Mobile Clinic Electronic Medical Record System

*Deliverable 2 – Requirements Document*

**Senior Project**

CIS 4911 - U01

**Instructor:** Sadjadi

**Team Members**

Steven Berlanga

Carlos Corvaia

Rigoberto Hernandez

Michael Montaque

Sebastian Zanlongo

**Date:** 4/26/2013

**Mentor:** Steven Luis

# Abstract

Orant Charities is a non-profit organization dedicated to bringing healthcare to at-risk Malawi citizens. Physicians travel to the country every few months, and stay there for about two weeks. Currently, Orant uses a paper-based system to track patients and their visits to the clinic. The paper forms are easily lost, and are difficult to analyze for trends and data. We are building a mobile app that will allow users to more easily record patient information and analyze these records at a later time on order to improve on their services.

This document will analyze the current system and its limitations. After this, we move on to the milestones, tasks, and deliverables that fit the hardware and software requirements set forth. We take a look at all of the User stories that encompass the scope of the system as well as all the static and dynamic UML diagrams.

Table of Contents

Abstract 2

1. Introduction 4

a.Problem Definition 4

b.Scope of System 4

c.Terminology 4

d.Overview of Document 4

2. Current System 5

a.Limitations and Problems 5

3. Project Plan 6

a.Project Organization 6

b.Work Breakdown 6

i. Deliverables 6

ii. Milestones 7

iii. Tasks 8

c.Cost Estimate 9

4. Proposed System Requirements 10

a.Functional Requirements 10

b.Analysis of System Requirements 11

i. Use Case Model 11

ii. Static Model 14

iii. Dynamic Model 14

5 Appendix 16

aAppendix A - Complete Use Cases 16

bAppendix B - Use Case Diagram 42

cAppendix C - Static UML Diagram 44

dAppendix D - Dynamic UML Diagram 45

eAppendix E - User Interface Designs 46

fAppendix F - Diary of Meetings and Tasks 52

6 References 56

# Introduction

## Problem Definition

The creation of an open source mobile app that can provide users with the ability to register and track patients and their visits, and allow for later analysis of these visits. The mobile client will implement a “triage app” that allows nurses, doctors, and pharmacists to manage patients. The app will provide users with clean, flexible forms to replace their existing paper-based forms. There will also be a local server and cloud server providing backups and synchronization of the data using a custom API.

## Scope of System

The scope of the project is to develop an iPad based system for managing three triage stations necessary for providing patient care in rural conditions. Each system will be in charge of a particular task oriented around patient care:

* + - * Triage System - provide the ability for the triage staff to be able to take patients personal information and vitals.
      * Physician System - allow a doctor to review vitals, conduct exams, render diagnosis and if necessary prescribe medication.
      * Pharmacy System - allow a pharmacist to filled the prescriptions given to a patient by a doctor.

There will be a Local server that will allow the iPads to pass information between each other. This information will be available through a web application that will allow administrators to access/edit patient information.

## Terminology

* + - RIU - Registered iPad User
    - RIN - Registered iPad Nurse

## Overview of Document

# Current System

Orant Charities currently employs two paper based forms in order to track the events in their clinics. Triage nurses start by taking down a new patient’s name, family name, village, age, sex, and weight. They can also record what the patient is complaining of, and any symptoms they exhibit. The patient and this form then move on to the Physician, which will exam the patient and diagnose them with a condition and possible a prescription. This prescription is written down on a second form, with details such as tablets per day, time of day to ingest, and any other additional instructions. Finally, the patient will take these forms to the Pharmacist, which dispenses medication.

The issue of managing patients and patient records is one that has constantly affected clinics and hospitals. Orant cannot afford nor requires the conventional record management systems being employed in hospitals today.

## Limitations and Problems

The current system suffers from many of the issues that affect any paper based system. There is the loss of patient records during normal operation or traveling to and from the United States. The largest issue is that paper based records are difficult to analyze and data mine. Orant cannot refine its processes and planning because it relies mostly on patient and doctor feedback, rather than on collected data.

Moreover, the paper based records are sometimes difficult to pair with a returning patient, which can also compromise patient privacy.

# Project Plan

In the project plan section we discuss the assignment of roles in our development process. A detailed project schedule is presented and explained. Lastly, a cost breakdown of the system is presented.

## Project Organization

We chose to follow the Agile development process with Scrum ideals. Group members were split into the following roles:

- Project Manager - Carlos Corvia

- Architect - Steven Berlanga

- Developer - Rigo Hernandez

- Minute Taker - Michael Montague

- System Analyst - Sebastian Sotolongo

## Work Breakdown

Here, we will review the work breakdown of the project. Table one covers the deliverables that need to be written. Table two shows the milestones for each deliverable. Table three indicates which tasks must be met for each deliverable.

### Deliverables

|  |  |
| --- | --- |
| **Deliverable** | **Description** |
| 1 | Feasibility Study |
| 2 | Requirements |
| 3 | Design |
| 4 | Final Document |

### Milestones

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Name** | **Deliverable** |
| 1 | Requirements Elicitation | 1 |
| 2 | Analysis of System Requirements | 1 |
| 3 | Feasibility Study | 1 |
| 4 | Setup of Server | 1 |
| 5 | Setup of Development Environment | 1 |
| 6 | Creation of Schemas | 1 |
| 7 | Mockup App | 1 |
| 8 | Completion of Triage | 2 |
| 9 | Completion of Doctor | 2 |
| 10 | Completion of Pharmacy | 2 |
| 11 | Submit beta to Orant | 2 |
| 12 | Basic Cloud Server | 2 |
| 13 | Basic Local Server | 2 |
| 14 | Creation of API | 2 |
| 15 | Integration of Photos | 3 |
| 16 | Integration of Biometrics | 3 |
| 17 | Completion of Cloud Server | 3 |
| 18 | Finalized App | 4 |
| 19 | Finalized Local Server | 4 |
| 20 | Finalized Cloud Server | 4 |
| 21 | Finalized API | 4 |
| 22 | Submit to Orant | 4 |

### Tasks

|  |  |
| --- | --- |
| **ID** | **Title** |
| US-103 | As an administrator I want the server to connect with the physical Device so that information can be persistent throughout the system |
| US-44 | As a triage nurse I want to login into the system so that I can work securely |
| US-37 | As a potential user I want to create my own user profile so I can be properly identified in the system |
| US-2 | As a triage nurse I want to manually register a new patient so that I can Check-In a patient. |
| US-70 | As a triage nurse I want to link family members together so that I can quickly bring up other patients |
| US-5 | As a physician I want to diagnose a patient to provide the patient care. |
| US-11 | As a pharmacist I to want finalize and check out the patient so that the staff knows that patient's treatment is complete. |
| US-36 | As a pharmacist I want to fulfill the patient's prescription so that I can confirm they were given medication |
| US-13 | As a pharmacist I want see the doctor’s prescription so that I know what drug to dispense. |
| US-21 | As an application administrator I want to verify when the local system backs up to the Cloud so that I can verify the data's status |
| US-15 | As a Triage nurse I want to prioritize the patient so that the critical patients get attention first |
| US-25 | As an application administrator I want easily startup, shutdown and reset the local server so that adequate operating & troubleshooting measures can be executed |
| US-43 | as a physician I want to manually identify a patient so that I can find their records |
| US-67 | As a Triage Nurse I want to be able to check in a patient so that I can take their vitals and collect their family history. |
| US-68 | As a Triage Nurse I want to be able to search for an existing patient to record their vitals for the current visit |
| US-73 | As a Triage nurse I want to quickly discharge a patient so that patients with major issues can quickly see the doctor |
| US-81 | As a triage / doctor / pharmacist, I want to check in and search for patients using their fingerprints. |
| US-82 | As a doctor, I want to assign patient's medication from a table. |
| US-91 | As a triage / doctor / pharmacist, I want to search for a patient. |
| US-92 | As a doctor I want to be able to checkout a patient if no medication needs to be prescribed. |
| US-93 | As a triage, I want to assign patients a priority in queue. |
| US-94 | As a pharmacist, I want to select patients from a queue. |
| US-95 | As a doctor, I want to select patients from a queue based on priority. |
| US-96 | As a triage, I want to assign patient vitals. |
| US-97 | As a doctor, I want to save a patient's diagnosis to their current visit in queue. |
| US-99 | As a developer, I want to be able to test production on my rails application on a staging server before being live. |
| US-104 | As a triage nurse I want to be able to quickly abort the patient I am working on so that I can address other patients |
| US-105 | As a triage nurse I want to be aware of all the patients that are currently in the system so that I can monitor their progress |
| US-107 | As a Physician I want to be able to quickly select multiple medication for a patient |
| US-109 | As an Application Administrator I want to be able to sync (upload information) between the Local Server and the Cloud app |
| US-110 | As an Application Administrator I want to be able to sync (download information) between the Local Server and the Cloud app |
| US-112 | As a triage nurse I want to be able to login so that I can access my dashboard |
| US-113 | As a Pharmacist I want to be able to login so that I can access my patient queue |
| US-114 | As a Doctor I want to be able to login so that I can access my patient queue |
| US-115 | As an app user I want to be able to see the total number of tablets/fl oz. available for any particular medication |
| US-116 | As a triage, I want to have a dashboard to manage my account. |
| US-117 | As a triage, I want to assign patients temperature and brief explanation (title) of a patient's visit. |
| US-120 | As a physician, I want to see relevant patient information in the queue. |

## Cost Estimate

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Quantity** | **Cost** | **Total** |
| MacBook Air | 1 | 999 | 999 |
| iPad 2 | 3 | N/A | 0 |
| Heroku | 1 | 150 | 150 |
| Total |  |  | 1149 |

# Proposed System Requirements

In this chapter we created a walkthrough of the functionality using the functional requirements from the use cases we implemented. The use case diagram along with a brief description of the relationships between the use cases and actors was also included. The chapter ends with an analysis of the requirements. The analysis of the system requirements is broken down into Use Case Models, that will describe the proposed functionality of the system. Then the Static Models which will contain object diagrams and class diagrams. finally the dynamic models will be made up of sequence diagrams and state machines.

## Functional Requirements

We will be implementing a total of 15 use cases for MC-EMR. Which will be broken on how the system is comprised of:

1. The Mobile IPad Application shall provided a RIU with the ability to login (use case is MC-EMR001) as seen in appendix B
2. The Mobile IPad Application shall provided a RIU with the ability to log out (use case is MC-EMR002) as seen in appendix B
3. The Mobile IPad Application shall provided a RIN with the ability to create a new patient
4. The Mobile IPad Application shall provided a RIN with the ability to register a patients fingerprint
5. The Mobile IPad Application shall provided a RIN with the ability to check in a patient into a queue
6. The Mobile IPad Application shall provided a RIN with the ability to quick check-out a patient
7. The Mobile IPad Application shall enable a RIU with the ability to search for existing patient data
8. The Mobile IPad Application shall also give the capacity to a RIU to add a new entries to existing patients
9. The Mobile IPad Application shall provide to the RIU to modify existing patient data
10. The Mobile IPad Application shall give the capacity to a RIU to register medication given to patients
11. The Mobile IPad Application shall enable a RID the ability to diagnose a patient.
12. The Mobile IPad Application shall enable a RIP the ability to prescribe medication.
13. The Local Server shall provide a RLSA with the ability to push local data to the cloud
14. The Local Server shall provide a RLSA with the ability to pull cloud data
15. The Local Server shall provide a RLSA with the ability to sync local iPad with the most recent information

## Analysis of System Requirements

### Use Case Model

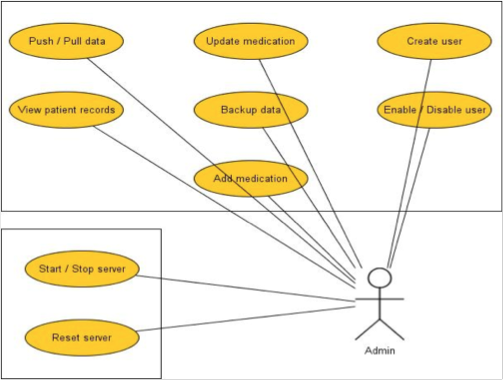


Figure 4.2.1

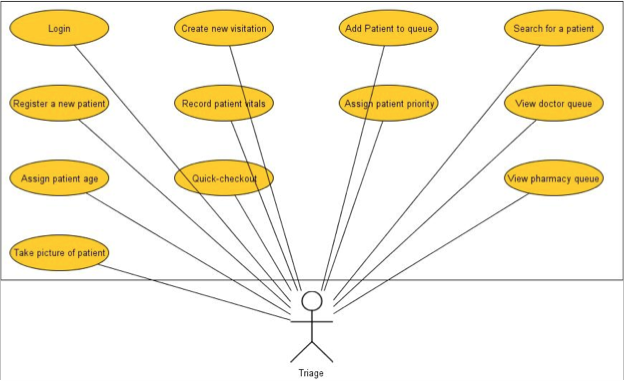


Figure 4.2.2

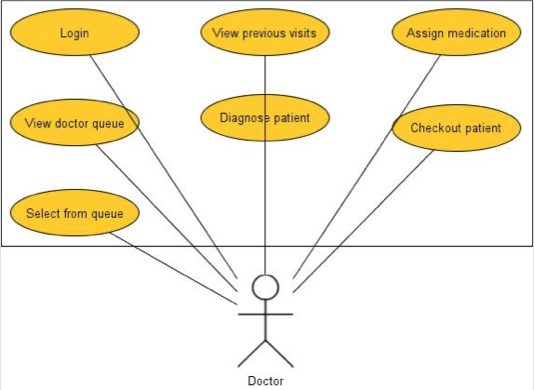


Figure 4.2.3

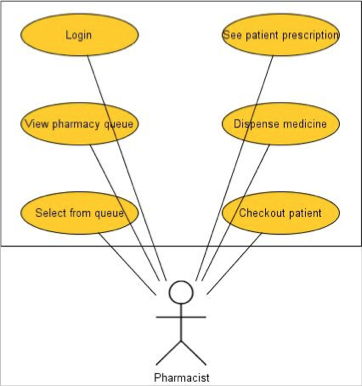


Figure 4.2.4

The fours actors in the system are a Registered IPad Nurse (RIN), Registered IPad Doctor (RID), Registered IPad Pharmacy (RIP), Registered Local Server User (RLSU). An RIN, RID and a RIP is a user that has gone through the registration process and can be authenticated by the iPad application only. A RLSU is a user that can authenticated by the iPad and Local Server applications.

A RIN can interact with every use case seen in Figure 4.2.2. A RID can interact with every use case seen in Figure 4.2.3. A RIP can interact with every use case seen in Figure 4.2.4. Lastly, a RLSU can interact with every use case seen in 4.2.1. There are similar use cases that are shared between different users such as Searching for a Patient, Logging into and out of the system and sending a patient into the waiting queue.

### Static Model

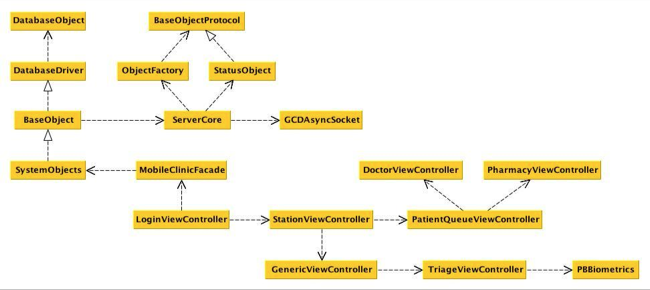


Figure 4.3.1

The diagram above (Figure 4.3.1) is the minimal class diagram that represents the structure of MC-EMR. This structure is replicated both in the iOS client and inside the Local server. The MobileClinicFacade is the entry point for any communication between the controllers and the database. The facade speaks to the specific object that is going to be modified. The ServerCode class sends a message to the GCDAAsyncSocket which communicates with another GCDAAsyncSocket class sitting on the Local Server which goes to the ServerCore to the Facade to the specific Controller;

### Dynamic Model

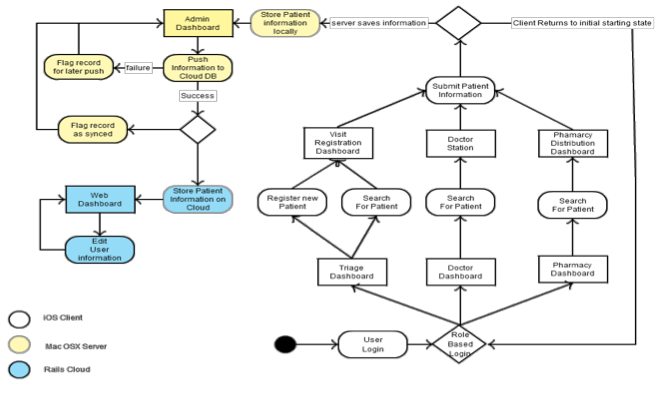


Figure 4.3.2

The diagram above (Figure 4.3.2) is the state diagram for the system. The System can have three paths after a user logging in. If a triage nurse logs in, he/she can either search for a patient or register a new patient. At this point, the triage nurse can register a new visit for the new patient.

If a Doctor/Pharmacy logs into the system, they can search for a patient in the queue. After selecting a patient they can write up a patient diagnosis and/or prescribe patient medication. After submitting the information, the user is sent back to their main screen.

When the information is sent to the Local Server, the server saves the information locally and sends the information to the cloud. The cloud application stores the information internally

# Appendix

## Appendix A - Complete Use Cases

Use Case ID: MC-EMR001 – Login

* + Details:
    - Actor:
      * RIU
    - Pre-conditions:
      * The application must be open.
      * The server must be active
    - Description:
      1. Use case begins when the user presses the Login button.
      2. The information is encrypted and passed to the server
      3. A username and password match is found, login flag is set
      4. Use case end as soon as the system receives the flag and displays personal RIU information.
    - Post-conditions:
      * RIU should be logged in and should see his/her main screen.
  + Alternative Courses of Action
    - In step D.3 (step 3 of Description section) the server does not find a match and the login flag is not set.
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server an exception will be throw letting know the user that the action was not performed
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 4 requests are made daily by RIUs.
  + Criticality: High. RIUs cannot access their accounts without being able to login.
  + Risk: Low. Implementation of this use case employs standard web-based technology as well as mobile device.

Constraints:

* + Usability
    - Intuitive steps.
    - No help required.
  + Reliability
    - 4% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - 5 seconds of delay is expected.
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR002 – Logoff

* + Details:
    - Actor:
      * RIU
    - Pre-conditions:
      * The user must have been logged into the system.
    - Description:
      1. Use case begins when the user presses the Logout buton
      2. The system sends the usertoken over to the local server
      3. The local server destroys the user token
      4. Use case ends when the Local server sends a signal to the iPad that the user has logged off.
    - Post-conditions:
      * RIU should be in the login screen
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the user’s token on the local server is not destroyed.
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 2 requests are made daily by RIUs.
  + Criticality: Medium. If a RIU cannot logout, this poses a security threat where a different user may use the system without having to login.
  + Risk: Low. Implementation of this use case employs standard web-based technology as well as mobile device.

Constraints:

* + Usability
    - Intuitive steps.
    - No help required.
  + Reliability
    - 3% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - 3 seconds of delay is expected.
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR003 – Create a New Patient

* + Details:
    - Actor:
      * RIN
    - Pre-conditions:
      * The RIN must have been logged into the system.
      * The RIN must be in the Registered Patient screen.
    - Description:
      1. Use case begins when a RIN fills in all the patient information and hits the “Register Patient” button
      2. The Client will store the patient information locally
      3. The Client will send the appropriate patient information to the local server
      4. The local server will store the appropriate patient information
      5. Use case ends when the Local server sends a signal to the iPad that the patient information has been saved.
    - Post-conditions:
      * RIN should be in the login screen
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server, RIN will receive a notification that the patient information could not be saved in the Local Server
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 125 requests are made daily by RIUs.
  + Criticality: High. If a RIN cannot register a new patient then the RID or RIP cannot see the new patients and will not be able to use the system effectively.
  + Risk: Medium. Implementation of this use case employs a intricate manner of communicate between an iPad and a Macbook.

Constraints:

* + Usability
    - No previous training time
    - On average the user should take less than 1 minute to complete the request
  + Reliability
    - 6% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved withint 3 seconds
    - System should handle 175 requests in 1 minute
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR004 – Register a Patients Fingerprint

* + Details:
    - Actor:
      * RIN
    - Pre-conditions:
      * The RIN must have been logged into the system.
      * The RIN must be in the Registered Patient screen.
    - Description:
      1. Use case begins when a RIN hits the “Register Patient Fingerprint” button
      2. The RIN presses the button corresponding to the appropriate finger the RIN is registering for the patient.
      3. The patient will swipe their finger on the biometric device
      4. The RIN will hit the “Save” button
      5. The Client will save the Fingerprint information on the Client and send the information to the Local Server
      6. Use case ends when the Local server sends a signal to the iPad that the Fingerprint information has been saved.
    - Post-conditions:
      * RIN should be returned to the patient registration page.
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server, RIN will receive a notification that the Fingerprint information could not be saved in the Local Server
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 60 requests are made daily by RIUs.
  + Criticality: Low. A RIN can still properly use and identify a patient without registration their fingerprints.
  + Risk: Medium. Implementation of this use case utilises a third party API to interact with the biometric hardware.

Constraints:

* + Usability
    - Previous training is required to perform this operation
    - On average the user should take less than 2 minute to complete the request
  + Reliability
    - 10% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved within 6 seconds
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR005 – Check a Patient Into Doctor Queue

* + Details:
    - Actor:
      * RIN
    - Pre-conditions:
      * The RIN must have been logged into the system.
      * The RIN must be in the Create New Visitation Page
    - Description:
      1. Use case begins when a RIN hits the “Send to Doctor” button
      2. The Client will send the patient information to the Local Server
      3. The Local Server will set the Patient’s inWaiting state to the Doctor Queue Flag
      4. Use case ends when the Local server sends a signal to the iPad verifying that the patient has been put into the queue.
    - Post-conditions:
      * RIN should be returned to the patient registration page.
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server, RIN will receive a notification that the Patient was not moved into the Doctor Queue.
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 125 requests are made daily by RIUs.
  + Criticality: High. If a RIN cannot move a patient into the Doctor Queue, the Doctor or Pharmacist will not be able to use the system.
  + Risk: Low. Implementation of this use case utilises a third party API to communicate between the Local Server and the Client.

Constraints:

* + Usability
    - No previous training required
    - On average the user should take less than 20 seconds to complete the request
  + Reliability
    - 8% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved within 4 seconds
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR006 – Quick Checkout a Patient

* + Details:
    - Actor:
      * RIN
    - Pre-conditions:
      * The RIN must have been logged into the system.
      * The RIN must be in the Create New Visitation Page
    - Description:
      1. Use case begins when a RIN hits the “Quick Checkout” button
      2. The Client will send the patient information to the Local Server
      3. The Local Server will close the Patients visitation.
      4. Use case ends when the Local server sends a signal to the iPad verifying that the patient’s visitation has been closed and the patient has been checked out.
    - Post-conditions:
      * RIN should be returned to the patient registration page.
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server, RIN will receive a notification that the Patient was not moved into the Doctor Queue.
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

-------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 25 requests are made daily by RIUs.
  + Criticality: Low. A RIN can still pass a patient through the system without checking out a patient quickly.
  + Risk: Low. Implementation of this use case utilises a third party API to communicate between the Local Server and the Client.

Constraints:

* + Usability
    - No previous training required
    - On average the user should take less than 5 seconds to complete the request
  + Reliability
    - 3% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved within 4 seconds
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR007 – Search Patient

* + Details:
    - Actor:
      * RIU
    - Pre-conditions:
      * The RIU must have been logged into the system.
      * The RIU must be in the Search Patient screen
    - Description:
      1. Use case begins when a RIU fills in the Patient Name and hits the “Search” button
      2. The Client sends the patient name over to the Local Server
      3. The Local Server looks for the patient information
      4. Use case ends when the Local server sends a signal to the iPad that the patient information has been saved.
    - Post-conditions:
      * RIU should display the patient information
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server, RIU will receive a notification that the patient information could not be found in the Server
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 5 requests are made daily by RIUs.
  + Criticality: Low. The likelihood of the Triage administering a patient is low due to the fact they they are a moving clinic throughout Africa
  + Risk: Low. Implementation of this use case employs a intricate manner of communicate between an iPad and a Macbook.

Constraints:

* + Usability
    - No previous training time
    - On average the user should take less than 1 minute to complete the request
  + Reliability
    - 2% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - System should handle 80 requests in 1 minute
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

---------------------------------------------------------------------------------------------------------

Use Case ID: MC-EMR008 – Add visit to existing patient

* + Details:
    - Actor:
      * RIN
    - Pre-conditions:
      * The RIN must have been logged into the system.
      * The RIN must be in the Search Results screen
    - Description:
      1. Use case begins when a RIN selects a patient from the Search Results screen
      2. The New Patient Visit screen will be shown
      3. The RIN fills in the appropriate patient information
      4. The RIN selects either Checkout or Check-In
      5. The iPad saves the new visit and attempts to sync with the Local Server
    - Post-conditions:
      * iPad will now display the New Patient screen.
  + Alternative Courses of Action
    - Select the Triage back button
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server, RIN will receive a notification that the patient information could not be saved on the server.
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

Use Case ID: MC-EMR009 – Edit Patient Information

* + Details:
    - Actor:
      * RIN
    - Pre-conditions:
      * The RIN must have been logged into the system.
      * The RIN must have searched for an existing
    - Description:
      1. Use case begins when a RIN selects a patient image
      2. The Client will enable all the text Fields
      3. Upon the RIN clicking “Save”, the Client will save the Patient information and send it to the Local Server
      4. The Local Server will save the Patient information
      5. Use case ends when the Local server sends a signal to the iPad that the patient information has been edited.
    - Post-conditions:
      * RIN will be in the patient detail screen.
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server, RIN will receive a notification that the patient information could not be saved in the Local Server
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 35 requests are made daily by RINs.
  + Criticality: Medium. If a RIN cannot edit a patient, the system can still run but there may be a high chance for error.
  + Risk: Medium. Implementation of this use case employs a intricate manner of communicate between an iPad and a Macbook.

Constraints:

* + Usability
    - No previous training time
    - On average the user should take less than 5 minute to complete the request
  + Reliability
    - 8% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved within 3 seconds
    - System should handle 45 requests in 1 minute
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

-------------------------------------------------------------------------------------------------------

Modification History

Owner: Steven Berlanga

Initiation date: 05/23/2012

Date last modified: 05/23/2012

Use Case ID: MC-EMR010 – Register medication given to patients

* + Details:
    - Actor:
      * RID
    - Pre-conditions:
      * The RID must have been logged into the system.
      * The RID must be at the Current Diagnosis screen
    - Description:
      1. Use case begins when a RID selects the Submit button
      2. The RID selects the Find Drug button
      3. After being presented with the Choose Medicine screen, the user selects the medication to prescribe
      4. RID presses the Select button
      5. After being presented with the updated Choose Medication screen, the RID selects the checkout button
      6. The iPad will save the patient visitation and attempt to sync with the Local Server
      7. Use case ends when the iPad displays the Patient Queue screen.
    - Post-conditions:
      * RID will be in the Patient Queue.
  + Alternative Courses of Action
    - RID can selected the Doctor back button to return to the Patient Queue
  + Extensions:
    - None.
  + Exceptions:
    - None
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

Use Case ID: MC-EMR011 – Diagnose a Patient

* + Details:
    - Actor:
      * RID
    - Pre-conditions:
      * The RID must have been logged into the system.
      * The RID must be in the patient Visitation screen
    - Description:
      1. Use case begins when a RID fills in the diagnosis information and hits “Add Prescription” button
      2. The Client saves the visitation information
      3. The Client passes the visitation information over to the Local Server
      4. The Local Server saves the visitation information
      5. Use case ends when the Local server sends a signal to the iPad that the visitation information has been saved.
    - Post-conditions:
      * RID will be in the prescribe medicine screen.
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server, RIN will receive a notification that the patient information could not be saved in the Local Server
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 130 requests are made daily by RINs.
  + Criticality: High. If a patient visitation can’t be saved, then Pharmacist can’t use the system..
  + Risk: Medium. Implementation of this use case employs a intricate manner of communicate between an iPad and a Macbook.

Constraints:

* + Usability
    - No previous training time
    - On average the user should take less than 10 minute to complete the request
  + Reliability
    - 4% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved within 6 seconds
    - System should handle 45 requests in 1 minute
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR012 – Prescribe Medicine to Patient

* + Details:
    - Actor:
      * RIP
    - Pre-conditions:
      * The RIP must have been logged into the system.
      * The RIP must be in the patient Prescription screen
    - Description:
      1. Use case begins when a RIP selects the prescription and clicks the “Prescribe Now” button
      2. The Client saves the prescription information
      3. The Client passes the prescription information over to the Local Server
      4. The Local Server saves the prescription information
      5. Use case ends when the Local server sends a signal to the iPad that the prescription information has been saved.
    - Post-conditions:
      * RIP will be in the patient prescription medicine screen.
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the system cannot establish connection with the Local Server, RIP will receive a notification that the prescription information could not be saved in the Local Server
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 115 requests are made daily by RIPs.
  + Criticality: High. If a prescription cannot be filled there is no way to now what medicine the doctors have left.
  + Risk: Medium. Implementation of this use case employs a intricate manner of communicate between an iPad and a Macbook.

Constraints:

* + Usability
    - No previous training time
    - On average the user should take less than 3 minute to complete the request
  + Reliability
    - 6% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved within 4 seconds
    - System should handle 75 requests in 1 minute
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR013 – Pull Information to Local Server

* + Details:
    - Actor:
      * RLSU
    - Pre-conditions:
      * The RLSU must have been logged into the system.
    - Description:
      1. Use case begins when a RLSU presses the “Pull Cloud Information”
      2. The Local Server sends a signal to the Cloud API
      3. The Cloud API packages the appropriate database information into a json response.
      4. Use case ends when the Local Server parses and receives the JSON response from the Cloud API
    - Post-conditions:
      * The Local Server now has the JSON response stored into the it’s SQLite DB.
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - None
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 230 requests are made daily by a RLSU.
  + Criticality: High. If information cannot be pulled down into the Local Server then the Server cannot maintain the most recent information
  + Risk: Medium. Implementation of this use case employs a intricate manner of communicate between a Mac OSX system and a rails API.

Constraints:

* + Usability
    - No previous training time
    - On average the user should take less than 13 seconds to complete the request
  + Reliability
    - 1% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved within 4 seconds
    - System should handle 130 requests in 1 minute
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR014 – Push Information to Local Server

* + Details:
    - Actor:
      * RLSU
    - Pre-conditions:
      * The RLSU must have been logged into the system.
    - Description:
      1. Use case begins when a RLSU presses the “Push Cloud Information”
      2. The Local Server packages up an object into a JSON object and sends a signal to the Cloud API
      3. The Cloud API receives the information, parses it and saves the object
      4. Use case ends when the Local Server receives and ok signal from the Cloud API
    - Post-conditions:
      * The Local Server now has the JSON response stored into the it’s SQLite DB.
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - If the JSON post is not what the API expects, the API will return and error code.
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 230 requests are made daily by a RLSU.
  + Criticality: High. If information cannot be pushed up into the Cloud then the Cloud will not hold any of the most recent information.
  + Risk: Medium. Implementation of this use case employs a intricate manner of communicate between a Mac OSX system and a rails API.

Constraints:

* + Usability
    - No previous training time
    - On average the user should take less than 13 seconds to complete the request
  + Reliability
    - 1% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved within 4 seconds
    - System should handle 130 requests in 1 minute
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

Use Case ID: MC-EMR015 – Sync iPad information with Local Server

* + Details:
    - Actor:
      * RLSU
    - Pre-conditions:
      * The RLSU must have been logged into the system.
    - Description:
      1. Use case begins when a RLSU presses the “Sync iPad Information”
      2. The Local Server sends a signal to the Client application
      3. The Client application will check if there is any dirty information.
      4. If there is any dirty information, the Client application packages up the information
      5. Use case ends when the Local Server receives the Client information and parses it.
    - Post-conditions:
      * The Local Server now has the Client response stored into the it’s SQLite DB.
  + Alternative Courses of Action
    - None
  + Extensions:
    - None.
  + Exceptions:
    - None
  + Concurrent Uses:
    - None
  + Related Use Cases:
    - None

---------------------------------------------------------------------------------------------------------

Decision Support

* + Frequency: On average 50 requests are made daily by a RLSU.
  + Criticality: High. If information cannot be synced between the Client applications then there will be an inconsistency between the information.
  + Risk: Medium. Implementation of this use case employs a third party API to communicate between iOS and Mac OSX applications

Constraints:

* + Usability
    - No previous training time
    - On average the user should take less than 13 seconds to complete the request
  + Reliability
    - 1% failures for every twenty-four hours of operation is acceptable.
  + Performance
    - Request should be sent and saved within 4 seconds
    - System should handle 130 requests in 1 minute
  + Supportability
    - The use case should be handled correctly by any iPad running iOS 6.0 or greater.
  + Implementation
    - iOS (Objective-C).

## 

## Appendix B - Use Case Diagram

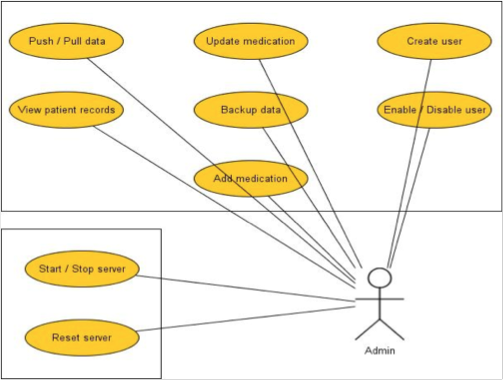


Figure 4.2.1

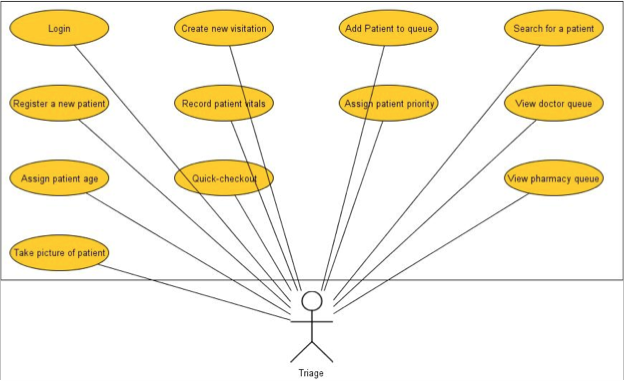


Figure 4.2.2

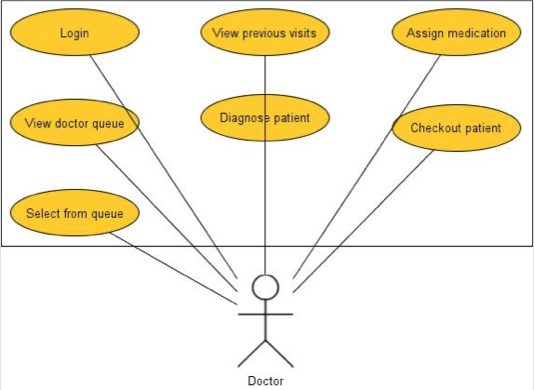


Figure 4.2.3

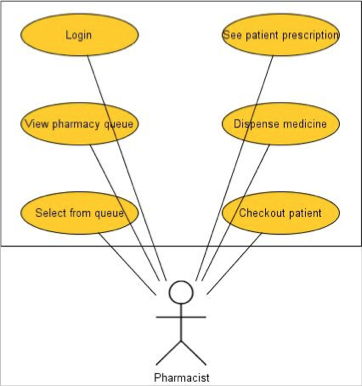


Figure 4.2.4

## Appendix C - Static UML Diagram

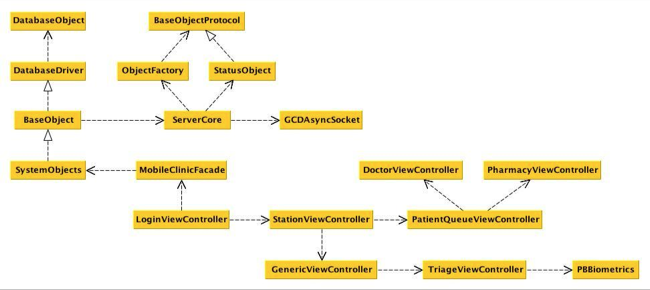


Figure 4.3.1

## Appendix D - Dynamic UML Diagram

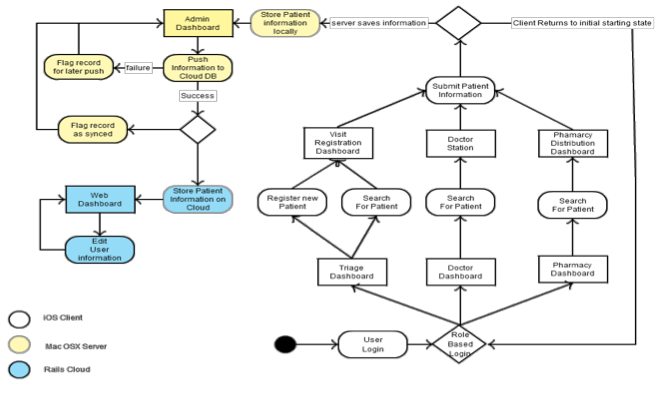
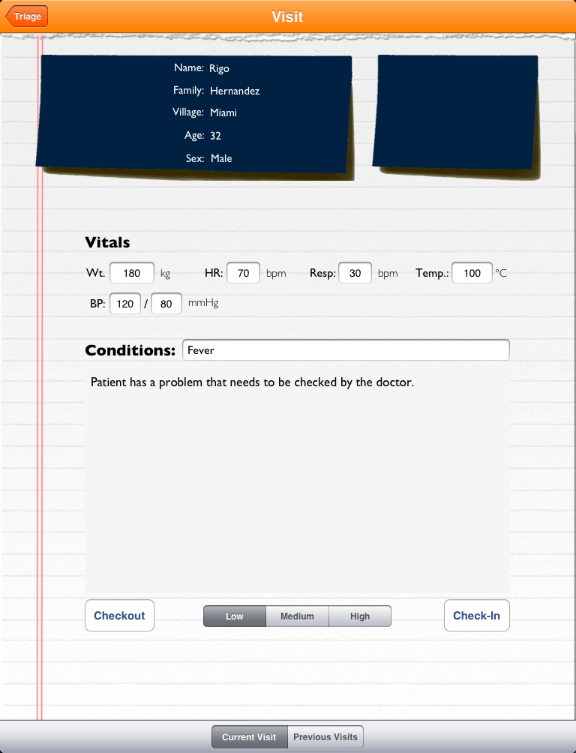
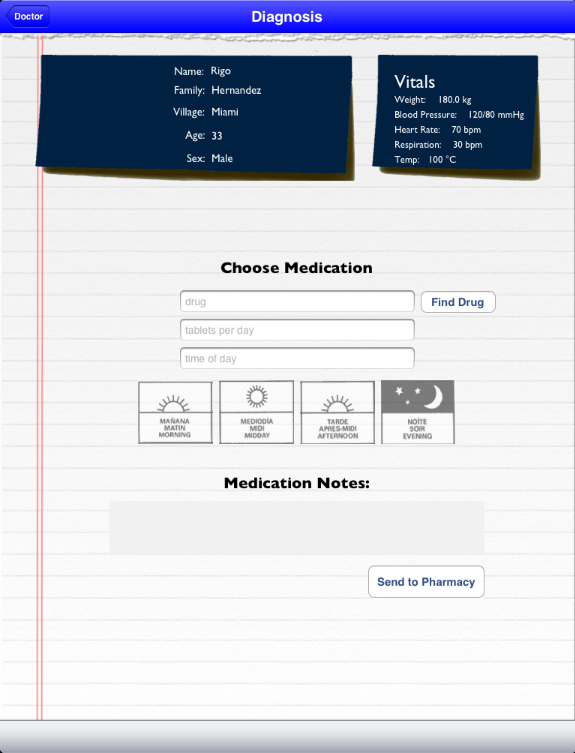


Figure 4.3.2

## Macintosh HD:private:var:folders:l_:63gtxm9n6md0z__23yvr805c0000gn:T:Messages:Transfers:Messages Image(423717536).pngAppendix E - User Interface Designs











## Appendix F - Diary of Meetings and Tasks

|  |  |  |
| --- | --- | --- |
| Meeting | 1 | 1/29/2013 |
| Attendance | Everyone |  |
| Review | Accepted Stories and Tasks |  |
|  | Unacceptable Stories |  |
|  | Time limit |  |
|  | Limited communication |  |
|  | Synchronization |  |
|  | Learning curve |  |
|  | Client-Server Implementation |  |
|  | Flojack API |  |
|  | Architecture Review |  |
|  | Acceptability Tests |  |
|  | Automated Tests |  |
|  | Story Map Refactoring |  |
| Next | Allow synchronization |  |
|  | Client and Server |  |
|  | iPad GUI |  |
|  | MVP of Triage |  |
|  | Login/Logout |  |
|  | Patient Creation |  |

|  |  |  |
| --- | --- | --- |
| Meeting | 2 | 2/5/2013 |
| Attendance | Everyone |  |
| Review | Accepted Stories and Tasks |  |
|  | Unacceptable Stories |  |
|  |  | Detailed Vitals |
|  |  | Sending patient data |
|  | Problems |  |
|  |  | Time |
|  |  | Synchronization |
|  | Solution |  |
|  |  | Git |
|  |  | Standard Schema |
|  | Architecture Review |  |
|  | iPad Client Architecture |  |
|  | OSC Architecture |  |
|  | WebApp Architecture |  |
|  | Story map Refactoring |  |
| Next | Diagnosis |  |
|  |  | Assign diagnosis |
|  | Sync iPad and Server |  |
|  | iPad GUI |  |
|  | MVP of Diagnosis |  |

|  |  |  |
| --- | --- | --- |
| Meeting | 3 | 2/12/2013 |
| Attendance | Everyone |  |
| Review | Storied attempted |  |
|  |  | Stories Removed |
|  |  | Stories Completed |
|  |  | In Progress |
|  | Tasks |  |
|  |  | Completed |
|  |  | In progress |
|  |  | Not initiated |
|  |  | Deleted |
|  |  | Removed |
|  | Architecture |  |
|  |  | Web Services |
|  |  | Client |
|  |  | Server |
|  | Internal Architecture |  |
|  |  | Presentation Layer |
|  |  | Objects |
|  |  | Base Object |
|  |  | Database |
|  | Class Diagram |  |
|  | Acceptability Tests |  |
|  | User Stories |  |
|  |  | I want to see all patient in system |
|  |  | Add medication to webApp |
|  |  | Update medication |

|  |  |  |  |
| --- | --- | --- | --- |
| Meeting | 4 | 2/26/ |  |
| Attendance | Everyone |  |  |
| Review | Stories Burn down |  |  |
|  | Tasks Run down |  |  |
|  | System Architecture |  |  |
|  |  | Client |  |
|  |  |  | iPad |
|  |  |  | Bonjour |
|  |  |  | CoreData |
|  |  | Web |  |
|  |  |  | Bootstrap CSS |
|  |  |  | jSQuery |
|  |  | API |  |
|  |  |  | Restful |
|  |  |  | Heroku |
|  |  | Server |  |
|  |  |  | MacBook |
|  |  |  | CoreData |
|  |  |  | OSX |
|  |  |  | Bonjour TCP |
|  |  |  | PostgreSQL |
|  | Acceptability Tests |  |  |
|  |  | Add visitation |  |
|  |  | Easy access to forms |  |
|  |  | See visitation records |  |
|  | Stories |  |  |
|  |  | Add medication through web |  |
|  |  | Update medication as pharmacist |  |

## 

# References