design meetings concise and focused while the other half of the team worked on mock user interfaces and fingerprint scanner research. We began by individually taking the requirements and coming up with a database schema based on them. During that time we individually asked questions to the project sponsor if the need arose and changed our individual designs accordingly. After a week we came together and presented the designs to one another describing the various problems we found or foresaw and our solutions to them. After all group members presented we went off on our own and refined our own designs with the other group members designs in mind. After another week we came together and found that our designs were looking more similar and that we were more agreeable on certain design points, we repeated this process for two more weeks until our designs were relatively close and then discussed with the whole team which design was the best and decided to use it as our initial database design. While the design has changed

over the lifespan of the project the core design decisions we made in our database design process remain, those being document generalization, fields, and the audit system.

One of the primary goals of the Day Health Planner was to keep track of ever changing documents and the fields recorded on those documents. As a design team we saw this as a difficult issue to tackle. We needed a database that was malleable yet well structured. If the user wanted to add a new document with new fields a lot of different variable types would need to be kept track of. Some of the initial designs that were proposed involved creating tables at run time for each new document added to the system or having a master document table that would add additional fields as needed to it. Both of these designs were flawed in that they involved the creation or modification of database schema while running and we really wanted to avoid that. After a bit of research we came up with field tables. We decided to have a table for document types and a table for each type of field possible on documents such as strings, integers and bits to name a few. We then used a separate linking table for each field type and were able to link each document type to specific field types. We then had a separate table that kept track of each instance of each document type which was linked to specific field values on the table (See Design Figure 1 for details). This design decision resonates through the project as we attempted to solve many other problems using this same generalized approach when applicable.




*Design Figure 1: Each field is kept track of in a field table while each type of field (String, int, etc) has their own table in which the values for each field associated to each document are kept.*

The second major design decision we made when designing the database was the Audit system. One of the biggest problems Trillium has had in the past is keeping an trail of all changed documentation in the case of an audit. Some of the initial solutions involved various database design patterns such as a transaction pattern with a table of all past transactions to using database tools that would track the data for us, but in the end we decided to go with our own custom solution. Our solution involved two parts, an audit table and the idea of active data and inactive data. The audit table acts as a change log for each field in the database. Whenever

a field is updated the audit table logs the user that changed the field, the location of the field, the past value for the field, and the value of the new field. With a bit of programming this system allows a developer in the future to develop a rollback system for the database if needed or for a user to look at all past changes to see who made a change and what the change was. It also helps Trillium keep compliant with the existing audit requirements that they must conform to by keeping a log of changes. The second major database design decision here was the decision to use active and inactive markers instead of deleting data. Because of our unique audit system it was difficult to keep track of data such as old patients that had been removed from the system. What we decided to do was rather than delete old table rows from the database and log it all to the audit table we would mark the table row as inactive and log the field change in the audit table. This way if old patients come back into the program or a new class come back it's as easy as changing a field to get them back into the system.

While the database design took up a good chunk of our initial time on the project we believe that the design decisions we made then have helped carry us through the project and made implementation easier than had we used some of the alternate methods proposed.

### Web Application or Desktop Application

When discussing requirements with AJ one of the major things we talked about as a group was what type of solution was the best for Trillium, a desktop application or a web application. Both types had their pros and cons so it was difficult for us to some to a decision on what we should do.

The argument for a desktop application was pretty strong initially. Many members on the team were very uncomfortable with developing a web based solution. Some of us had never worked with with web development, and those of us who had had worked with different technologies, Ruby on Rails, Angularjs, and Node.js to name a few. The one thing that was consistent across the group is that everyone knew Java development, so we started to analyze the requirements to figure out if a java based solution was possible. Upon digging deeper into the requirements we realized that since the fingerprint scanner was hardware based it would be very difficult to access through a web interface as the drivers would need to be accessed by the application. With our initial analysis we determined that the best solution would be a java based desktop application.

As a team we started to pitch our java based solution to AJ and he voiced a minor opinion towards having it be web based but was okay with either solution as long as it met the requirements. He then, however, brought up some requirements that made he believed would favor a web based solution. The first requirement was that the fingerprint scanner could be separate than the main application and should be a lightweight application tailored towards session leaders so that they could easily take notes and attendance without the rest of the UI getting in the way. This requirement made us strongly believe that the fingerprint scanner would likely have to be a separate application that integrated into the database or main application,

thus freeing us from one of the constraints tying us to the idea of a desktop application. The second requirement was that the application needed to be able to run on any pc with little to no installation for each individual machine. This requirement is really difficult to satisfy with a desktop based solution and really pushed us further into the idea of a web based solution. After a little more talk on the group's part we decided that a web based application was the best solution and went forward with that in mind.

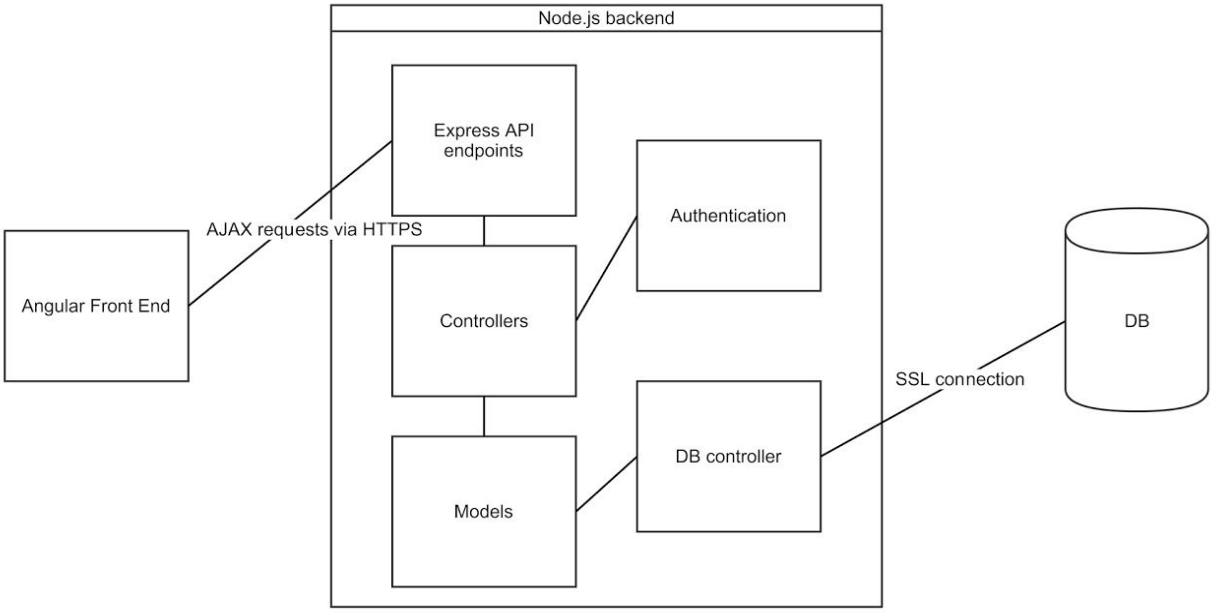
## Tech Stack

Now that the decision had been reached to do a web application we as a team had to choose the technology stack to implement our solution. The one thing we knew we needed was a Microsoft SQL Server database, but to our knowledge most web technologies have the ability to integrate with SQL Server through the use of configurable drivers, so we were free to choose a technology stack that worked best for us.

During our talks to determine whether to do a web based solution or a desktop application one thing was clear to us; a large portion of the group prefered Java and didn't know much web development. With that in mind one technology that stood out to us was the Java Spring Framework. Only one person on the team had actual experience with Spring, but it seemed like a tool that would be relatively easy for the rest of us to learn. It was a server based framework for web applications that would allow us to use java for our models and controllers and HTML, Javascript, and AngularJS for our views. After asking some other groups about it we found that the other group doing a project for Trillium was also using Java Spring and volunteered to help us with the setup as well as provide us with their notes on how to solve many of the problems that they ran into as they developed their project. With that in mind we decided that Spring was the way to go and began implementing a tech stack with Spring at the center of it all. After doing the tutorials and learning the basics we began implementing our tech stack but very quickly ran into problems. Spring had great tutorials for getting controllers setup, but documentation was sparse for getting Spring hooked up to a Microsoft SQL Server which we struggled with for many weeks. We decided to ask the other group for help but it had turned out that had not had the same requirement to use Microsoft SQL Server as us and instead used PostgreSQL, so the support we thought we had couldn't help us. By the end of the first semester we still did not have the database hooked up to the Spring framework so we decided we needed to change our tech stack up a bit.

During break week we decided a radical change was needed to catch the project up with where it needed to be. We needed to be able to rapidly develop the project and still get through our problems with the database. We built a new tech stack still using HTML and AngularJS for our view, MySQL as a temporary database while we solved our SQL Server problems, and Node.js as our controller and model. The reason we chose Node.js was that two of the team members had used it in the past and knew that it was capable of doing what we needed it to do and that they could develop rapidly with it. Within a week of switching our new technology stack had caught us up to and surpassed where we were with the old technology stack. With that being

said there were some tradeoffs that we had to make when we switch. Node.js makes data much harder to manipulate and use algorithmically, so we knew that statistics was going to be much harder for us to calculate in the new technology stack. Our solution to this was to put one team member on statistics early and have them work on it with as much time as possible, but sadly we were not able to get statistics into the final product. We did, however, document our findings and left them with Trillium so that a future team may be able to more rapidly solve this problem. As we finish the project the tech stack remains the same (see Design Figure 2).




*Design Figure 2: A basic overview of the final system's tech stack. An Angular Front end hooked up to a Node.js backend with a SQL Server database.*

## Fingerprint Scanner

The last major design decision that influenced our project heavily was the fingerprint scanner. For the first six weeks of the project we all put our heads together to come up with the best possible solution for the fingerprint scanner. None of us had any kind of experience with this technology and it seemed very crucial to the project. Our decision was to dedicate one person to the scanner, to research, design, and prototype the solution we needed to have completed. Our results were far from perfect.

At first glance the session application seemed pretty simple, it was an application that had to keep track of patient attendance for the session based on fingerprint and had to let the instructor take notes on the session and each individual patient after the session had been completed. After looking around the web for fingerprint scanners with SDKs we quickly found some highly rated ones and ordered them so we could begin prototyping the fingerprint scanner application. Upon receiving the scanners we realized that there was no SDK included with them nor could

we find any compatible one's online. After contacting the company and hearing no response we began to get worried about the scanner.

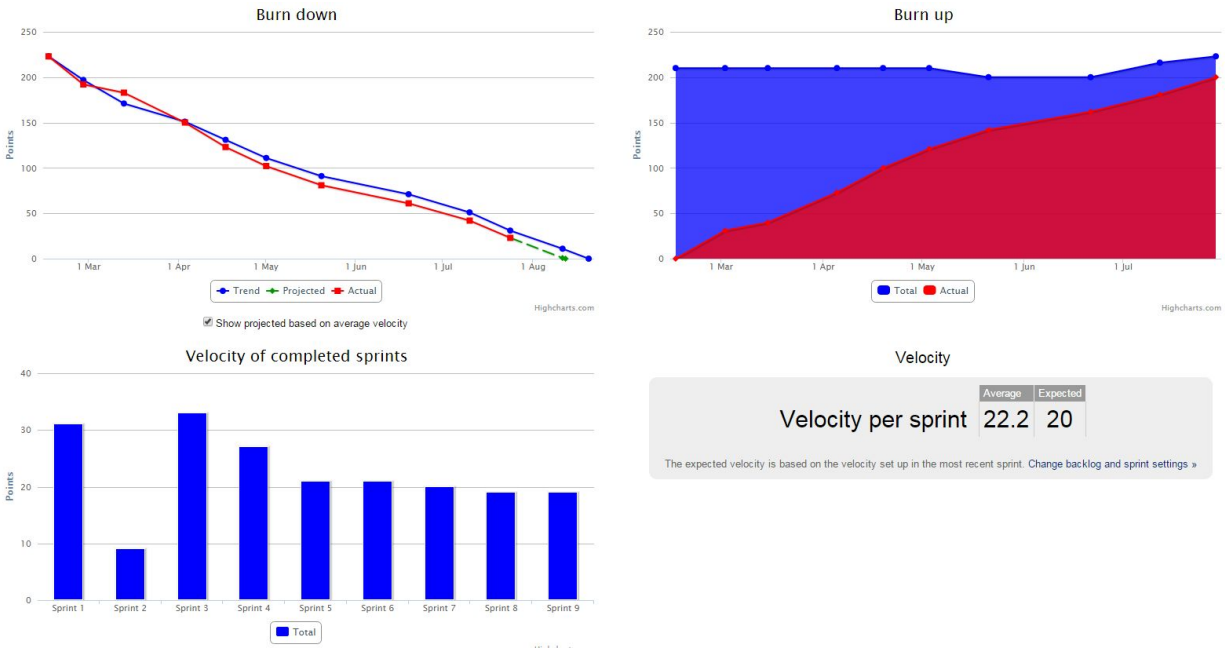
After having the sponsor spend money only to receive broken scanners we began to look into other scanners much more thoroughly and skeptically. While doing this research we came across a company called 360 Biometrics. 360 Biometrics was a company that made custom fingerprint scanning applications and sold the fingerprint scanners to go with the code. After contacting them we found out that their team would be willing to create the session application for us for a reasonable price. We got AJ in contact with them and he approved the software contract. We then spent our time building an API for their scanner to interact with our system as well as generating a list of what we thought to be thorough requirements for the 360 Biometrics team. As the weeks went on one of our team members kept in contact with 360 Biometrics as their product owner.When the team finally delivered their solution to us it was nothing like we thought it would be. After a quick demo from the 360 team it was obvious to us that the scanner application would not satisfy the requirements of the project and we asked them to fix some things so that we could use it. The team made some of the changes but claimed that the budget had run out for the project and left us with the application in a state that was not conducive to integration with our existing system. In the end the fingerprint scanner is in a partially completed state.

## Process and Product Metrics

Our team employed three main metrics for this project: Scrum Velocity, Hours Spent, and Bugs Closed. Each type of metric yielded very different results and gave us good insight to our project's progress as we learned how to better track them and utilize the results.

### Velocity

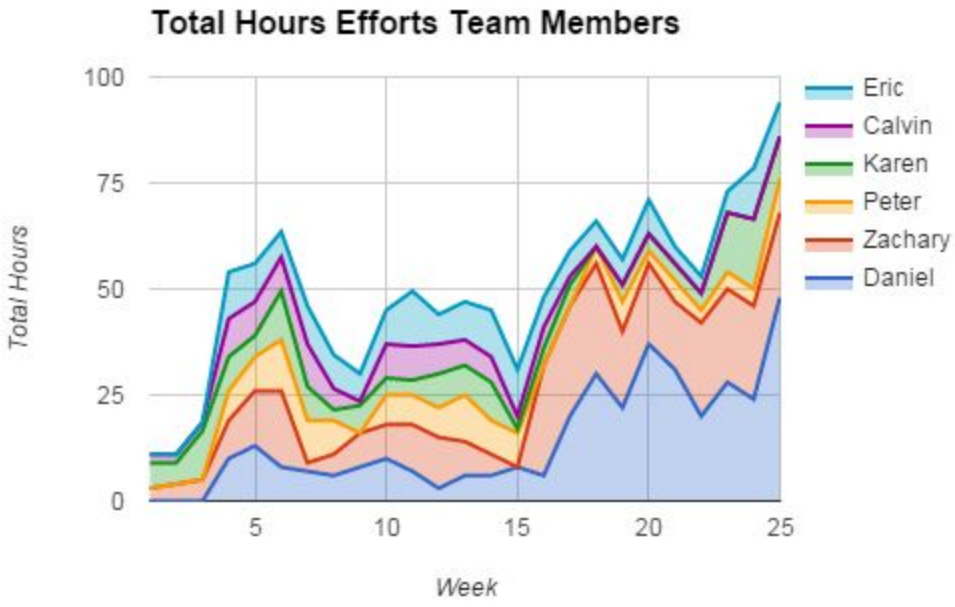
The metric that was most helpful to us over the course of this project was likely velocity. Through the project we used an agile tool called easy backlog which aided us in keeping track of our Scrum Velocity. This was an invaluable tool for planning more often than not because we knew what we could plan to get done from week to week. Our average velocity for the project was 22 story points per sprint (See Metrics Diagram 1).




*Metrics Diagram 1: The burndown chart, burn up chart, velocity per sprint, and average velocity of our project.*

## Time Tracking

Time tracking was our second metric, and while it was far less useful than velocity it definitely provided some interesting results. As you can see around week 7 we had a pretty big spike in work. Once the summer semester starts the amount of work dramatically increases all the way until week 25 where the project ends.



*Metrics Diagram 2: The total hours of effort for us as a team on a week to week basis.*

### Bug Tracking

TODO

## Product State at Time of Delivery

Our Scope for this project was definitely more than we could handle in the course of the senior project, despite that we did get quite a bit accomplished.

### Completed

- We have a working web application that will replace the existing workflow at Trillium
    - The users can add new patients to the system and track their time at Trillium through the various patient workflows
    - The user can add new classes to the system as well as new instances of those classes called sessions
    - The user can add new document types to the system with a set of tracked fields that will be stored in the database
    - An instance of any document type can be made and there is a workflow for inputting the related fields for that document type into the system
    - Billing can be kept track of for each individual patient and patients have multiple status flags to represent which stage of the billing process they are at
- There is a working SQL Server database behind the web application keeping all of the data stored in one location
- A fully functioning test plan was developed and has been used to test the product at various stages of development
- Important documentation such as the design document, requirements specification, and the process document are up to date and will be given to Trillium with the rest of the project upon completion so that future teams have something to base future modifications off of

### Partially Complete

- We have put a lot of research into what it would take to get statistics functionality done, and while it was not completed, due to time constraints, we have documentation with our progress on the feature and the steps we believe would need to be completed to get statistics done
- While the session application from 360 Biometrics can no longer be worked on the product is in a non-ideal state for what Trillium needs. For now it may be able to partially fulfill their requirements, but in the future it is very likely that a better solution will need to be found

Future Features

TODO

## Project Reflection

After reflecting on the project the team had come up with a lot of good feedback on what we learned, what went well, what could have gone better and more.

The main thing that everyone on team agreed upon was that  building an Enterprise system from the ground up was a very valuable experience. Never before did any of us have to undertake a project this big from the start. On Co-ops and jobs we had worked on systems with this scale, but we were never in charge of the entire experience, from the requirements gathering, to the design and implementation. The whole idea of choosing the best technologies for the solution was new to us and we learned how important it was to take not only the developer's experiences, but also the context of the project into account when choosing technologies.

Another thing we all realized during the retrospective was how our process evolved over the course of the product and how in the end really boosted productivity once we finally refined our process to worked for us in addition to finding the tools that worked with our process. By starting with Scrum we were really forcing a process to the project when in reality we were really using the Waterfall method under the hood. Recognizing this half way through the process and adjusting our practices to be more closely in line with Scrum really helped us in our initial implementation push. Another thing that really aided us towards the end of the project was technology. We have looked at a number of systems to track work and to chat. We started with EasyBacklog and texting or emails. Our productivity was a little slow due to the overhead all of those technologies had associated with them. By the end of the project we had found GitHub's issue tracker and Slack to be very helpful as they worked well for all team members and integrated well into our existing process.

Scope was another thing we reflected upon pretty heavily. At the start of this project we definitely committed to far more work than we should have. Because of that everything always seemed rushed and we were spread too thin as a team. Looking back at the session application we really should have dropped the requirement for this project the second we started running into problems with it. We spent a lot of time trying to get that to integrate into the project and ended up having a solution that that was far from ideal but ate up a lot of our valuable time. Had we said no to a couple of the features or made them "stretch goals" we would be in a better spot today.

Overall as a team we think the project had its hiccups but ended well. The customers seem satisfied with what we are delivering here at the end, and while everything did not get done Trillium did not expect one senior project team to finish the whole project. this senior project has

been a rewarding experience for all of us and will likely be a great experience to have moving forward into the working world.

## References

None