Low Cost Patient and Healthcare Record System

Team B14

Final Deliverable Document

4/29/2017

Members:

Britt Ahlgrim, Sydney Barovsky, Ben Durette, JP Rivera, David Vitale

Advisor:

Dr. Lars Olson

TABLE OF CONTENTS

I. Executive Summary

The problem addressed by this project deals with an inefficient patient record keeping system in El Salvador. A distinct aspect of the health care system in El Salvador are small community clinics scattered all throughout the country with the goal of providing preventative services. Through this model, each person in a community is required to be seen a specific number of times either in a consultation by the doctor or in a home visit by a health promoter. This service is very important to the functioning of health care in El Salvador, however, the paper record keeping method for it is very inefficient.

The proposed solution was an electronic scheduling system with basic patient records. This system would allow health promotores to automatically see their upcoming appointments and other information for the day just by logging in. This system needs to be performance dynamic from a technical standpoint, since many health clinics do not have access to the internet or even electricity.

The generated solution is a lightweight web application to handle patient records and scheduling. By logging in, a user can view a patient list, a scheduling center, and a medical records tracking system. This software was designed to be secure, intuitive to non-technical users, and lightweight enough to run low cost, low performance computers.

Future steps start with communicating with our advisor Dr. Olson on the future scope of this project for the health clinics in El Salvador. Dr. Olson works with the Ministry of Health in El Salvador to develop technology for their use. The proposed future we have is to work with the engineers from the Ministry of health to see how best to deploy this product to a single health clinic in El Salvador. Also we plan on contacting the different doctors we worked with to communicate our progress and their final feedback.

II. Introduction

This document covers the entire year's worth of work towards starting to solve the issue of a lacking documentation system for healthcare records in El Salvador. This document describes the process of discovering the underlying problems and desires from the customers, identifying target specifications required of the end solution, design work of the system based on these specifications, verification, final implementation, and an analysis on risks and financial logistics for the problem. This year of problem solving and development was divided into phases of research, design, and implementation, and this document reflects that process timeline. The document was crafted based on previous documents that were developed and finalized at different points in the year, with the addition of other information that became more relevant towards the end of the project.

A. Project Background

In many Central and South American countries, the healthcare systems are not up to par with the rest of the world due to lacking infrastructure and economic support. Due to the smaller amount of money that can be spent on healthcare, tracking patient's information is done the most affordable way available. In countries like El Salvador, which we used as a base country because there was more information and access to information due to our project advisor, Dr. Olson, and group member, Ben Durette. The system for tracking patient information and appointment history is on half-sheets of paper that are updated each time the patient is seen. While this system does a decent job of recording information about the patient and visits, the filing of a system like this is labor intensive and time-consuming.

Apart from the human labor hours put into filing, an even bigger problem is that with the manual system, there is a high chance that a patient will not be seen as frequently as needed. The El Salvador health care system has scheduling policies around how often a patient should be seen based on his/her age, any risk factors, and any chronic illnesses s/he may be suffering from. Doctors and clinicians in El Salvador need to know when a patient should be seen, so that they can attend to that patient and record any new findings or developments. Currently, the system allows for many patients to 'fall through the cracks' because the workers in the clinics and hospitals are not able to sift through hundreds of sheets of paper to determine which patient to see next.

Our project addresses this issue by providing an electronic record-keeping system that stores information about each patient so doctors and clinicians can more easily pull up a patient's information and update it with new appointments, while being able to view the history of appointments as well. Not only does the system allow for better and easier visibility of patient information, but it also generates the next required appointment time based on the previous appointment and patient factors. The list of patients displays the next time a patient should be seen, which can be sorted in order, to provide maximum visibility for doctors and clinicians to view which patients are highest priority to be seen.

B. Project Objective Statement

Our goal is to design and develop affordable hardware and software solutions for electronic health record systems for locations in the developing world, such as clinics in El Salvador, which could include scheduling, queueing, and patient record information communicated to doctors and medical support staff using SMS communication in combination with an Android/iOS application. A working prototype is set to be completed around December 2016 and the final project is set to be completed by May 2017, while using a budget of $2000 (NIH Grant) and five team members.

III. Customer Needs

This first table is a record of one of the first contacts we had, a doctor from El Salvador who has had a wealth of experience in the health clinic's of El Salvador.

Table III.1: Responses from Dr. Diaz

Question	Responses	Customer Need
Thoughts on SMS text message to deal with the scheduling and queuing (which would be preferable)	The SMS messages can't just be a number and a name, they must include more information. Ideally, we would prefer a local network within the ECOS, and implement a cloud-based infrastructure later down the road.	● Use SMS or some other data transfer method that is widely accessible ● Include all required patient and appointment information
What is the current technology in the clinics?	Currently, everything is done by hand. This is a very good system but is not working because it is not an efficient way to keep up with the data. Each promotore needs to reference a hard copy of a chart to see the patient schedule and information. Patients with needs to be seen increase because they get pushed back on the schedule, or they are forgotten due to mistakes and oversights. Dr. Diaz would like to make computers or tablets accessible for the promotores. He said he would be able to get funding from the public health system for this. Upon starting usage of the system, there are 10,000 people whose data needs to be entered. Someone would need to input this information, unless we build some way to scan the old information into the system. Currently, nurses gather the vital signs, and the doctors enter them into the computer.	● More efficient and backed up system, to replace hard copy and paper ● Account for more accuracy and efficiency for scheduling and ensuring patient care ● Time permitting - allow for integration from other systems.

	This technology needs to be simple so that people are able to use it without too big of a learning curve. Technically all the patient information would need to be updated to the system. This could then be updated on a daily basis. The social security system does make use of electronic health records. Hospitals also have electronic health records within the hospitals. No electronic system is in use at the ECOS level. We want to build this technology for one ECOS, to then be expanded upon for multiple ECOS. In conclusion - nobody uses the same system, and there are systems within systems.	
How important is it for clinics to "talk"? Do people usually go to multiple clinics?	The clinics rarely have the need to talk to one another. ECOS is a team of clinics, which is the third level of the public health system, and each clinic has an area zone of patients. Patients can go to any clinic, but go to the clinic they 'belong' to because it is most convenient for them. If the ECOS can't treat a patient, the patient will get escalated to see either the first or second level. This would be helpful to share individual patient information between the different levels. Currently, this is done with a note from the promotore to the higher levels, which could continue to be the process for the time being.	● System with all of an individual clinic's patient information ● Information about the different levels of healthcare
How does Patient Confidentiality compare to the US?	The patient confidentiality regulations are not as strict as in the US, but they do exist in El Salvador. Physicians and nurses should have access to each patient's information. It would be good to limit the specific information that a	● Authorization system for viewing and modifying rights to nurses, doctors, and promotores

	nurse or promotore or doctor would need access to.	
Can you explain the Fichero System more in depth?	The fichero system doesn't talk about diseases, its sole purpose is to verify that people in the community have not been missed, especially that people with critical conditions are not being missed. The Fichero verifies that everyone is up to date and the clinic is doing a good job. The old fichero system is not working and is inefficient. Every person has a chart, one per person, which contains a schedule of when they were visited, as well as name, age, and other basic personal information. Promotores have potentially 3,000 sheets, in one month, and need to see 200 of them. As the promotores get backed up, they push the sheets to the back of the month. As a person comes in for a consultation, it is too difficult to look through the stack to find that person's file, so it doesn't get updated. If a promotore goes on a regular visit, the promotore refers the person to go to the clinic to get something checked. Should be able to put in the system that they were referred to go into the clinic, and can tell you if the person came in or not. This can also let the doctor know what specific day people should be coming into the clinic.	● Ensure that patients are being visited at the necessary frequency ● Uphold to standards enforced by Fichero system
What minimum functionality needs to be included?	The promotores, doctors, and nurses should be able to see what patients need to be checked within the next week. Depending on the classification, the system should know how often the person needs to be seen and when to see the patient next. It also should be	● Insert and update patient information ● Information and requirements for patient classifications ● Keep track of schedule for the patients and doctors

	able to edit patient information to update a patient's last visit or if a patient moves or dies or any other patient information changes.The system should also make the distinction of if a patient was seen by a doctor or by a promotore and if it was a home visit or a clinic visit. Ideally, this would account for only week days and be aware of holidays. Ideally, the system would require less typing, and more clicking input. Ideally, all of the information would be printed in the same format as the "fichero" document, because they need all the paperwork in hard copy for supervision.	
Who will pay for the data and internet usage?	Dr. Diaz is confident that he will be able to get buy-in and funding from the minister of health and the public health system.	● Affordable system
Where are we going to keep the server? Where should databases be stored? Would a SMS "cloud" based structure or a LAN based structure be preferred?	There are discussions about keeping a server in San Salvador or another urban area with access to internet and wifi, as well as security and higher quality infrastructure. In the future, we may look into making this a cloud-based system.	● A server of some sort, whether stored in the clinic, or cloud-based
Hardware: Would a keyboard attached to a tablet be useful, or unnecessary (for primatores)?	A tablet would be preferred for promotores, as they are always on the move. A tablet with keyboard or a full computer would be preferred for doctors and nurses as they are more stationary, but they could use a tablet as well.	● Tablet or smartphone or computer ● Ease of use
What is the bare minimum number of	The bare minimum information would be the patient's name, date of birth, sex, address, classification, and the last	● Patient date of birth, sex, address, classification, date of

fields for the database structure?	date of visit and visit information, and when the patient should be seen next. Other helpful information would be all personal information, information about relatives, medical history, the structure of their family, how many kids are in the household, etc.	last visit, deadline for next visit ● Information about classification system and scheduling restraints
Other information about Strategy/Logistics	Application may be the best, but we would have to figure out data later. We will implement in one ECO and see how it works, then we try to expand to metropolitan area, then the rest of the country. The best way would be to focus on the smartphone, make a really good product, and then sell the product. Possibly use smartphone app that could save info in the phone, have wifi in the ECO, once they get back to the ECO it could update the information.	

Table III.2: This table specifies particular customer needs, their rank and justification for addressing them in the design of the software application

Rank	Need	Justifications
1	The need would be to have an electronic file system with reminders to let the promotores/doctors know their daily patient schedule	This is the most basic description of our top level application: all technical details aside, this is what the end result should be
2	Affordable system	El Salvador is not a country that is predominantly wealthy: the cheaper the system is, the more likely the government will adapt the system in the promotores
3	Use SMS or some other data transfer method that is widely accessible	The "internet" isn't widespread in El Salvador: to keep things centralized and easily accessible, the most practical wireless communication should be used
4	Include minimum patient and appointment information - DOB, sex, address, classification, date of last visit, date of next visit	This information is needed to calculate the schedule, and it is the ultimate end information given to the promotores to help them on their patient meetings
5	Should be time efficient (intuitive to use)	Time is valuable; minimum training so learning curve is not so high - intuitive

		calendar system
6	Insert and update patient information	All existing paper records need to be added to the system, and new information needs to be able to be easily added in. The patient information should be able to update the patient information at anytime, whether it is part of the patient's scheduled, expected appointments or not.
7	Calculation of next visit date based on classification, last visit date	The system needs to be "smart": i.e., as the schedule is updated with recent visits, it should auto-update on when the next visit should be. The promotore should be able to override any scheduling input
8	Privacy system for viewing and modifying authorizations to nurses, doctors	Although not as strict as HIPAA compliancy laws in the U.S., the system will not be adopted if patient information is not encrypted or anonymized
9	Client side application to run on Android	Android tablets and phones are very cheap, and have a standardized operating system
10	A server for database and to run the system (Decide cloud-based or LAN-based in the future	We need to be able to store the data and run the system
11	Needs to be accessible anywhere; wherever you're accessing it, must be the same information	Promotores should be able to view the information on-site
12	Application needs to run on a basic feature phone (i.e. SMS based)	Are there enough people that don't have smartphones and we can't provide them
13	Providing tablets, phones, or computers	Is the hardware we're thinking of implementing this system on readily available? Can our grant funds be used to provide hardware? Will it get stolen"?
14	Application to run on Desktop	This would be more helpful for nurses and doctors to have a better work environment
15	Ability to save patient information for next day's visits offline on the device	If a promotore on a road visit wouldn't have internet: increases centralization of information
16	Need to show number of "home visits"; that need to be done that week	Helps promotores know how to allocate their week based off of how much home visits they want
17	Ability to update records/record information during a house visit	Automatically update the information while not at the clinic. This requires connections

		back to the clinic to provide real-time data updates and reads
18	Ability to easily input text about each section of patient records	Adding notes after-the-fact will provide better data integrity with more sufficient information about the patient information, even though it was not captured at the initial time of the appointment
19	System with all of an individual clinic's patient information	For a more advanced system, holding ALL patient information leads to more accurate data for analysis and better diagnoses
20	Information about the different levels of healthcare	Our system mainly deals with 1 promotore: it'd be nice to have it communicate with the rest of the El Salvador public health system. If possible, add a note if a patient is referred to a specialist, or seen at a hospital - another level of the healthcare system.
21	Allow for integration from other systems.	It would be useful to have as much digital information as possible: just maybe not realistic

IV. Target Specification

The target specifications for our system are listed below, in *Table 3: Target Specifications*, in no particular order.

Table IV.1 Target Specifications

Need #	Metrics	Marginal Value	Ideal Value
1	Number of Users	2	>100
2, 6, 10, 11, 17	On times/Off times	8 hours during weekdays	24 hours, 7 days a week
6, 9, 10, 12, 13, 14	Supported Devices	Android tablets	Any smartphone or tablet, laptops, basic feature phones
10, 11, 20, 21	Scalability	Working at one clinic	Working at all clinics in El Salvador
2	Cost of system	$2000	<$2000
3, 6, 10, 11, 15, 17,	Data Transfer	Works through	Works through

21		corded connections	corded connections, wifi, internet, and SMS
4, 7, 8, 10, 15, 18, 19	Patient Information	Stores minimum required patient information	Stores all patient information with HIPAA compliance
5, 6, 7, 12, 13, 14, 18	Usability	<1 day of training and setup required	<1 hour of training and setup required
1, 5, 7, 16, 18, 21	Efficiency	Keeps basic schedule of promotores' and doctors' appointments	Alerts doctors and promotores when a patient's appointment time has not been met

V. Problem Decomposition

To understand what functionality and features are needed to provide the most benefits and usefulness to the customer. The problem decomposition gives a description the gains and pains that were addressed in the project, as well as a description of who will be using the system and the final proposition.

Table V.1 Value Proposition. This table shows the Value Proposition as taken from the Value Proposition Canvas worksheet, listed in table form

Gain Creators - Prototype and tested User Interface and User Experience to optimize usability - SMS data connections to transfer data without internet - Information and calculations of patient severity and required next appointment	**Gains** - Small learning curve and process change for users - high rate of adoption - Ability to transfer data with a more accessible and cheaper means - More efficient and stable scheduling system to ensure patients are being seen - More reliable than current system
Pain Relievers - Save time finding people who need to be visited - Electronically organized system keeps all files sorted - Automation ensures no one falls	**Pains** - Time wasted sifting through paper files - Unorganized sorting of paper files based on who needs to be soon soonest

through cracks - Allows each person to be updated easily	- People *fall through cracks* when a their files get jumbled in the pile, their file may not be updated or viewed as soon as it should be
Products and Services - A scheduling system to manage patient visits and consultations - A patient healthcare and history record information system	**Customer Jobs** - Doctors - Nurses - Health Promoters - El Salvador Public Health System
Value Proposition Summary - Providing an electronic system that stores basic patient information and enhances the scheduling system for home visits and consultations	**Substitutes (Current Solutions)** - Manual, hand-written patient scheduling and visit information (currently used) - Expensive healthcare database solutions (used in developed countries)

VI. Documentation of Final Design

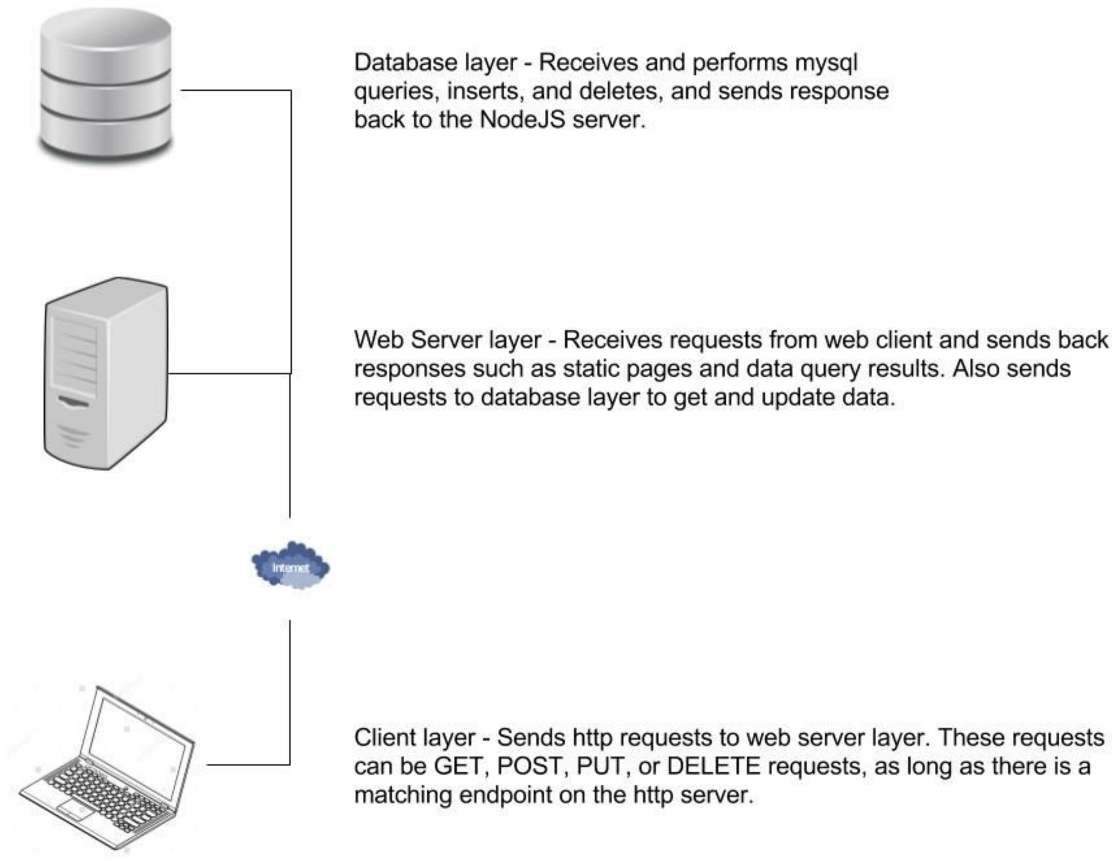
This system can be set up and distributed with any modern Linux distribution with a few external libraries installed.

A. Backend

This application employed NodeJS as the framework for the backend server. NodeJS was chosen because it is open-source, widely-used, and there are many third-party libraries. NodeJS is free to use with with well-documented tools and procedures. The fact that NodeJS is relatively new was appealing because at the rate that technology is advancing, going with older technologies could become outdated more quickly. Also, NodeJS offers many packages that can be integrated very easily into the application, such as express web framework, passport for authentication, and others. The full listing of packages can be found in Appendix A - Full listing of Packages Used on Backend.

The backend acts as an http server, taking requests from the client, and fulfilling the requests with static page responses, or data loaded from the MySQL server. The calls are made from the client-side via ajax calls. The backend also verifies authentication on each page other than the login and registration pages, to implement the security necessary for these pages. A diagram of the flow of requests and how the server responds to them is in Figure VI.A.1: NodeJS Server Design.

Figure VI.A.1: NodeJS Server Design



Database layer - Receives and performs mysql queries, inserts, and deletes, and sends response back to the NodeJS server.

Web Server layer - Receives requests from web client and sends back responses such as static pages and data query results. Also sends requests to database layer to get and update data.

Client layer - Sends http requests to web server layer. These requests can be GET, POST, PUT, or DELETE requests, as long as there is a matching endpoint on the http server.

In order for the NodeJS web server to function as needed, there are many third-party packages that are utilized on the web server layer. These are all described in table VI.A.1: List of NodeJS Packages, below.

Table VI.A.1: List of NodeJS Packages

Package Name	**Description**
Express	Express web framework is a very common package used in NodeJS frameworks to help with routing and using middleware. We found this package to be extremely helpful in speeding up the development process as much of the setup work for the http server was done for us by express.
Express-Session	Express-session is a tool used alongside express to keep track of a user’s session. Express-session does this by keeping track of cookies in

	the user’s browser and maintains a continuous session for each user. This package is especially necessary for authentication, as it will keep a user logged in for the duration of their session so that they can perform functions on each page without being redirected to the login page.
Cookie Parser	Cookie parser is also used with express-session to keep track of sessions. Because authentication and express-session rely on cookies, a cookie parser is needed to check the user’s session information.
Body Parser	The Body Parser package was utilized in the application to extract information from incoming requests. This is helpful for POST and DELETE requests that contain larger information for the requests such as parameters to search the database for.
Path	The Path package is simply a package to append to and manipulate packages on the local file system. This package was primarily used for declaring static file paths for the server to reduce the number of dynamic requests the server makes, and be able to quickly serve up static pages.
MySQL	The MySQL package is the provided package to connect to the database. There is not much to note about this, other than it allows us to open active connections to the database and query, insert, and delete against it.
Passport	Passport is the package used for authentication. Passport provides many different ways to authenticate users, but our application relies on local authentication. Local authentication allows us to maintain and manage users’ usernames, and store them securely.
Flash	Flash allows the application to send flash messages to the client screen without having to build that into the html and client javascript pages. The flash messages in our application are used for notifying the user of incorrect credentials or invalid registration username.

B. Frontend

The login page is the first screen the user would see when the open the application. The page contains fields for the user to enter their username and password to login into the system. Once the user enters their credentials in, and clicks on the login button they are redirected to the homepage. If the user does not have either of them they can click on the register button.

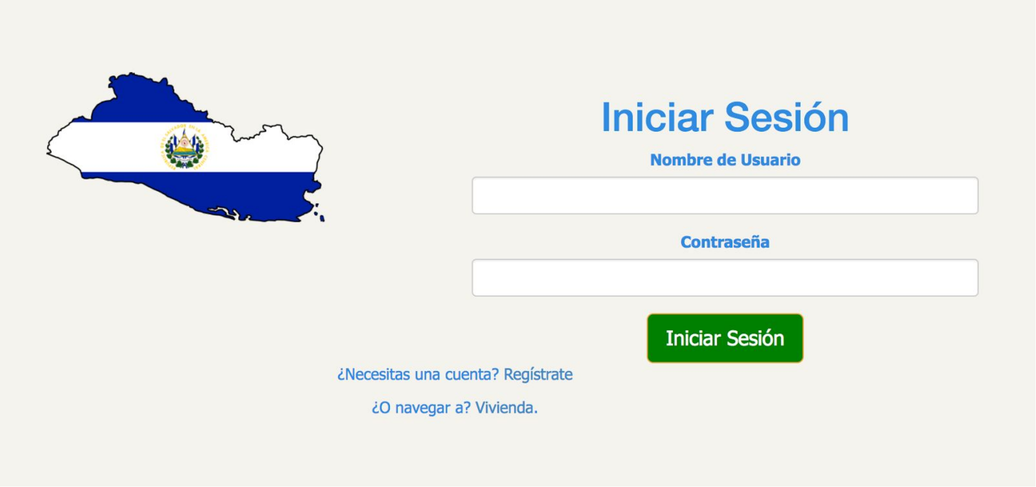


Figure VI.B.1 : Homepage

This is a screen capture of the login page

This is the registration page that the user can navigate to from the login page if they do not have a username and password. Once they enter in a username and password and hit register they are redirected to the homepage.

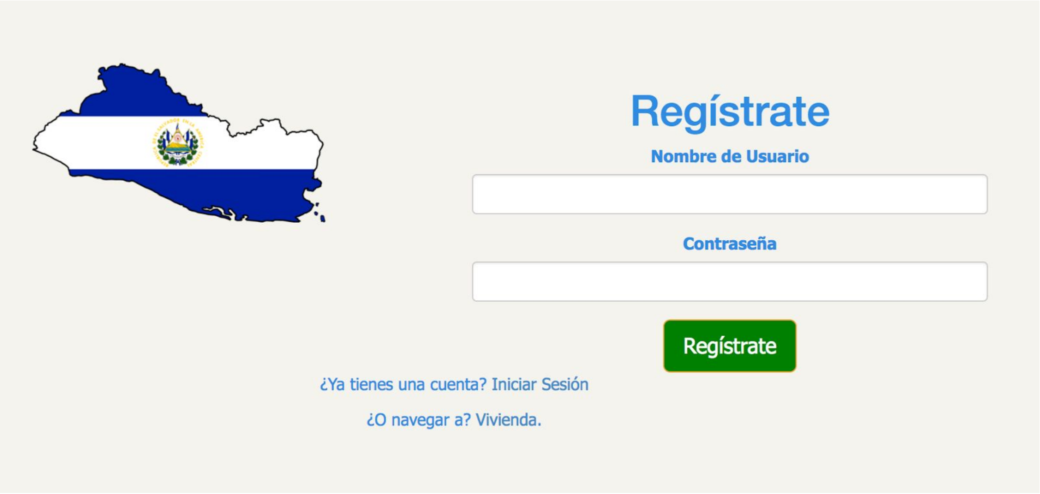
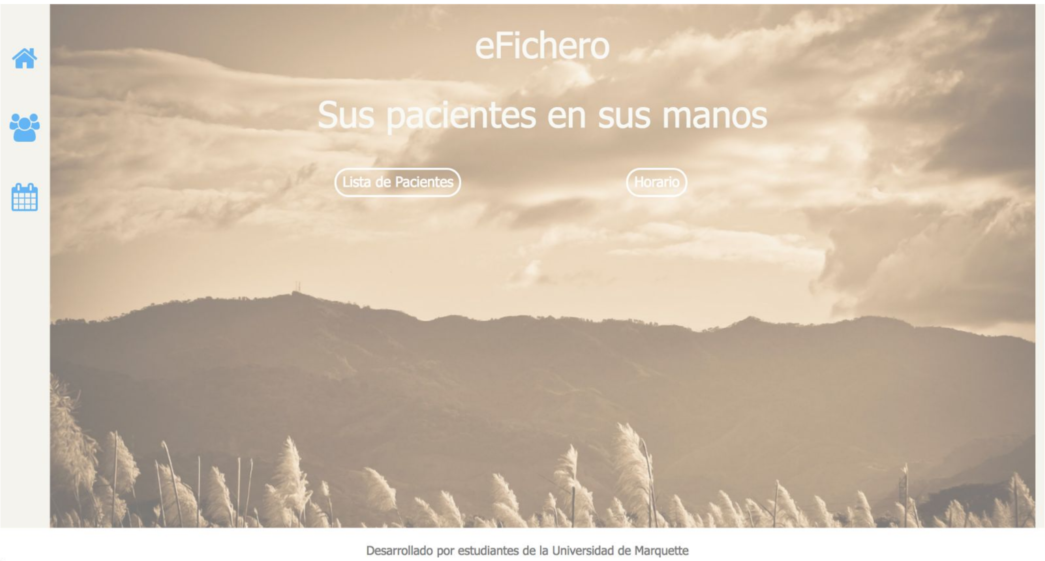


Figure VI.B.2: Register

This is a screen capture of the registration page

This is the homepage of the application, where the user can navigate to the list of patients, and the schedule of patients. The page is designed to be very simple and show the user, the different pages they can navigate to.

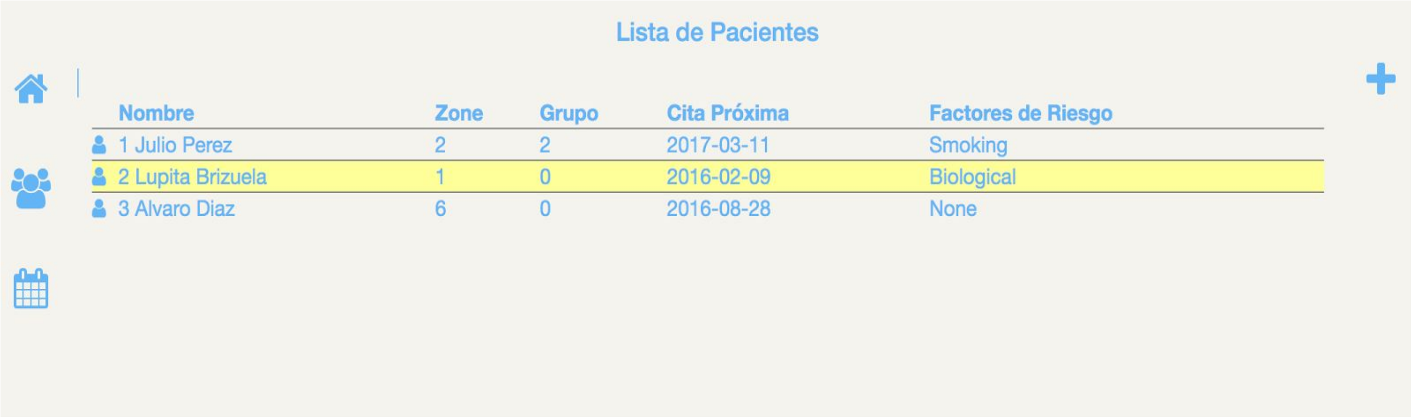


Desarrollado por estudiantes de la Universidad de Marquette

Figure VI.B.3: Homepage

Screen capture of the homepage

This screen shows the list of the patients currently in the system. The information on each is the name, the zone, the group, date of the appointment, and their associated risk factors. The table was made in a way that whenever the user is hovering their cursor over a particular patient the entire row of that patient is highlighted to denote selection. The user is also able to navigate back to the home page, and to the schedule using the icons on the navigation bar. The user is also able to filter each of the different columns by clicking on the name of it.



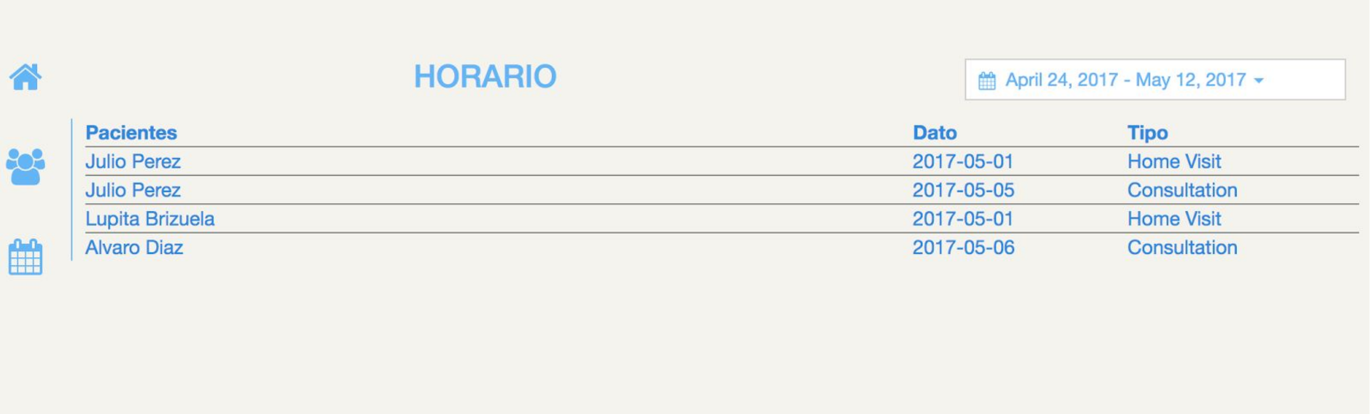
Lista de Pacientes

Nombre	Zone	Grupo	Cita Próxima	Factores de Riesgo
1 Julio Perez	2	2	2017-03-11	Smoking
2 Lupita Brizuela	1	0	2016-02-09	Biological
3 Alvaro Diaz	6	0	2016-08-28	None

Figure VI.B.4:List of patients

This is a screen capture of the list of patients

This is the schedule page which contains all of the scheduled patients within a certain date range. The information contained on this page is the patient name, the date, and the type of visit that is expected to occur. If the user would like to change the dates of viewing patients they would select the date drop down menu on the top right corner. The user can navigate back to the homepage, or the list of patients using the navigation bar on the left side of the screen.



HORARIO

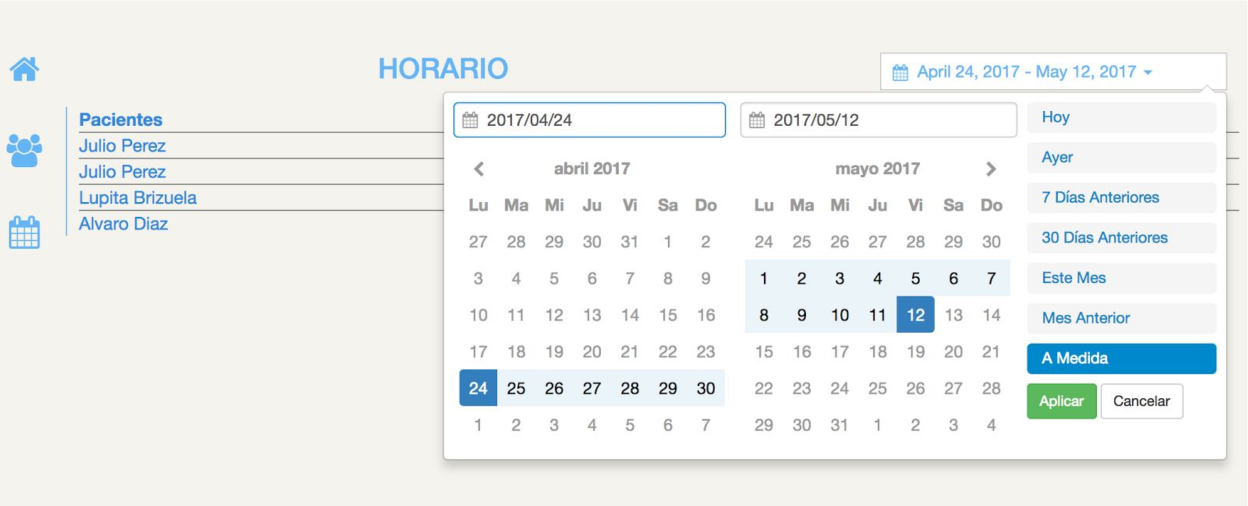
April 24, 2017 - May 12, 2017

Pacientes	Dato	Tipo
Julio Perez	2017-05-01	Home Visit
Julio Perez	2017-05-05	Consultation
Lupita Brizuela	2017-05-01	Home Visit
Alvaro Diaz	2017-05-06	Consultation

Figure VI.B.5: Schedule

This is the screenshot of the schedule page

This screen shows what the user sees when they click on the data selection drop down menu, and all the different options contained within. The different options for date viewing are today, yesterday, previous 7 days, previous 30 days, this month, and last month. There also is an option for the the date range, which is currently selected which allows the user to enter in a specific date range. This functionality is incredibly useful because it allows the user flexibility in viewing their data.



HORARIO

April 24, 2017 - May 12, 2017

Pacientes
Julio Perez
Julio Perez
Lupita Brizuela
Alvaro Diaz

2017/04/24

2017/05/12

abril 2017

Lu	Ma	Mi	Ju	Vi	Sa	Do
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

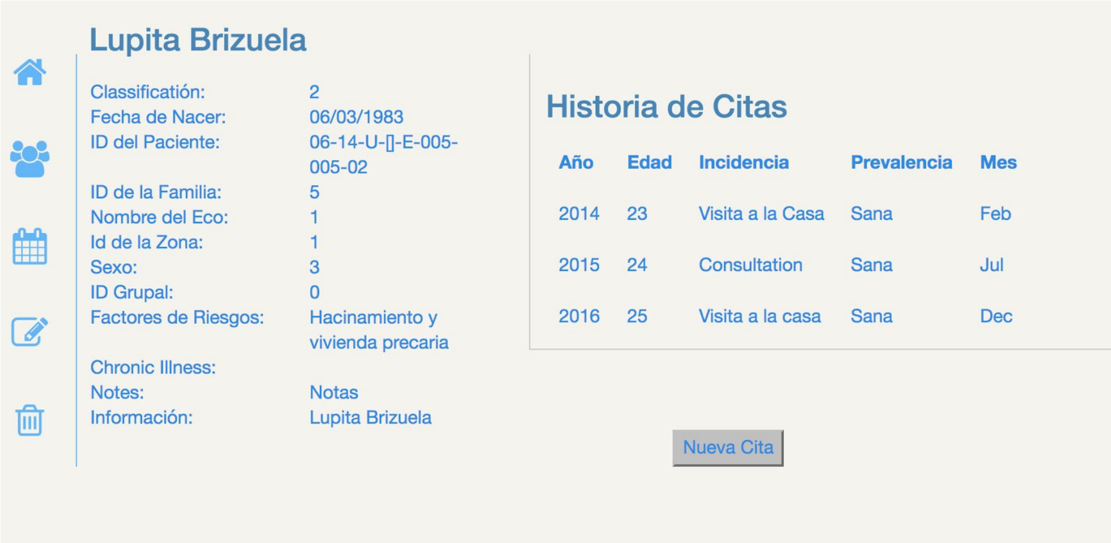
mayo 2017

Lu	Ma	Mi	Ju	Vi	Sa	Do
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Figure VI.B.6: Schedule page with date selection

This is a screen capture of the schedule with the calendar widget

This page shows the specific patient information displaying the patient classification, the risk factors, the sex, zone, information and the history of their past visits with doctors. The page was designed to only show the information that is absolutely essential for the user to have displayed to them. On the right side of the page, the history of the patient, which will expand as the history of the patient increases.



Lupita Brizuela

Classificatión:	2
Fecha de Nacer:	06/03/1983
ID del Paciente:	06-14-U-[]-E-005-005-02
ID de la Familia:	5
Nombre del Eco:	1
Id de la Zona:	1
Sexo:	3
ID Grupal:	0
Factores de Riesgos:	Hacinamiento y vivienda precaria
Chronic Illness:	
Notes:	Notas
Información:	Lupita Brizuela

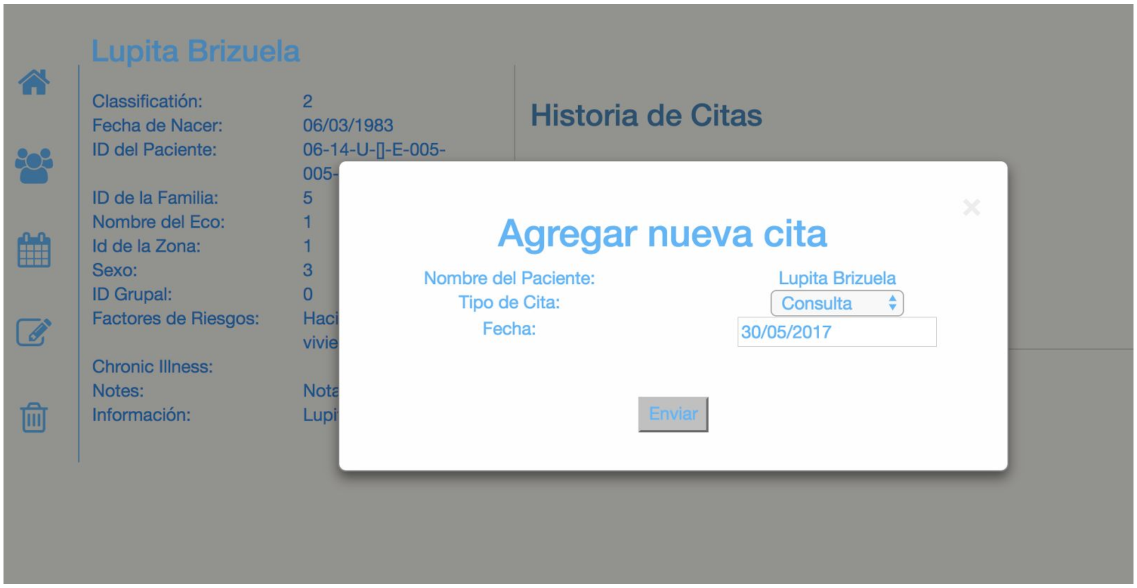
Historia de Citas

Año	Edad	Incidencia	Prevalencia	Mes
2014	23	Visita a la Casa	Sana	Feb
2015	24	Consultation	Sana	Jul
2016	25	Visita a la casa	Sana	Dec

Figure VI.B.7:Patient information

Screen capture of the patient information page

This screen shows the process of adding in a new patient appointment into the system, it asks for the appointment type, and the date when the user would like to set this appointment. Once all the correct information has been entered the user clicks submit to add a new patient appointment.



Lupita Brizuela

Classificatión:	2
Fecha de Nacer:	06/03/1983
ID del Paciente:	06-14-U-[]-E-005-005-
ID de la Familia:	5
Nombre del Eco:	1
Id de la Zona:	1
Sexo:	3
ID Grupal:	0
Factores de Riesgos:	Haci vivie
Chronic Illness:	
Notes:	Nota
Información:	Lupi

Historia de Citas

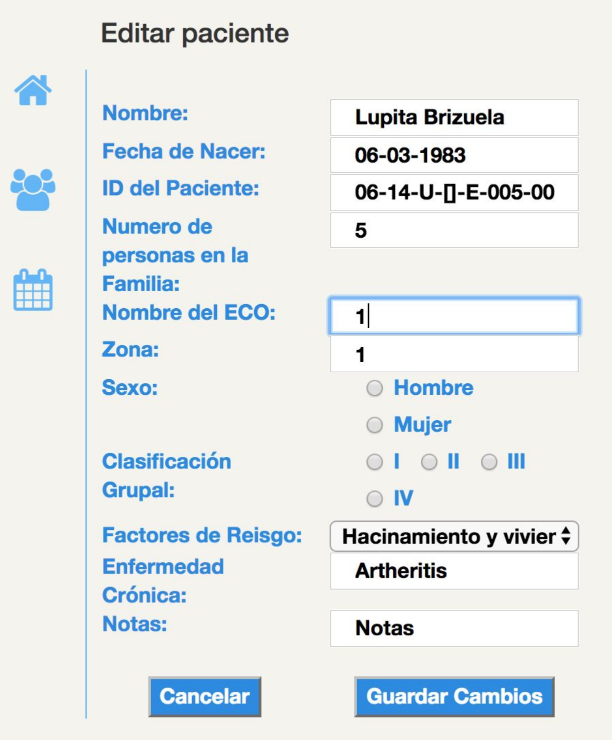
Agregar nueva cita

Nombre del Paciente:	Lupita Brizuela
Tipo de Cita:	Consulta
Fecha:	30/05/2017

Figure VI.B.8: Add new appointment

Screen capture of the add new appointment page

This page shows the features to edit an existing patient in the system. It displays all of the data that is currently in the system on a particular patient. The user is able to then save any changes that they have made



Editar paciente

Nombre: Lupita Brizuela

Fecha de Nacer: 06-03-1983

ID del Paciente: 06-14-U-▯-E-005-00

Numero de personas en la Familia: 5

Nombre del ECO: 1

Zona: 1

Sexo: ○ Hombre ○ Mujer

Clasificación Grupal: ○ I ○ II ○ III ○ IV

Factores de Reisgo: Hacinamiento y vivier

Enfermedad Crónica: Artheritis

Notas: Notas

Figure VI.B.9: Edit patient

This is a screen capture of editing the patient

C. Database

1. MySQL

Out the many open source database systems available for use (SQL, SQLite, PostgreSQL, etc.), MySQL was chosen for multiple reasons. For one, it is the most popular database used in industry, making it an industry standard. With such a short amount of time for the project and limited experience by the team with databases, it was logical to go with a database that didn't have steep learning curve, as MySQL does. The eFichero is currently being made for only one clinic, but it has the scope of being expanded for a whole network of clinics. This reason made it important to pick a database that was easy to scale, which also led to MySQL.

2. Database Architecture

The database is made up of a series of tables that are connected and store all of the necessary information for the patient records. In the following tables, each column corresponds to a table in the database and its attributes. For further details on the database structure, see the appendix.

Table VI.C.1: The first five tables in the database and their corresponding attributes.

Appointment	**ECO**	**Family**	**Group**	**Next Appointment**
- Patient ID - Appointment ID - Appointment Date - Appointment Type	- ID - Name - Zones	- ID - Name - Head Person of Family ID	- ID - Name - Description	- Group ID - Next Appointment Date

Table VI.C.2: The second five tables in the database and their corresponding attributes.

Patients	**Sex**	**Visit History**	**Zone**	**Authentification**
- ID - Name - Date of Birth - Patient ID - Family ID - ECO Name - Zone ID - Sex - Group ID - Risk Factor - Chronic Illness - Notes	- ID - Name	- Patient ID - Age - Incident - Prevalence - Deceased - Date of Visit	- ID - Name	- ID - Username - Password

The database is able to store patients, their basic health information, and their appointment history. Based on the appointment history, the application is able to perform its primary function of determining the next appointment for each patient.

This database is able to work dynamically with the web application, allowing the user to both load, save, and delete content. For example, when the user opens the patient list, the web application makes a call to the database to load all the elements stored in the ‘Patients’ table. When the use clicks on a specific patient, another call is made to the database to request the specific information of the patient. From there, the user is able to edit the information of the specific patient and save it or delete the patient altogether. From the patient list page, the user can create entirely new patients as well. In addition to adding, editing and deleting patients, the

the database is also used to store appointments for each patient. On each patient's individual patient info page, new appointments are able to be saved into the database. The web application then uses these recent appointment dates to calculate when the patient needs ot be visited next.

The database is stored on the same server as the web application itself. A long term goal would be to have separate servers for the database and hosting the web application, but given the nature of this short term project in its first year, they are both currently hosted on the same server.

D. Raspberry Pi

The idea behind the Raspberry Pi is to act as an access point for the system. Ideally, it works in three parts: a wireless access point, host/server, and captive portal. First, the Pi would act as a wireless access point, to which users could connect via the WiFi. Second, the Pi would be able to host the website locally. Third, the Pi would allow access to the website as a captive portal, so that the users would only be able to access the hosted website. This implementation is ideal for the rural clinics as it is a small, inexpensive device that would allow for website access within the clinic as needed.

Currently, the website is being hosted on a cloud server, and the Raspberry Pi is acting as an access point to allow for access to the cloud server.

E. Other ways to deliver the system

E.1.1: Node.js

Node.js is very a common web development framework that is supported on many different platforms. Anyone with a technical background in web development could get this project up and running with minimal effort on Linux, Windows, or MacOS. Because these operating systems can run on various processor architectures, this project becomes very distributable.

E.1.2: Linux used as a server

Linux is a very commonly used server platform. Since it is free to use, widely used, and very configurable in regards to security, it is a great choice of a platform. There exist Linux bash scripts in our codebase repository to help set up a deployed instance, as well as a very large user community and available examples.

E.1.3: Packing up a Linux Operating System image

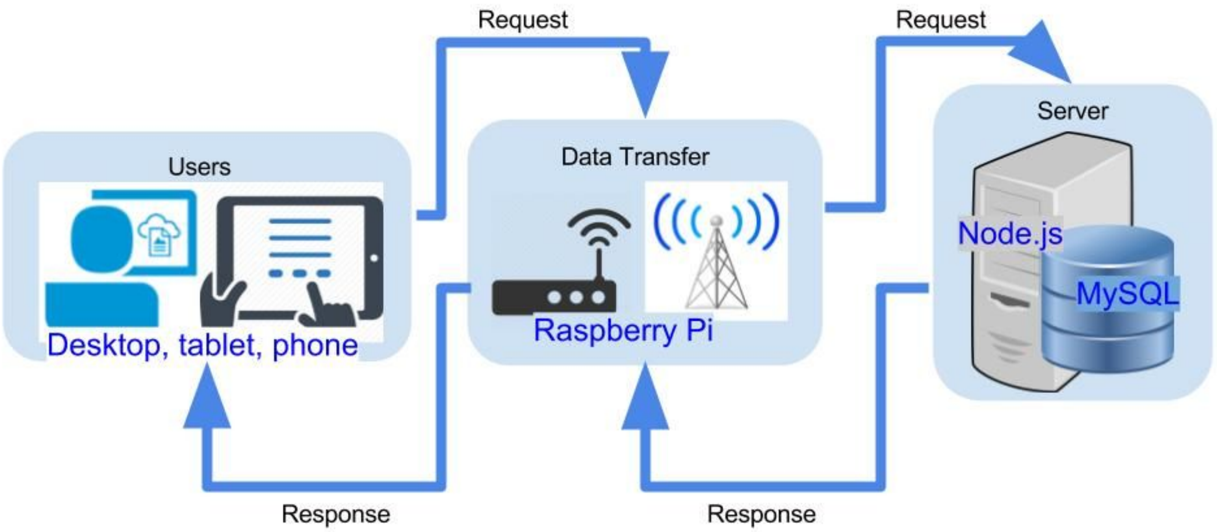
It is a common IT practice to create an Operating System Image for distribution purposes. Installing the default operating system does not often include all the necessary software needed for various purposes. In our case, we would take a standard Ubuntu Linux installation, and add Node.js, NPM, MySQL, all of our NPM installation packages, and the final stable version of our software. We would then add a script to run on startup to have our web instance run continuously This image could then be run on any locally hosted computer or cloud computer: after installing it, our web server would just start.

E1.4: Local hosted servers

Since much of El Salvador does not have connection to the centralized internet, having a locally hosted server would be a valuable way to distribute this system. This would require someone with a semi-technical background to find a host-server computer, and a wireless router. That person would use the image described in Section E1.3 and install it on the host computer. Since this image would already be setup to have everything configured correctly, anyone could see the web application by connecting to the wireless router. This could be a valuable solution for clinics that are not connected to the centralized internet.

Figure VI.E.1.4: Locally Hosted Server Setup

This figure provides a visualization of how a locally hosted server setup might look.



E1.5: Cloud hosted servers

Some of the major cities in El Salvador do have connection to the main internet, and therefore local cloud hosting services. In the U.S., the majority of all enterprise applications are cloud hosted with systems such as Amazon Web Services, Microsoft Azure, etc. This project did briefly host the final version of the application on Amazon Web Services. Server space was purchased, and SSH access was allowed. The system was then logged into, and all software was set up. HTTP and HTTPS ports were granted incoming access to all IP addresses, and other security measures were taken to prevent misuse of the data. The El Salvadoran government does manage a set of servers, and there are other means to connect to the centralized El Salvadoran internet.

VII. Verification

A. Introduction

The following section contains the experimentation verification process of each of the different target specifications that were gathered in the beginning of the project, along with their respective results. In this section, the application will be referred to as the eFichero application for simplicity.

B. Original Target Specifications

The original target specifications were identified and assigned values based on the customer needs gathering process. Table VII.B.1, below, shows the set of target specifications determined in the beginning stages of the project timeline.

Table VII.B.1: Target Specifications

The Target Specification table shows the set target specifications for the eFichero system. Each of these metrics are broken down into more specific, individual experiments in Section C.

#	Metrics	Units	Marginal Value	Ideal Value
1	Number of Users	Number of users	2	>100
2	On times/Off times	Time (Hours)	8 hours during weekdays	24 hours, 7 days a week
3	Supported Devices		Android tablets	Any smartphone or tablet, laptops, basic feature phones
4	Scalability		Working at one clinic	Working at all clinics in El Salvador
5	Cost of system	Currency (dollars)	$2000	<$2000
6	Data Transfer	Binary	Works through corded connections	Works through corded connections, wifi, internet, and SMS
7	Patient Information	Binary	Stores minimum required patient information	Stores all patient information with HIPAA compliance
8	Usability	Time (Days)	<1 day of training	<1 hour of training and

			and setup required	setup required
9	Efficiency	Binary	Keeps basic schedule of promotores' and doctors' appointments	Alerts doctors and promotores when a patient's appointment time has not been met

C. Experiments

A number of experiments were conducted to determine meeting the requirement of the set target specifications. This section goes into detail the experiments performed and resulting conclusions for each target specification. This section is broken down first by target specification, then by experiment.

C.1: Number of Users

C.1.1: Experiment 1 - Number of Users

The purpose of this experiment is to validate the number of users that can access the system and have login credentials for the system at any given time.

C.1.1.1: Experimental Procedure

1. Run the build script on a local desktop which runs the node.js application
2. Open two different browsers
3. Sign up two different users and verify the application operates as expected
4. Close the application

C.1.1.2: Equipment

1. Personal laptop
2. The code base for the eFichero application

C.1.1.3: Results

The results of the experiment verify that there can be two users on application at the same time. There was no identified maximum limit to the number of users that can be logged into the application at the same time.

C.1.1.4: Interpretation of Results

This indicates that two different users can use the application concurrently, this meets the marginal user expectation. Further, within reason, any number of users can be registered and logged onto the system at the same time.

C.1.1.5: Conclusion

The eFichero application can support at least two different users simultaneously, which was the marginal number of users in the target specifications. A potential benefit would be to further test the number of potential users on the system.

C.2: On Times/Off Times

C.2.1: Experiment 1 - Access Server Throughout Whole Day

The objective of this experiment is to ensure that the application is accessible at any time of day, testing the reliability of the system.

C.2.1.1: Experimental Procedure

1. Run server hosting eFichero application
2. Use desktop or tablet to access application and data on database throughout various points of a day, and various days of a week

C.2.1.2: Equipment

1. Tablet
2. Server

C.2.1.3: Results

No results, have not been able to test.

C.2.1.4: Interpretation of Results

Current implementations of eFichero are only run on a local computer or a remote, cloud based server. These are sufficient for testing functionality of application, but are not the end goal of running the application from a local server run on a Raspberry Pi. The goal of this experiment is to verify the reliability of the Raspberry Pi as a server and its ability to allow access to the application at any point of the day or week. We currently do not have this server configured and running, so we are not able to test the accessibility of the application.

C.2.1.5: Conclusion

Experiment fails because it is not able to be tested.

C.3: Supported Devices

C.3.1: Experiment 1 - Hosted on Different Operating Systems

The objective of this experiment is to determine the operating systems on which the web application will perform as anticipated.

C.3.1.1: Experimental Procedure

1. Determine the operating systems to test on
2. Run the application on each of those operating systems
3. Verify that each of the pages is accessible on each of the operating systems.

C.3.1.2: Equipment

1. Windows Machine
2. MacOS Machine
3. Linux Machine

C.3.1.3: Results

Table VII.C.3.1.3.1 shows the results from the experiment. The table is organized by each machine, and then by each view. A view is said to have 'passed' on a given machine if it renders in the expected format, and failed if there is anything wrong with the machine.

Table VII.C.3.1.3.1: Experimental Results

The Experimental Results table contains the experimental results as listed by machine type tested on and each of the application pages. Each of these was measured on a 'Passed'/'Failed' scale.

MachineType	Login View	Signup View	Home View	Patient List View	Schedule View
Windows	Passed	Passed	Passed	Passed	Passed
MacOS	Passed	Passed	Passed	Passed	Passed
Linux	Passed	Passed	Passed	Passed	Passed

C.3.1.4: Interpretation of Results

These results demonstrate that the operating machine on which the application is hosted does not impact the application. All of the endpoints for the site were all able to be routed and accessed from each of the machines that they were tested on.

C.3.1.5: Conclusion

Because all of the machines are able to support the system, this proves that the hardware chosen to implement the system on will not have a great effect on how the program will run. This is a positive response as it allows for more flexibility and changes between the different clinics and our test system versus a production system. Knowing that the operating system does not affect how the node.js framework acts allows for less restrictions with hardware overall.

C.3.2: Experiment 2 - Different Browsers

This experiment will test the capabilities of various commonly used browsers to ensure that there is flexibility in the users' choice of browser and the site will still behave as intended.

C.3.2.1: Experimental Procedure

1. Determine the browsers and versions that are being tested for the application.
2. Use BrowserStack.com to verify the page loads and renders as expected
3. Make note of differences between expected and actual renderings of the page.

C.3.2.2: Equipment

1. BrowserStack.com
2. Hosting machine for application

C.3.2.3: Results

Table VII.C.3.2.3.1 displays the results of each of the pages. If the page loads as expected, it is represented with the text, 'Pass'; if there are differences between what is expected, there are notes in the table describing those differences.

Table VII.C.3.2.3.1: Experimental Results

The Experimental Results table contains the results from experiment analyzing each of the behaviors of the browsers. A result is marked with a 'Passed' if the results were as expected, and a 'Failed' otherwise. If the results were not what was expected, the footnotes at this table explain the differences.

Browser	Login page	Signup Page	Home Page	Patient List	Schedule	Patient information
Chrome 57	Pass	Pass	Pass	Pass	Pass	Pass
Chrome 38	Pass	Pass	Pass	Pass	Pass	Pass
IE11	Pass	Pass	Pass	Pass	Pass	Pass
Firefox 52	Pass	Pass	Pass	Pass	Pass	Pass
Firefox 51	Pass	Pass	Pass	Pass	Pass	Pass
Firefox 50	Pass	Pass	Pass	Pass	Pass	Pass
Edge 14	Pass	Pass	Pass	Pass	Pass	Pass
Edge 13	Pass	Pass	Pass	Pass	Pass	Pass

C.3.2.4: Interpretation of Results

These results show that all of the current versions of the most used browsers are compatible with the application.

C.3.2.5: Conclusion

These results demonstrate that the devices that are using the systems have the modern versions of the browsers installed. These results come from an additional period of development and testing after it was seen from previous experiments that the UI design of the site was not compatible with all expected browsers. The results demonstrate that all functionality will be available for users on a wide range of devices.

C.3.3: Experiment 3 - Different Devices

This experiment tests various commonly used devices to ensure that the application being used is compatible on multiple hardware devices.

C.3.3.1: Experimental Procedure

1. Determine the devices that would use the application.
2. Use BrowserStack.com to verify the page loads and renders as expected on the different devices
3. Make note of differences between expected and actual renderings of the page.

C.3.3.2: Equipment

1. BrowserStack.com
2. Hosting machine for application

C.3.3.3: Results

Table C.3.3.3.1 displays the results of each of the pages. If the page loads as expected, it has ‘Pass’; if there are differences between what is expected, there are notes in the table describing those differences.

Table VII.C.3.3.3.1: Experimental Results

The Experimental Results table contains the results from experiment analyzing each of the behaviors of the browsers. A result is marked with a ‘Pass’ if the results were as expected, and a ‘Fail’ otherwise. If the results were not what was expected, the footnotes at this table explain the differences.

Browser	Login page	Signup Page	Home Page	Patient List	Schedule	Patient information
Windows	Pass	Pass	Pass	Pass	Pass	Pass
MacOS	Pass	Pass	Pass	Pass	Pass	Pass

Linux	Pass	Pass	Pass	Pass	Pass	Pass
Android	Pass	Pass	Pass	Pass	Pass	Fail[1]
iOS	Pass	Pass	Pass	Pass	Pass	Fail[1]

1. Fail - The information overlaps each other making it unreadable on a smaller screen.

C.3.3.4: Interpretation of Results

These results demonstrate that the functionality of the site will work on a variety of devices' browsers.

C.3.3.5: Conclusion

There was evidence that the UI was not as strong as it could have been because the responsive design of the UI is not as dynamic as it should be for smaller screen sizes.This limits the flexibility for our customers, so given time, we will work on fixing the UI so that it is fully functional for more browsers. However, it is to be noted that the functionality that is not working across all browsers is not part of the core functionality, and the users could still complete their minimum necessary functions without it.

C.4: Scalability

C.4.1: Experiment 1 - Number of Users

The objective of this experiment is to get the single server hosted web application be successfully accessed by five people.

C.4.1.1: Experimental Procedure

1. Set up the web application on a single server/computer
2. Connect a router to the server/computer
3. Get 5 people to connect to the device and interact with the web application
4. Assert that everything works as expected (no crashes)

C.4.1.2: Equipment

1. A single laptop/Linux computer
2. Router
3. 5 People and their personal computers/cell phones
4. Tablet

C.4.1.3: Results

After five people connected to the server with various devices, the web application worked as expected. There were no crashes, no significant slowdown. Everyone's web experience was unaffected.

C.4.1.4: Interpretation of Results

Using a standard set box machine as a web server, five people can use this web application with success. Any number of people under five can use this web application with success. A computer with less RAM/lower processing power under a less of a load could also handle a load of five or less people at a slower pace.

C.4.1.5: Conclusion

This type of setup could be used for a small clinic: one dedicated computer web server with up to five people connected to the server could be technically feasible. Since, in practicality, a lesser computer power system with less people connected would be in use, this system would be a success if implemented.

C.4.2: Experiment 2 - Cloud Stress Test

The purpose of this experiment is to stress test the web application against many of the challenges of large scale cloud deployed web applications

C.4.2.1: Experimental Procedure

1. Set up the web application using cloud9 or AWS cloud hosting
2. Get a personal computer to run automated HTTP get and HTTP post requests script
3. Use automated script to see if web application still works as expected (i.e. does our web application scale up to cloud deployment?)

C.4.2.2: Equipment

1. A single personal computer
2. Cloud9 / AWS account
3. Automated testing script

C.4.2.3: Results

After deploying to cloud9, testing scripts were run on the hosted instance. Doing many HTTP requests a second, the web server took advantage of dynamic cloud9 RAM, successfully scaling and not slowing down response time significantly.

C.4.2.4: Interpretation of Results

Using modern cloud hosting technology, this application is efficiently written to scale properly. Although further investigation would need to be done to see what number of users could actually use this web application at a time, a number in the hundreds could be very plausible if successfully integrated using cloud hosting technology.

C.4.2.5: Conclusion

This type of web application could be used on a larger scale, cloud hosted for a hospital or larger health clinic with a magnitude in the 100s of unique users. Cloud hosting would need to be found that is accessible in El Salvador, but this is technically feasible.

C.5: Cost of System

C.5.1: Experiment 1 - Budget Review

The objective of this experiment is to ensure that the cost of a full system for a single clinic does not exceed that of our budget.

C.5.1.1: Experimental Procedure

1. Open an Excel Spreadsheet.
2. Construct a table that contains: Order Status (To Be Ordered, Ordered, Delivered, etc), Category (Hardware, Software, Testing), Item, and Cost.
    a. NOTE: components used to aid in construction, but not included in the final system design should be included.
3. Calculate the overall cost of the system based on the cost of the individual parts.
4. Compare the overall cost of the system to the given budget.

C.5.1.2: Equipment

1. Excel Spreadsheet

C.5.1.3: Results

The results from this experiment are listed in Table VII.C.5.1.3.1: Overall System Cost, below. The table is a visual representation of the cost breakdown for each item purchased. Also shown is Table VII.C.5.1.3.2: Budget Comparison, demonstrating the percentage of the provided budget that was used for this project.

Table VII.C.5.1.3.1: Overall System Cost

The Overall System Cost table contains the data necessary to determine the cost of constructing and operating the eFichero system. For simplicity, it only contains the items that have been ordered and items to be ordered (TBO).

STATUS	**CATEGORY**	**ITEM**		**COST**
Delivered	Software	Wifi Router	LINKSYS E1200	$28.45
Delivered	Testing	Tablet	Samsung Galaxy Tab	$149.95
Delivered	Hardware	Linux Server	Raspberry Pi 3	$39.95

Delivered	Hardware	Linux Server	Raspberry Pi Enclosure	$7.95
Delivered	Hardware	Linux Server	Power Cable	$7.50
Delivered	Hardware	Linux Server	SD Card w/ Raspian	$11.95
Delivered	Hardware	Linux Server	Ethernet Cable	$2.95
Delivered	Hardware	Linux Server	Wifi Adapter	$19.95
Delivered	Hardware	Linux Server	External Hard Drive (1TB)	$55.99
TBO	Hardware	Battery		TBD
TBO	Hardware	Solar Panels		TBD
TOTAL				**$324.64**

Table VII.C.5.1.3.2: Budget Comparison

The Budget Comparison table shows a comparison between the Target Specification values (Marginal; Ideal) and the Actual Value as determined in Table VII.C.5.1.3.1. Values are compared on a Pass/Fail basis rather than nominally.

	Marginal Value	Ideal Value	Actual Value	Pass/Fail?
Cost of System	$2000.00	<$2000.00	$324.64	**Pass**

C.5.1.4: Interpretation of Results

Table VII.C.5.1.3.1 shows the cost of the individual components and their respective costs. Those individual costs are added up to reach the *current* cost of the system. For simplicity, only components that have been ordered/delivered, and components listed as 'to be ordered (TBO)' were listed. Cost for the Battery and Solar Panels was not listed as the need and specific required items has not yet been determined. If additional costs were to arise, the cost of the system would increase accordingly.

Table VII.C.5.1.3.2 compares the current, actual cost of the system to the marginal and ideal values as required by target specifications. The marginal and ideal values are $2000.00 and less than $2000.00, respectively. Comparatively, the actual value is $324.64, which is $1675.36 under budget.

C.5.1.5: Conclusion

Comparing the (above) results, and the Cost of System desired values, we find that the experiment successfully passes. As of now, the system costs $324.64; however, it is possible that additional components could be required and ordered as needed. Marginal/ideal values, as required by the target specification, are $2000 and less than $2000, respectively. Since the results are analyzed on a pass/fail basis, and $324.64 < $2000.00, it can be determined that the cost of the system exceeds the ideal value and as such passes. If the system cost were to increase, due to additional required components/cost, the results would need to be tested again; however, it is unlikely that we will exceed the ideal value for cost given the current progress of the system.

C.6: Data Transfer

C.6.1: Experiment 1 - Data Transfer Methods

The objective of this experiment is to verify the various methods of transferring data with eFichero.

C.6.1.1: Experimental Procedure

WIFI/Internet

1. Setup remote cloud base server, run eFichero
2. Load application, run various functionality

Hardwire

1. Run eFichero on tablet
2. Connect tablet to local area network server (Raspberry Pi)
3. Save data

SMS

1. Not applicable, out of scope

C.6.1.2: Equipment

1. Tablet
2. Raspberry Pi

C.6.1.3: Results

The results from this experiment are listed in Table VII.C.6.1.3.1: Experimental Results, below.

Table VII.C.6.1.3.1: Experimental Results

The Experimental Results table contains the results for testing on different data transfer method. The data transfer method is marked with Pass/Fail notation as expected. If a method failed, or the experiment could not be performed, notes are listed in footnotes of the table.

Data Transfer Method	Pass/Fail?
Wifi	Pass
Hardwire	Fail[1]

SMS	N/A[2]

[1] Not yet tested

[2] Out of Scope

C.6.1.4: Interpretation of Results

Primary mode of data transfer used with eFichero is through a WiFi internet connection. When running the application from a server, the data is able to be access through the internet. This has been tested both on a local computer, running as it's own server, and running the application on a remote, cloud based server. Both tests have verified that the application is able to be run and the database is able to be accessed. However, these have both been tested by running the eFichero from a personal computer of cloud base server, which are different then our ultimate source, which will be a local area network server, which is currently being set up.

The hardwire data transfer implementation has not been able to be tested yet, so it currently fails. This implementation is dependent on the local server made from the Raspberry Pi to be running, which it currently isn't. Once this server is running, the hardwire data transfer method will be tested.

In our original scope, we thought there would be use in transferring data through SMS data format (through cellular towers). Throughout additional conversations with our contacts, we have narrowed down the scope to not needing to transfer data through SMS, so this is now out of scope.

C.6.1.5: Conclusion

Overall, this test both passes and fails. SMS data transfer is out of scope, so that is irrelevant. The data is able to be transferred through a WIFI internet connection, which is the primary mode of data transfer. However, since the actual server has not been setup, we cannot yet test the data transfer in its entirety. Once the local server is setup, we will be able to test data transfer to the actual local server through WIFI and with a hardwire.

C.7: Patient Information

C.7.1: Experiment 1 - Each patient contains required information

The objective of this experiment is to verify that each patient in the eFichero database contains the required data.

C.7.1.1: Experimental Procedure

1. Open eFichero application
2. Login to the MySQL database through computer terminal
3. Open Patients table
4. For a current patient saved to the database, verify each of the required data fields is populated

C.7.1.2: Equipment

1. Hardware to run the application such as a personal computer or a tablet
2. The code base for the eFichero application

C.7.1.3: Results

The results from the Patient Information experiment are listed in Table VII.C.7.1.3.1, below.

Table VII.C.7.1.3.1: Experimental Results

The Experimental Results table contains the result from eFichero patient data collection

Data Field	Present?
Patient Name	Yes
Date of Birth	Yes
Patient ID	Yes
Family #	Yes
ECO Name	Yes
Zone #	Yes
Sex	Yes
Group Classification	Yes
Risk Factor	Yes
Chronic Illness	Yes

C.7.1.4: Interpretation of Results

A patient was loaded in the database. The ten data fields in Table VII.C.7.1.3.1 are the required data fields for each patient. Each data field was verified to be present for the patient in the database.

C.7.1.5: Conclusion

Test passes, all required data fields for a patient are present in database.

C.7.2: Experiment 2 - Patient storage is HIPAA compliant

The objective of this experiment is to verify that each patient in the eFichero database is compliant with federal HIPAA (Health Insurance Portability and Accountability Act of 1996) standards.

C.7.2.1: Experimental Procedure

1. Compare eFichero application with each aspect of the HIPAA compliance checklist

C.7.2.2: Equipment

No equipment needed for this test

C.7.2.3: Results

Table VII.C.7.2.3.1: Experimental Results

The Experimental Results table contains various tests about HIPAA compliancy for the eFichero application. If an item on HIPAA compliance is met, it is marked with 'Pass', otherwise, it is marked with 'Fail'

	HIPAA Checklist Items	Decription	Pass/Fail
1	Transport Encryption	Is always encrypted as it is transmitted over the Internet	Fail
2	Backup	Is never lost, i.e. should be backed up and can be recovered	Fail
3	Authorization	Is only accessible by authorized personnel using unique, audited access controls	Pass
4	Integrity	Is not tampered with or altered	Fail
5	Storage Encryption	Should be encrypted when it is being stored or archived	Fail
6	Disposal	Can be permanently disposed of when no longer needed	Pass
7	Omnibus/HITECH	Is located on the web servers of a company with whom you have a HIPAA Business Associate Agreement (or it is hosted in house and those servers are properly secured per the HIPAA security rule requirements).	Fail

C.7.2.4: Interpretation of Results

The results from the experiment demonstrate that the only passing HIPAA standards are the requirement of Authorization, and the permanent disposal of information. The other metrics that are defined as necessary for HIPAA compliance in a web application were not met at this point in the project.

C.7.2.5: Conclusion

Overall, eFichero is not HIPAA compliant. Some of these aspects are not currently implemented, but certain could be added in the future. For example, transport encryption, storage encryption and having a backup could all be implemented a future version. However, at this stage, eFichero does not comply with HIPAA standards. In the future we need to reach out to our contacts and decide what the best course of action would be moving forward.

C.8: Usability

C.8.1: Experiment 1 - Creating a new user

This experiment tests the ability to create a new user for the application.

C.8.1.1: Experimental Procedure

1. User navigates to the patient creation page
2. The user inputs the information about the patient into the different text fields
3. Click on the submit button to send the data to the mysql server, which will now be displayed on the patient list html page
4. Navigate to the patient list html page and verify that the previously input user appears on the webpage
5. Close the application

C.8.1.2: Equipment

1. Personal laptop
2. Code base for the eFichero application

C.8.1.3: Results

The current system has a webpage that allows the user to input the data they need about a patient and a submit button that can be clicked. However, when the button is clicked the patient information is currently not being saved to the mysql server or being displayed on the patient list html webpage.

C.8.1.4: Interpretation of Results

These results indicate that the creation of a new user has failed, but the webpage that allows the user to input new patient has been created. The next step in this target specification is to handle the data being input by the user and transferring it to the mysql server. Essentially the user is able to enter in the information needed, but the application is not saving the data.

C.2.1.5: Conclusion

The create feature failed, the next step that needs to be taken is write the code that will store the input patient info into the backend mysql database. The design changes that needs made is the inclusion of a MySQL query which can store the data. A potential solution is to use php code to handle the queries however this decision needs to made by the team as a whole to make sure the solution is cohesive with the entire project.

C.8.2: Experiment 2 - Delete a patient

This experiment tests the ability of the user to delete a patient record.

C.8.2.1: Experimental Procedure

1. Execute and login to the eFichero application
2. Navigate to the patient list web page and select a specific patience
3. Click on the delete button
4. Refresh the patient list to make sure that the patient has been deleted
5. Close the application

C.8.2.2: Equipment

1. Personal laptop
2. Code base for the eFichero application

C.8.2.3: Results

The results are that in this current system there is no delete button or delete functionality in place.

C.8.2.4: Interpretation of Results

Interpretation is that the delete feature needs to be implemented into the next software sprint. This will be a crucial feature for the user to delete the patients that they are no longer responsible for. The proposed solution is to use mysql query to remove a patient based off thier unique ID in the system.

C.2.2.5: Conclusion

The delete feature still needs to be implemented into the eFichero application.

C.8.3: Experiment 3 - Navigation

The purpose of this experiment is to ensure proper navigation of each button, within each page on the system.

C.8.3.1: Experimental Procedure

1. Run the build script and open the web application
2. Verify the login page is readable, that all expected elements are present and login to the eFichero application
3. Log-in to the system
4. Verify the first page is the home page and all the elements are present such as the navigational buttons to the patient list,schedule, and home
5. Navigate to the patient list page

6. Verify that the patient information is viewable and readable and all the expected elements are present.
7. Navigate to the schedule page
8. Verify all the scheduling information is present and the navigational elements are present
9. Close the application

C.8.3.2: Equipment

1. Personal laptop
2. Code base for the eFichero application

C.8.3.3: Results

Table VII.C.8.3.3.1: Experimental Results

The below table contains navigation testing results for the eFichero application

Navigation Item	Present?	Readable?	Pass/Fail?
Login Button	Yes	Yes	Pass
Signup Button	Yes	Yes	Pass
Home Button	Yes	Yes	Pass
Schedule Button	Yes	Yes	Pass
Patient List Button	Yes	Yes	Pass
Patient Information Sidebar	Yes	Yes	Pass
Patient Information Tabs	Yes	Yes	Pass
Signup Link	Yes	Yes	Pass
Login Link	Yes	Yes	Pass
Delete User	No	Yes	Pass
Schedule Date Selection	Yes	Yes	Pass

The experiment described above passed with the exception of no delete button. Other than that all of the different navigational buttons were present and readable.

C.8.3.4: Interpretation of Results

The results show that all of the desired functionality for a minimal viable product are present, readable, and usable.

C.8.3.5: Conclusion

In conclusion the navigational aspect of the application was a success. It should be noted that the features listed are the very minimum required for an operable site. There should be work done in the future to define features and functionality to make the system more user-friendly.

C.9: Efficiency

C.9.1: Experiment 1 - Comparison

The purpose of this experiment is compare our system to industry standard applications.

C.9.1.1: Experimental Procedure

1. Research what web technologies are used in similar scope web applications
2. Compare these results with framework choices made early on

C.9.1.2: Equipment

1. Personal Computers
2. Internet Connection

C.9.1.3: Results

The results from the Efficiency experiment are listed below in Table VII.C.9.1.3.1: Experimental Results.

Table VII.C.9.1.3.1: Experimental Results

The Experimental Results table contains an analysis of various researched web framework tools.

Application	Notes
Node.js	Node.js with Express.js on the back-end is a very common and reliable web server framework for small to large scale web application deployments.
MySQL	MySQL is an industry standard relational database management system for small to large scale database deployments.
Angular.js	Angular.js is a reliable front-end framework for creating modern front-end experiences.

C.9.1.4: Interpretation of Results

Our results showed us that this type of web application is a very common type of web application stack. Node.js with Express.js is not bloated, and is used in many web applications similar in function and scale to our web application. Using Angular.js for the front-end is a modern, efficient, and ultimately appropriate way to display front end functionality.

C.9.1.5: Conclusion

The research done proved that this type of web application uses efficient tools and practices for its purpose. It is not too "bare-bones", and it is also not "bloated". Using Node.js with a MySQL server and Angular.js on the front end is a common small-scale web application stack, and is appropriate for this project.

D.Conclusion

The results from all of the experiments across the metrics are listed in Table VII.D.1, below. The majority of Metrics either passed or on the path to passing by before this project's due date. Therefore, this project's experimental verification completed successfully.

Table VII.D.1: Experimental Verification Results

The Experimental Verification table contains the results from all the experimental verification testing.

#	Metrics	Pass/Fail	Design Improvement
1	**Number of Users**		
1.1	Number of Users	Pass	None
2	**On Times/Off Times**		
2.1	Access Server Throughout Whole Day	Fail	Implement server to test
3	**Supported Devices**		
3.1	Hosted on Different Operating System	Pass	None
3.2	Different Browsers	Pass*	Add additional compatibility configurations
3.3	Different Devices	Pass*	Add additional formatting for smaller screens
4	**Scalability**		
4.1	Number of Users	Pass	None

4.2	Cloud Stress Test	Pass	None
5	**Cost of System**		
5.1	Budget Review	Pass	None
6	**Data Transfer**		
6.1	Data Transfer Methods	Pass	Additional testing should be done for LAN and WiFi testing
7	**Patient Information**		
7.1	Each patient contains required information	Pass	None
7.2	Patient storage is HIPAA compliant	Fail	Additional features need to be added in future versions for HIPAA compliancy, such as transport encryption, storage encryption, having backups, etc
8	**Usability**		
8.1	Creating a New User	Fail	This feature is in progress for completion, will be implemented by the end of the sprint
8.2	Delete a Patient	Fail	This feature is in progress for completion, will be implemented by the end of the sprint
8.3	Navigation	Pass	None
9	**Efficiency**		
9.1	Application Comparisons	Pass	None

3.1: Core functioning works on all browsers, however, a few additional features do not work on versions of Chrome and Internet Explorer

3.2: Application runs on desktop computers, but is not currently formatted to run on devices with smaller screen sizes.

VIII. Performance Calibration

A. Why is Server Stress Important?

Computing power is very cheap in the developed world. With Gigabytes of RAM in smartphones and other cheap devices, it can be easy to forget about some of the slower networks that exist around the world. This application was therefore designed to be as lightweight and network-friendly as possible, so a user with even the slowest of internet in El Salvador could manager to use it with ease.

B. How Server Stress Analytics Were Collected

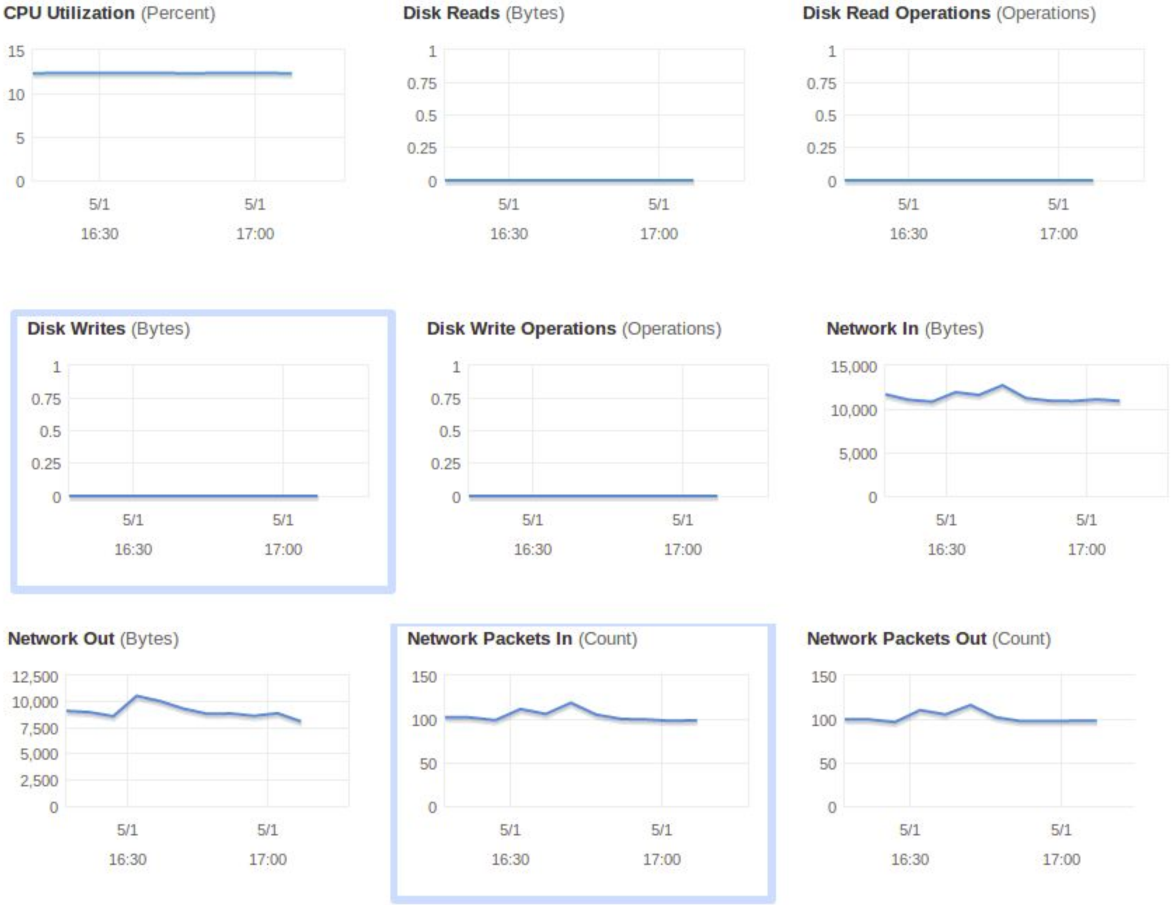
This web instance was hosted on Amazon Web Services for a period of few days. Various students and other users were given temporary logins, and were asked to use the website as they normally would. Amazon Web Services keeps various metrics of network usage, CPU usage, memory usage, etc. These metrics were automatically collected and analyzed after a few days.

C. Results

The results collected from Amazon Web Services are shown in Figure VII.C.1, below.

Figure VIII.C.1: Usage Metrics

Various Amazon Web Services analytics of a representative hour in time of average usage during this experiment.



D. Conclusion

One of the most important parts of the above analytics is the CPU usage. Amazon Web Services allocated us shared access to a 3.3Ghz x86-64 CPU, and 1Gb of server memory. Only 12% of the CPU was utilized during a representative hour, which means that a less powerful CPU could easily handle the load of this web application. This is a success for the target usage of this web application.

The other most important part of the above analytics was the Network In/Out usage. Averaging at around 10-15 Kb in both directions is considered to be very low for a modern web application. This would scale very well for lower bandwidth network infrastructures, thus being a success for the target usage of this web application.

IX. Risk analysis* needs further detail with severity calculations

The table below contains the different risks associated with the design and creation of the eFichero application. These are some of the risks that could have affected the completion of the

project throughout the year starting with any from the fall 2016 semester. The first aspect is the event itself which describes a risk that could have taken place, then the severity of this risk, and the probability of it occurring. The scale of each of these is from 1 - 10
Risk scale 1 -10

Table IX.1: Risks associated with the design of the eFichero application

Event	Severity	Probability	Risk	Mitigation plan	Justification
Administrative Risks					
Initial prototype is unfeasible	0.6	5	3	Work with the customer, to make sure all their needs are addressed in the next prototype iteration	Since the final design will be based on any prototypes we develop, we want it to match customer needs
Components are late	0.6	4	3.4	We can attempt to buy components we need from local stores, or order them at rush delivery from	Even though our project is mostly software based, there are some components we need to order for testing purposes.

Software incompatible with customer hardware	0.9	8	7.2	Find out out what hardware the customer is using, and design the software around those specifications. Order a common tablet used in El Salvador and run the application on it.	It is possible that the technology used by our customer might not support the newest version of internet applications so we design our application to be robust.
Negative legal ramifications of storing El Salvadorean patient medical records on U.S. based servers	0.9	7	6.3	Research any El Salvador patient confidentiality regulations, as well as U.S. legal ramifications of hosting that information	It's important for the team to think about making the application feasible with international standards
Physical server computer getting stolen off of medical record center area	0.9	9	8.1	Look into designing cloud based server that communicates with U.S. based servers over SMS	It is important to consider the setting in which the application will be used
The application is in the wrong language	0.9	2	1.8	Show the application to the customer to make sure that each aspect is in culturally appropriate Spanish	Since the application will be used in El Salvador we need to make sure all fields are in Spanish, specific to El Salvador. It is important that the application has a great user experience

Overlook major cultural or technical differences between U.S. and El Salvador (i.e. SMS differences per country)	0.9	5	4.5	Have some liaison to help us test certain functionality in El Salvador; do heavy research on various differences	There is a chance that the solution we provide is not the standard used in El Salvador.
Not usable enough for practitioners who don’t normally use advanced technology	0.7	7	4.9	Research intuitive input systems; prototype often and use liaison to receive consistent feedback	It is important to know how the application will be used by the customer, and design to their needs
Not able to make deadlines	0.5	5	2.5	Finish deliverables ASAP, communicate clearly and often about if team members cannot meet their responsibilities for whatever reason	It is possible that the team falls behind the schedule’s goals, and it is important to have a plan in place to handle it
Software Risks					
Software contains too many bugs	0.4	8	3.2	Use a testing framework as karma.js and jasmine.js to test the software	It is important to have a testing environment to catch any possible errors that are occurring in the application

Authentication service to login is non-functional	0.5	2	1	Research existing implementations of authentication services for small web applications	It is possible that the authentication service, is not fully functional and we need to be able to minimize the risks
The backend of the application cannot connect to the front end	0.9	5	4.5	The mitigation plan, is to iteratively test the connection between the mysql server and the frontend	The data that is present in the backend needs to be displayed to the user for the basic functionality in the system to be present
Outside user can hack the database	0.9	4	3.6	Advise the users to use good passwords, and set up more encryption, once the application is in use.	Since the data used in this application is private patient data, it is very important to consider the different risks associated with a potential security breach
Raspberry Pi Risks					
Raspberry Pi crashes	0.5	6	3		
Calculating for the correct power input	0.5	5	2.5	It is important to keep in mind the different components that are being connected to the system, to make sure the right amount of power is drawn	This is important because of the different risks associated with drawing too much power from a source.
Difficult to understand error from Pi if	0.7	6	4.2	Before using the raspberry pi system, advise	For the user to be able to diagnose a problem with the raspberry pi the user

it occurs				the user to have access to a linux based terminal to login to the Raspberry pi	needs to have knowledge with linux to be able to figure out the issue.

X. Economic Analysis

This section outlines the timeline and economic viability of the project. It includes detailed project scheduling, individual component costs, system costs, and customer/market information.

A. Project Timeline

The timeline of this project is best broken up into two parts: preparation work; and system work. From September through December, we focused mostly on preparation work. We compiled background and market information, as well as the desired customer needs; this process required at length conversations with a number of contacts, including two doctors who (have) practice(d) in El Salvadorean health clinics. We also sent out surveys to other providers who work in this clinic to fill out for more information. Then we compiled a list risks to the project and their respective rank - what they are and how likely they are to happen. From there we were able to decide on the system requirements and an overall system plan. Aside from the preparation work, we also created the initial code base and decided on what hardware we were going to move forward with. From January through May, we compiled the system. We ordered the components and moved forward with the system code. In terms of the code we made many decisions based on the feedback we had received from our contacts - we tried to keep our user interface similar to that of a system they have in place (a system responsible for obtaining demographic/census information). Since we used an agile development process, we were constantly testing as we moved forward, fixing bugs as we went rather than finishing the system and having to return back later to fix things.

(For the specific tasks completed each semester, refer to the Gantt Charts in the Appendix)

B. Budget Breakdown

For this project, we were given a budget of $2000.00 from an NIH grant. The goal was to create a low cost system, with an ideal cost of less than $2000.00 for development and overall implementation. Our final system cost $362.59, $1637.41 under budget. Many of the costs are associated with the development and testing of the system, and would not be relevant costs to an implemented system. The overall estimated cost of the system (excluding development expenses) would include the Raspberry Pi ($39.95) and its Enclosure ($7.95), External Hard Drive ($55.99) and Power Supply ($7.50), for a total of $111.39. It is important to note that this cost assumes that the website will be hosted on the Pi, and files stored in the attached hard drive. Another option would be to host the website in the cloud, which would incur an associated monthly fee. The exact breakdown of the cost for development and implementation of the system is as follows:

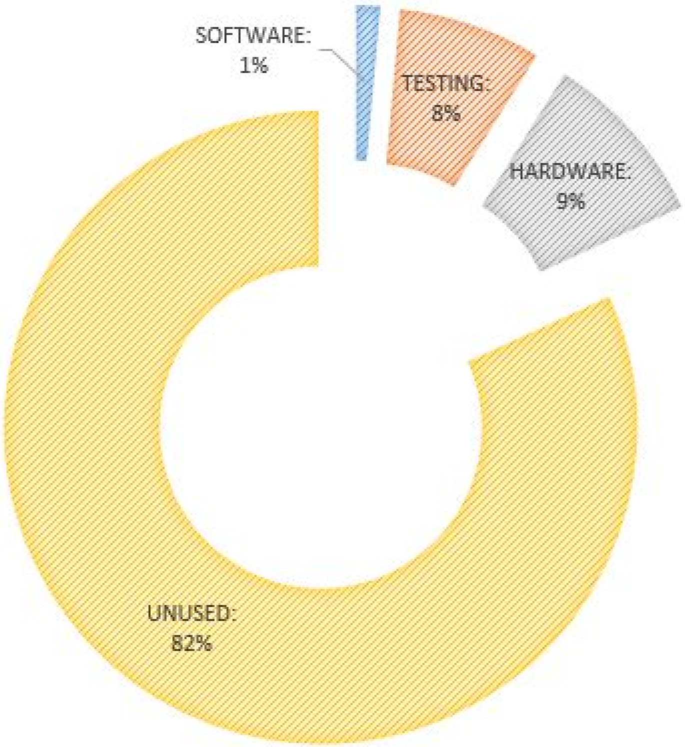
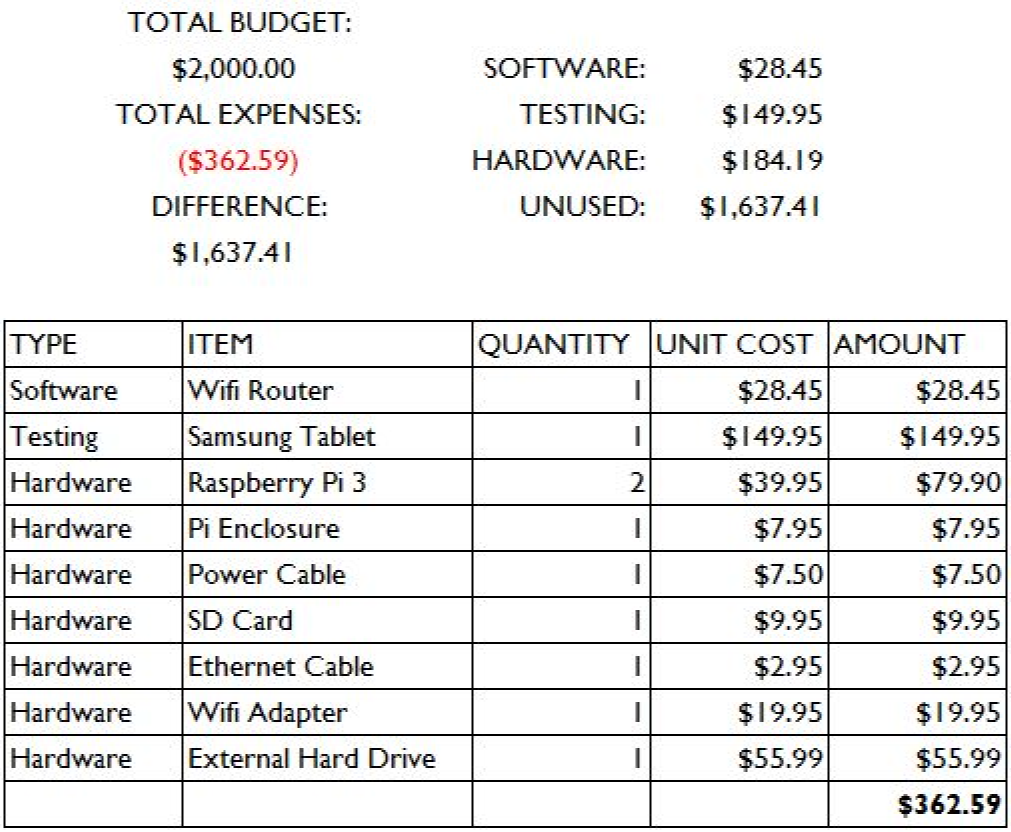


Figure X.B.1: Budget Breakdown Chart

The above figure is a graphical representation of the budget breakdown. In total, the four sections combine to equal our budget of $2000.00. Combining the software, testing, and hardware results in the total expenses.



TOTAL BUDGET:		
$2,000.00	SOFTWARE:	$28.45
TOTAL EXPENSES:	TESTING:	$149.95
($362.59)	HARDWARE:	$184.19
DIFFERENCE:	UNUSED:	$1,637.41
$1,637.41		

TYPE	ITEM	QUANTITY	UNIT COST	AMOUNT
Software	Wifi Router	1	$28.45	$28.45
Testing	Samsung Tablet	1	$149.95	$149.95
Hardware	Raspberry Pi 3	2	$39.95	$79.90
Hardware	Pi Enclosure	1	$7.95	$7.95
Hardware	Power Cable	1	$7.50	$7.50
Hardware	SD Card	1	$9.95	$9.95
Hardware	Ethernet Cable	1	$2.95	$2.95
Hardware	Wifi Adapter	1	$19.95	$19.95
Hardware	External Hard Drive	1	$55.99	$55.99
				$362.59

Figure X.B.2: Budget Breakdown Tables

The above figure (screenshot of an Excel spreadsheet) shows the numerical budget breakdown.

C. Customer/Market Information

Our current target customer is a rural health clinic in El Salvador; however, there is potential for this project to find use in other countries that lack electronic record systems as well. The current system in place in these clinics is a paper filing system that is very time intensive and unorganized. The benefit to our customer in using our system is to reduce the amount of time required for a health provider to determine which patients need to be seen and when, as well as keeping all the data in one organized location. Though there is the benefit of improved use of time to the health providers, it also provides more consistent care to the patients within the system.

XI. Issues and Standards

As a whole, our project did not have to meet many specific issues or standards because it was a new project and was not funded by any official organizations. This project is in its first year and was created by Dr. Olson, so at this point it is essentially a proof of concept to hopefully be implemented in the future. The project was funded by an NIH (National Institutes of Health) grant, so a minor required obligation was that the team members were each required to attend a course on research ethics. In addition to this, for the project to actually be used by El Salvador's health care system it will need to comply with international healthcare standards. From the start, we identified HIPAA compliance standards as a goal for our project. Also, as with any engineering product, a significant amount of testing is necessary to prove efficacy of a product before it can be released.

A. HIPAA Compliance

To be used in a clinic, eFichero needs to meet healthcare patient privacy standards, to account for this, HIPAA compliance standards were set as a goal. HIPAA certification was not obtained given the scope of this project. Application for certification is a significant process that can only be done when the product is ready for market. eFichero is not in a state that is ready for actual industrial use, so HIPAA certification was not applied for. However, as stated in the verification portion of this document, an assessment was made to determine how compliant eFichero is with general HIPAA guidelines. As stated in section C.7.2, eFichero is not currently HIPAA compliant. The product does meet various general standards, such as the ability for authorization into the application and the ability to delete patient information, however, it does not meet more in depth standards such as data transport and storage encryption. For future work on eFichero, HIPAA compliance is highly desired before actual implementation.

B. NIH

The funding from this project comes from the National Institute of Health, as stated above and this section includes more details on the organization and trainings. The NIH organization is the largest public funder of Biomedical engineering projects in the world. These projects range in wide variety and has a set of trainings necessary to use NIH fundings. These trainings were both online and in person. The online portion of the training required the user to read over documentation on a specific topic such as a breach in ethical conduct during research, then take an online exam. The in person portion of the trainings had students and faculty gather to discuss the lessons learned and the different implications of unethical conduct in the world of academia. Overall the trainings from the NIH grant were beneficial for the completion of the project in an ethical manner.

C. Testing

In addition to HIPAA compliance, eFichero will need significantly more testing before it could be released. Testing would be required for HIPAA compliance as well as technical performance. The stand alone server and wireless router would need to be tested for reliability, security, and performance. The application would need to be tested for successful performance of all functions, secure access for only health professionals, and robustness. Lastly, usability testing would be needed to ensure eFichero actually improves the quality of care and efficiency of the clinics in El Salvador.

XII. Project Legacy

Over the past eight (8) months our group has created a low cost healthcare record system for rural health clinics in El Salvador. We were able to maintain a healthy team dynamic throughout the year, which allowed us to overcome *most* challenges with ease; these challenges stemmed from hardware issues, software issues, and communication issues.

One of the most impactful issues that arose during this project was communication with the clinics in El Salvador. Though we had two (2) contacts that worked in the El Salvador health system, constant communication proved to be difficult due to the distance and language barriers. Speaking with contacts required translation (Spanish-English), which we were able to overcome thanks to group members being able to speak Spanish and translate back to us later. The distance proved to be difficult as we as a group could not see the problem first-hand; we relied on what communication we had with our contacts as well as first hand experience from a group member and our advisor. In terms of the impact on our project, the communication issues made it difficult for us to get feedback in a timely manner; on more than one occasion we got feedback that changed the direction of our project and set us back.

Though this was a primarily software-based project, it did require the use of a few hardware components. There were two issues regarding the hardware: the first was in determining the hardware components required, and having a component break. First, we had a number of ideas on how to connect the users to the system, ranging from providing a computer to building a cell tower. In solving this, our group debated the pros and cons of each system, before arriving on the idea of a small computer system, which would allow for local access to the website; this was implemented using the Raspberry Pi. The next issue arose with the Raspberry Pi - the SD card port connections broke away from the board and the system would no longer boot. Though the device was under warranty and could have been replaced, the process would have taken too long, so a new Pi was ordered.

While we are happy with the current state of our project, there is always room for improvement. The website could be updated to include different or more information, or the user interface could be changed to allow for better ease of use. The hardware could be fully implemented as designed, and powered using solar panels and batteries as to not require a power source. In all, it could be to the benefit of the clinics in El Salvador if this project was continued and further improved. To any future teams who wish to continue on with this project: we recommend finding as many contacts as possible and creating a schedule of contact with them; and when working with hardware, given money is available, order two (2) of any vital component.

XIII. Conclusion

After perceiving a need in El Salvadoran health clinics for an electronic scheduling system, customer needs were probed and analyzed from various sources. A list of target specifications was established to guide development towards a common goal and problems were successfully broken down into manageable pieces by each development team member.

The final design was established and developed. Using a Node.js backend, incoming web requests from the client could successfully be handled and sorted to perform their acheived purpose. These functions include sending static files for the front-end user-interface, passing off dynamic data to write to the database, and handling exceptions. The static user-interface files were designed to be intuitive and easy-to-use by the non-technical health promotores, as well as to contain the prevalent information in the Spanish language. The database connections were handled by writing to a MySQL database instance, taking advantage of fully relational database managements and backups. This whole system could then be packaged and distributed in either a Raspberry Pi portable server setup, local server hosting, cloud server hosting, or other options.

The aforementioned design was successfully created and verified against a large range of experiments. This collection of experiments could verify the proof-of-concept success of the final design, using tests such as various client-side devices, scalability, cost, patient data information, usability, and more. This system's performance was calibrated using Amazon Web Services analytics, which demonstrated that it could easily be scaled to El Salvadoran health clinics. Risks and economic impacts were analyzed, attempting to alleviate all negative effects as much as possible. Mitigation plans for these known issues were drawn out, offering concrete steps for how to resolve them. The legacy of this project was finally looked at, looking at what concrete non-technical steps could be taken to preserve the spirit of this project.

The scope of this project naturally progressed more towards the research side of things. The development completed provides a good proof-of-concept, as well as a theory of how to alleviate the current problem. It can therefore be considered a success, being mindful that future steps are needed to ensure the actual adoption of this software.

XIV. References

[1] Downer Sean R , Meara John G , Da Costa Annette C Sethuraman Kannan (2006) SMS text messaging improves outpatient attendance . Australian Health Review 30, 389-396.

[2] "Epic." *Epic*. N.p., n.d. Web. 26 Oct. 2016.

[3] "The Auto-Scheduler Supports the Best Recruiting Processes." *The Next Generation of Automated Scheduling*. N.p., n.d. Web. 26 Oct. 2016.

https://luxsci.com/blog/what-makes-a-web-site-hipaa-secure.html
http://smallbusiness.chron.com/hipaa-phi-policy-procedures-58616.html

XV. Appendices

A. Appendix A - Full listing of Packages Used on Backend

1. Express Web Framework

Express web framework is a very common package used in NodeJS frameworks to help with routing and using middleware. We found this package to be extremely helpful in speeding up the development process as much of the setup work for the http server was done for us by express.

2. Express-Session

Express-session is a tool used alongside express to keep track of a user's session. Express-session does this by keeping track of cookies in the user's browser and maintains a continuous session for each user. This package is especially necessary for authentication, as it will keep a user logged in for the duration of their session so that they can perform functions on each page without being redirected to the login page.

3. Cookie Parser

Cookie parser is also used with express-session to keep track of sessions. Because authentication and express-session rely on cookies, a cookie parser is needed to check the user's session information.

4. Body Parser

The Body Parser package was utilized in the application to extract information from incoming requests. This is helpful for POST and DELETE requests that contain larger information for the requests such as parameters to search the database for.

5. Path

The Path package is simply a package to append to and manipulate packages on the local file system. This package was primarily used for declaring static file paths for the server to reduce the number of dynamic requests the server makes, and be able to quickly serve up static pages.

6. MySql

The MySql package is the provided package to connect to the database. There is not much to note about this, other than it allows us to open active connections to the database and query, insert, and delete against it.

7. Passport

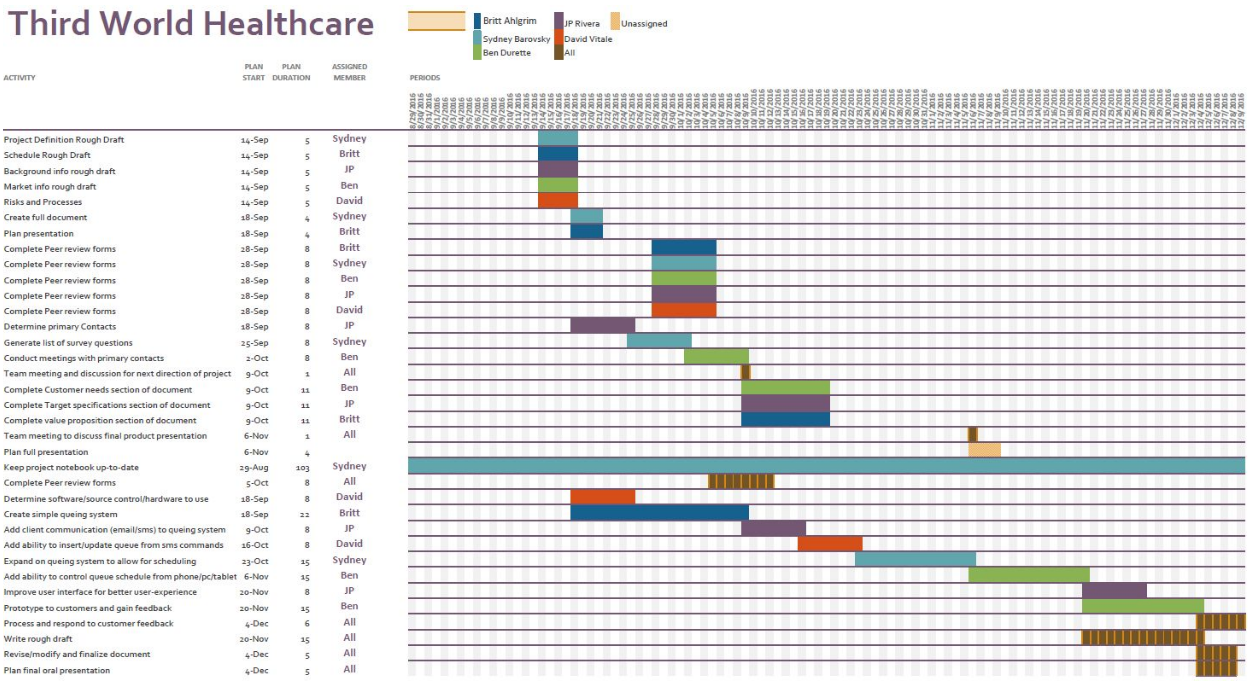
Passport is the package used for authentication. Passport provides many different ways to authenticate users, but our application relies on local authentication. Local authentication allows us to maintain and manage users' usernames, and store them securely.

8. Flash

Flash allows the application to send flash messages to the client screen without having to build that into the html and client javascript pages. The flash messages in our application are used for notifying the user of incorrect credentials or invalid registration username.

B. Appendix B - Gantt Charts

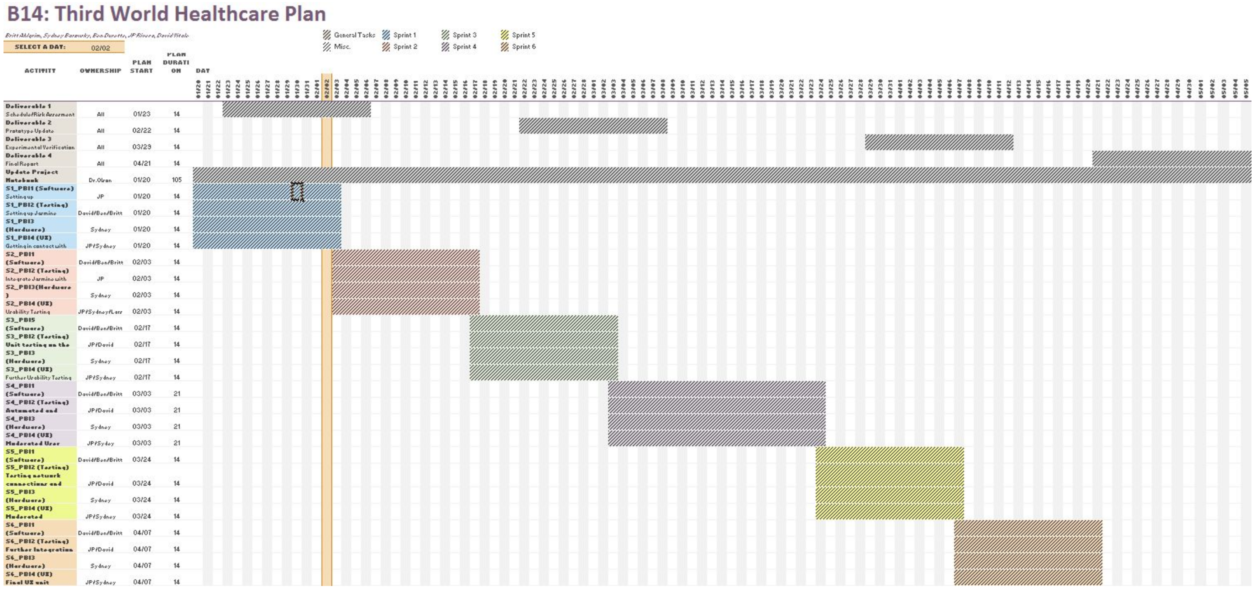
1. September - December



Third World Healthcare

ACTIVITY	PLAN START	PLAN DURATION	ASSIGNED MEMBER
Project Definition Rough Draft	14-Sep	5	Sydney
Schedule Rough Draft	14-Sep	5	Britt
Background info rough draft	14-Sep	5	JP
Market info rough draft	14-Sep	5	Ben
Risks and Processes	14-Sep	5	David
Create full document	18-Sep	4	Sydney
Plan presentation	18-Sep	4	Britt
Complete Peer review forms	28-Sep	8	Britt
Complete Peer review forms	28-Sep	8	Sydney
Complete Peer review forms	28-Sep	8	Ben
Complete Peer review forms	28-Sep	8	JP
Complete Peer review forms	28-Sep	8	David
Determine primary Contacts	18-Sep	8	JP
Generate list of survey questions	25-Sep	8	Sydney
Conduct meetings with primary contacts	2-Oct	8	Ben
Team meeting and discussion for next direction of project	9-Oct	1	All
Complete Customer needs section of document	9-Oct	11	Ben
Complete Target specifications section of document	9-Oct	11	JP
Complete value proposition section of document	9-Oct	11	Britt
Team meeting to discuss final product presentation	6-Nov	1	All
Plan full presentation	6-Nov	4	
Keep project notebook up-to-date	29-Aug	103	Sydney
Complete Peer review forms	5-Oct	8	All
Determine software/source control/hardware to use	18-Sep	8	David
Create simple queing system	18-Sep	22	Britt
Add client communication (email/sms) to queing system	9-Oct	8	JP
Add ability to insert/update queue from sms commands	16-Oct	8	David
Expand on queing system to allow for scheduling	23-Oct	15	Sydney
Add ability to control queue schedule from phone/pc/tablet	6-Nov	15	Ben
Improve user interface for better user-experience	20-Nov	8	JP
Prototype to customers and gain feedback	20-Nov	15	Ben
Process and respond to customer feedback	4-Dec	6	All
Write rough draft	20-Nov	15	All
Revise/modify and finalize document	4-Dec	5	All
Plan final oral presentation	4-Dec	5	All

2. January - May



B14: Third World Healthcare Plan

Britt Ahlgrim, Sydney Borowsky, Ben Duretto, JP Rivera, David Vitale

SELECT A DAY: 02/02

▨ General Tasks	▨ Sprint 1	▨ Sprint 3	▨ Sprint 5
▨ Misc.	▨ Sprint 2	▨ Sprint 4	▨ Sprint 6

ACTIVITY	OWNERSHIP	PLAN START	PLAN DURATION	DAY
Deliverable 1 Schedule/Risk Assessment	All	01/23	14	
Deliverable 2 Prototype Update	All	02/22	14	
Deliverable 3 Experimental Verification	All	03/29	14	
Deliverable 4 Final Report	All	04/21	14	
Update Project Notebook	Dr.Olsan	01/20	105	
S1_PBI1 (Software) Setting up	JP	01/20	14	
S1_PBI2 (Testing) Setting up Jasmine	David/Ben/Britt	01/20	14	
S1_PBI3 (Hardware)	Sydney	01/20	14	
S1_PBI4 (UX) Getting in contact with	JP/Sydney	01/20	14	
S2_PBI1 (Software)	David/Ben/Britt	02/03	14	
S2_PBI2 (Testing) Integrate Jasmine with	JP	02/03	14	
S2_PBI3(Hardware)	Sydney	02/03	14	
S2_PBI4 (UX) Usability Testing	JP/Sydney/Lars	02/03	14	
S3_PBI5 (Software)	David/Ben/Britt	02/17	14	
S3_PBI2 (Testing) **Unit testing on the**	JP/David	02/17	14	
S3_PBI3 (Hardware)	Sydney	02/17	14	
S3_PBI4 (UX) Further Usability Testing	JP/Sydney	02/17	14	
S4_PBI1 (Software)	David/Ben/Britt	03/03	21	
S4_PBI2 (Testing) **Automated and**	JP/David	03/03	21	
S4_PBI3 (Hardware)	Sydney	03/03	21	
S4_PBI4 (UX) **Moderated User**	JP/Sydey	03/03	21	
S5_PBI1 (Software)	David/Ben/Britt	03/24	14	
S5_PBI2 (Testing) **Testing network connections and**	JP/David	03/24	14	
S5_PBI3 (Hardware)	Sydney	03/24	14	
S5_PBI4 (UX) **Moderated**	JP/Sydney	03/24	14	
S6_PBI1 (Software)	David/Ben/Britt	04/07	14	
S6_PBI2 (Testing) **Further Integration**	JP/David	04/07	14	
S6_PBI3 (Hardware)	Sydney	04/07	14	
S6_PBI4 (UX) **Final UX unit**	JP/Sydney	04/07	14	