**Functional Specification**

*Overview*:

The application I will be creating is a mobile communication software for health care providers in a hospital setting. The goals of the application would be to:

1) Help organize information on a patient. Act as an extension to an existing Electronic Health Record System

2) Allow providers to communicate/chat to each other in the application for a specific patient.

3) Allow providers with appropriate roles to order and complete procedures.

4) Give providers a ToDo list for a given patient.

In addition to the main application, I will be creating a mock EHR system that will help generate patients and their appropriate information.

***Core Requirements***

1. **Azure Web Api** – I am choosing this type for my application mostly due to the fact of scalability in the future. If I were to pursue this application outside of class, I would not want to be limited to use solely through a mobile device. In addition, I could see a benefit of having web-based single page application to have access to some of the same controllers as this application. In addition, if ever put into use this type of application would have to rely on a heavy number of other cloud services such as an Electronic health record system, perhaps “Internet of Things” health care devices, and security systems.
2. **Azure SQL** – Will store all information for my app (more information in *Application Detail* section).
   1. Patient Level Information
   2. User/Role Information
   3. Biometrics on Patient
   4. Chat history between providers
   5. Diagnoses, Procedure types and other lists that the app will need.
3. **Rest Interfaces -** There will be a controller to access each of the information above (more information in *Application Detail* section).
4. **Authentication** – Will use twitter as authentication technique to meet the final project requirements
5. **Authorization** – All rest interfaces will be authorized and in addition only accessible to certain roles. Roles will be described further in the *Application* *Details* section. The roles will be:
   1. Hospital Administrator
   2. Health Care Provider: Physician
   3. Health Care Provider: Surgeon
   4. Health Care Provider: Support Staff
6. **Azure Web Jobs** – As part of the final project, I will be creating a web service that acts like an Electronic Health Record system. This service will update patient biometric status and other patient information from time to time. This web service will be done by a web job.
7. **Blob**– will use Blob storage to store imaging information for patients
8. **Secure Services** – Will use entity framework to help prevent SQL Injection attacks. In addition, will use Authorize decorator to lock down all costly resources.
9. **Universal App using C#**- I will be using this client type for the final project.
10. **Will Unit Test all Rest Services**

*Electives*

1. **Azure Notification Hubs** – Notification hubs will be used to alert the current user of any changes in a patient status (critical stable, or died)
2. **Azure Websites** – Will create a small dashboard page to show usage statistics of the application as well as statistics on the mock Electronic Health Record system.
3. **Azure Search** – Will utilize this service to allow Hospital administrators to quickly go through patients, check status of patients, check all unassigned patients, check all patients in critical condition.

***Application Details***

*Assumptions*

While the following assumptions do not hold in reality, they are made in order to make the application more manageable to design and implement

* A patient can never be re-admitted to the hospital
* The scope of possible diagnoses are ~100, and the scope of possible procedures are ~20
* The only relevant biometric information is the ones stored in our *Biometrics* table.
* An admitted patient only has one thing wrong with them (only get one diagnosis code)
* A patient will only have a change in status up to 10times a day.
* Diagnoses are not gender or age specific

*Details – Backend SQL*

1. **Provider** – represents application’s user (healthcare provider). Include twitter user id used to recognize user by their twitter login. Will include user role.
2. **Patient** - will represent a fake list of patients. Includes patient name, age, gender, medical status (discharged, dead, critical, stable), diagnosis, admittance date, and discharge date.
3. **Biometrics** – represents a patient’s biometrics for a particular admission. Includes blood pressure, glucose levels, Can be many biometric ratings per patient. (foreign key to *Patient*)
4. **ProviderPatient** – represents assignment of patients to users. Many to many relationship of *Provider* and *Patient*. . (foreign key to *Patient* and *Provider*)
5. **DiagnosisCode** – list of diagnosis codes possible for a patient
6. **ProcedureCode**  - list of procedure codes possible to be done and a category of what type of provider-role can complete the procedure.
7. **PatientProcedure**– represents procedures done on a patient. Many to one relationship with *Patient table* (foreign keys to *ProcedureCode* and *Patient*) This will serve as the to do list for the patient
8. **PatientChatLog** – represents the provider to provider chat log for a given patient. Each row represents a message. Includes message, date of message, patient id, and provider id. Many to one relationship with *Patient* (foreign key to *Patient*)
9. **PatientImaging** – represents the imaging documents for a patient. Includes image type column. Many to one relationship with patient. (foreign key to *Patient)*

*Details – Roles*

Non-User

Super User – this user has the ability (and only the ability) to generate new fake patients from the mock electronic health record system. This user is not meant to be a “real” role in the system, and therefore cannot access any other part of the application other than using their screen to “ask” for new fake patients

Application Users

1. Hospital Administrator – receives new patients assigned to them and must assign to an existing physician. If physician orders a surgery procedure, must assign to surgeon. Can view their assigned patients chat logs. Receive all of their assigned patient’s alert notifications.
2. Physician – can participate in their assigned patients chat logs. Can assign a diagnosis to a patient. Receive all of their assigned patient’s alert notifications. Can complete any “physician” type procedures. Can “order” any procedure. Can discharge patients. Can upload patient imaging. Can only have 5 patients assigned at a time.
3. Surgeon - can participate in their assigned patients chat logs. Receive all of their assigned patient’s alert notifications. Can complete any “surgery” type procedures. Cannot order any procedures. Can only have 5 patients assigned at a time.
4. Support Staff – can view and complete all outstanding “support” type procedures of all patients. Can participate in chat log, can view chat log. Cannot upload patient imaging. Cannot order any procedures.

*Details – Business Logic*

Mock Electronic Health Record System

In reality, this application would hook into an already existing electronic health record and requires a patient pool that has changing health status. Therefore, part of this application will include the ability to generate fake patients along with updating their biometrics during their hospital stay. The functions of this system are:

1. Per Super User request, generate a number of patients AND assign them to a hospital administrator (split the assignment evenly to all current users in the *User* table who are hospital administrators).
2. Every x number of minutes (decided with TA) the system will “roll the dice” to determine if a patient’s biometrics and medical status will be changed. Changes biometrics based on a random percent difference from last biometrics. Based on biometrics will change medical status to stable/critical/dead. Will only change each patient up to 4 times a day. Will only change patients that have not been discharged or are not already dead.

Rest API

1. Patient – all calls require authenticated user
   1. Create Fake Patients
      1. POST Call with number of patients to generate
      2. Only accessed by Super User Role
   2. Get Assigned Patients
      1. GET call with userId parameter
      2. Only accessed by hospital administrator, physician, and surgeon
      3. Returns patient names and medical status where medical status is NOT dead or discharged
   3. Get Single Patient
      1. GET call with patientId Parameter
      2. Returns Patient data along with ToDo List, Procedure History, and Provider Assignments, and PatientChatLog.
   4. Assign Patient to Provider
      1. PUT call with patientId as parameter and patient object with an assigned physician/surgeon
      2. updated AssignedDate on Patient Object to current DateTime
      3. Only accessed by hospital administrators
   5. Discharge Patient
      1. PUT call with patient id
      2. Only accessed by physician
      3. Updates patient with discharge as status code
2. Biometric – all calls required authenticated user
   1. Get Biometric Data for a patient
      1. GET Call with patient id parameter
      2. Only accessed by physician and surgeon
      3. Returns all of patients biometric data
3. Provider– all calls require authenticated user
   1. Get Providers
      1. GET Call
      2. Get Provider Name and their role
   2. Get Providers By Role
      1. GET Call with role parameter
      2. Get Provider Name and their role
4. Chat – all calls require authenticated user
   1. Get Message by Patient
      1. GET Call with patient id parameter
      2. Surgeons, physicians, and support staff can access
      3. Returns messages and messenger ordered from earliest to most recent
   2. Add Message in patient chat log
      1. POST call with message object and patient id
5. Procedure/Diagnosis
   1. Get items
      1. GET Call
      2. returns a list of procedure/diagnosis items used for drop downs in the interface
6. PatientProcedures – all calls require authenticated user
   1. Get procedures needed by patient
      1. GET call
      2. Returns all To Do items for logged in user
   2. Complete a Procedure item
      1. PUT call with PatientProcedure object
      2. Only completed by appropriate role
      3. Change Completed property on PatientProcedure Object to true
   3. Create PatientProcedure Item
      1. POST call with procedure code
      2. Creates a PatientProcedure object with procedure code and assigns to user depending on a few things
7. PatientImaging– all calls require authenticated user
   1. Create Image
      1. POST call with patientid and file object puts in blob
   2. Get Iamge List
      1. GET call with patientid as parameter
      2. Returns all documents for a given patient
   3. Get Image
      1. Get call with documentId
      2. shows image in universal app
8. Dashboard
   1. This will probably represent several different routes, but ultimately will return interesting aggregates of data that will be return to the Azure website

*Front End*

Universal App

1. Login Screen
   1. Allow users to login with Twitter.
   2. Allows anyone to register as any type of provider
2. Redirects to *Provider Home Page if Physician or Providers that need support procedures if Support Staff, or create fake physicians page if SuperUser*Provider Home Screen
   1. Hospital Administrator Home Page
      1. See a list of all “active” patients assigned to them through EHR
      2. Options to see list discharged patients or all dead patients that have every been assigned to them
      3. Filter list by their patients that currently do not have providers assigned to them or patients that need a surgeon for a surgery procedure.
      4. Can click/tap on any patient and go to *Patient Screen*
   2. Physician/Surgeon Home Page
      1. See a list of all “active” patients assigned to them
      2. Can click/tap on any patient and go to *Patient Screen*
   3. (moved to Patient Screen)Provider Support Home Page
      1. See a list of all procedures needed to be done by support staff along with the patient associated with the procedure.
      2. Can click/tap on a procedure to “complete” it. Can click/tap on a patient to go to *Patient Screen*
3. *Patient Screen*
   1. See patient name, age, gender, diagnosis code, medical status, and list of procedure codes (both done and to be done).
   2. See biometric readings of patient
   3. (image has its own page) (this is done on procedure page)Physicians can assign a procedure to the patient
   4. Link to *Home Screen*
   5. Link to *Image Screen*
   6. Link to *Chat Screen*
   7. Physician can change patient status to discharge
4. *Image Screen*
   1. Show list of images uploaded for the current patient.
   2. Clicking on document will bring user to new page to view that image on your device
   3. option to upload a document to back end
5. Procedure Screen
   1. Show current procedures
   2. Add procedures for appropriate roles
   3. Complete procedures for appropriate roles
6. *Chat Screen*
   1. Show all chat log with message, who posted message, and date time message was received
   2. Messages should be shown in a list ordered by when it was created (oldest chats first)
   3. Post a new message

Azure Web Site

In addition to the mobile app, I will create a single page dashboard web site using angularjs and d3 to show usage statistics of various transactions in this system.