Deliverable #2

Philip Gabardo, Thomas Woudenburg, Shane Noormohamed,
Matthew Paine, Joshua Barkovic, Jonathan Beadle, Jeff Buscarino

November 14, 2013

1 Introduction

1.1 Purpose

The purpose for this document is to establish the architectural design of the proposed medical application: "MedConnect". This will be achieved through object oriented analysis using UML diagrams. The intended audience for this document is the developers of the system.

1.2 System Description

MedConnect is a mobile medical records system that is used by doctors, nurses and patients. The system is designed to maximize security for the purpose of maintaining the confidentiality, integrity and availability of the information. The system stores data remotely in a digital format, reducing the need for paper usage to keep track of patient records. By maximizing the security and portability of user information, the system meets the users' needs for privacy while remaining efficient.

The system is used by doctors, nurses and patients to coordinate appointments and track relevant medical information about patients. MedConnect allows authorised users to view, create, modify, and remove patients, charts and appointments based on permissions. The system also includes a feature for doctors to plot certain patient data. All data is stored in an encrypted form. MedConnect restricts users from accessing certain information based on user permissions.

1.3 Overview

The next section of this document includes use case diagrams which illustrate the functional requirements of the system. The following section includes an analysis class diagram that will depict the overall design of the system. Next, the architectural design of the system will be defined explicitly. The final section includes Class Responsibility Collaboration (CRC) Cards for all classes included in the system. The appendix contains a signed document that confirms that contributions of each team member.

2 Use Case Diagrams

Any use cases with identical names are in fact identical use cases.

2.1 Administrator

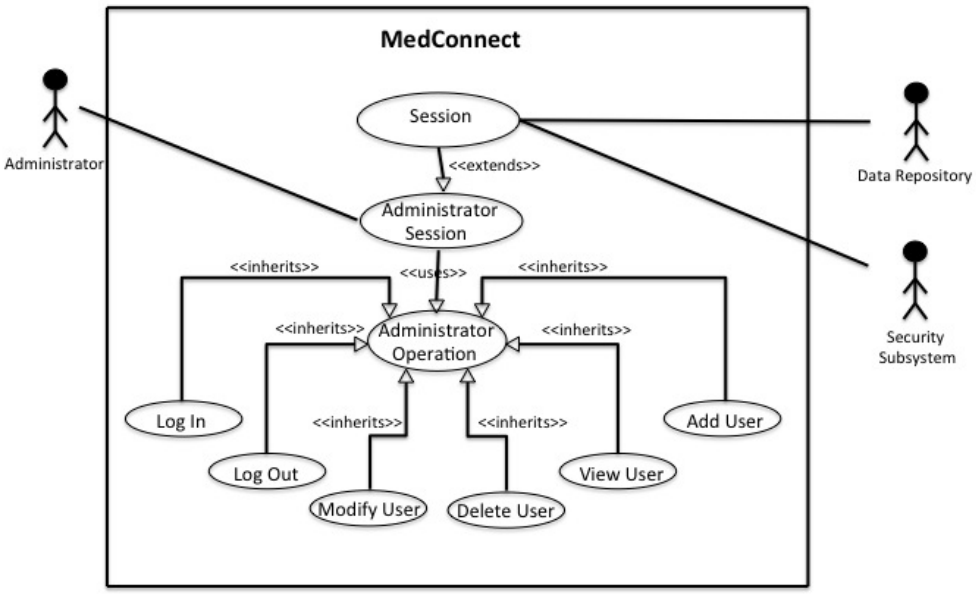

Figure 1: Administrator Use Case Diagram

2.2 Doctor

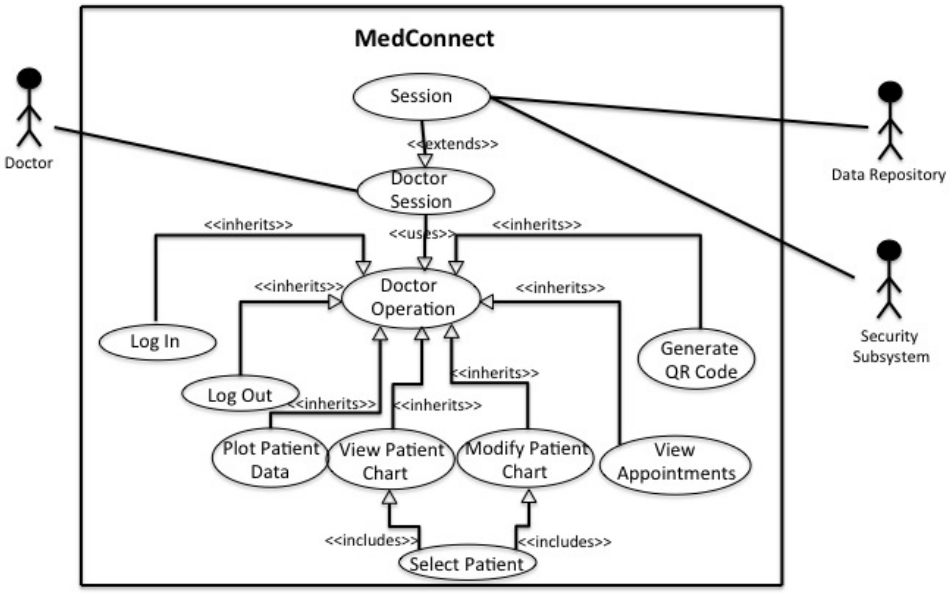

Figure 2: Doctor Use Case Diagram

Note: In the "Select Patient" use case, the doctor either manually enters the patent's name or scans the patient's QR code.

2.3 Patient

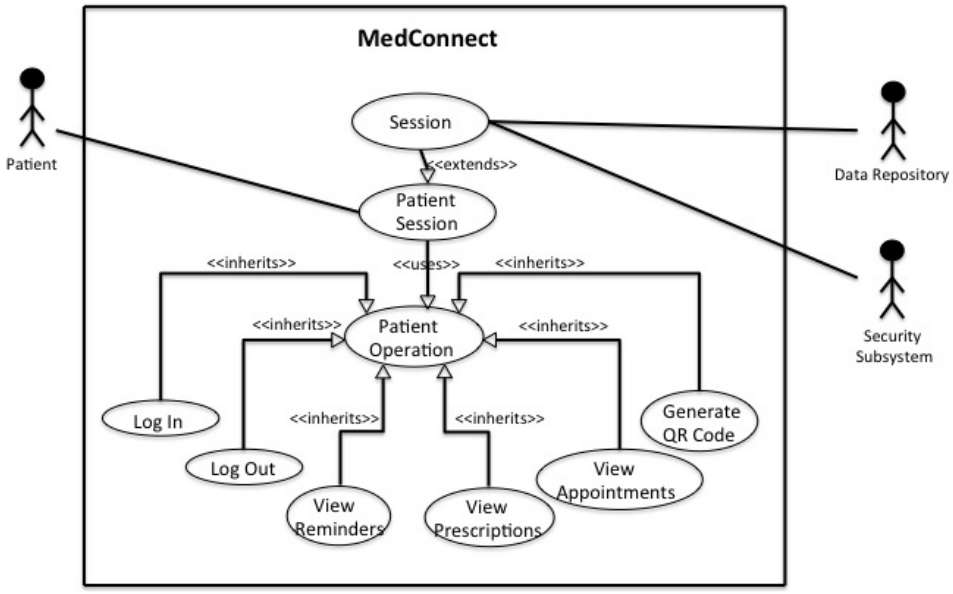

Figure 3: Patient Use Case Diagram

2.4 Nurse

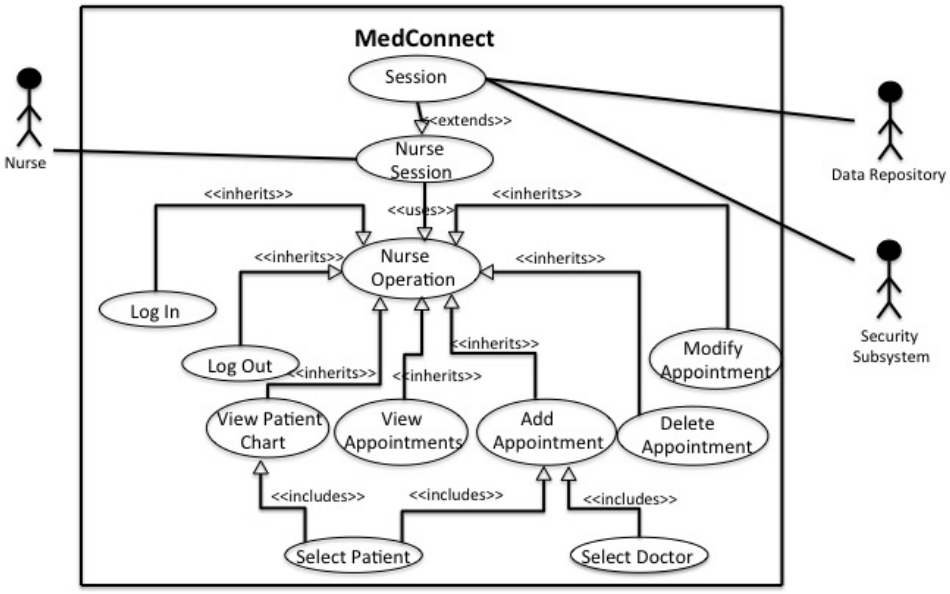

Figure 4: Nurse Use Case Diagram

Note: In the "Select Patient" use case, the nurse either manually enters the patent's name or scans the patient's QR code. In the "Select Doctor" use case, the nurse either manually enters the doctor's name or scans the doctor's QR code.

3 Analysis Class Diagram

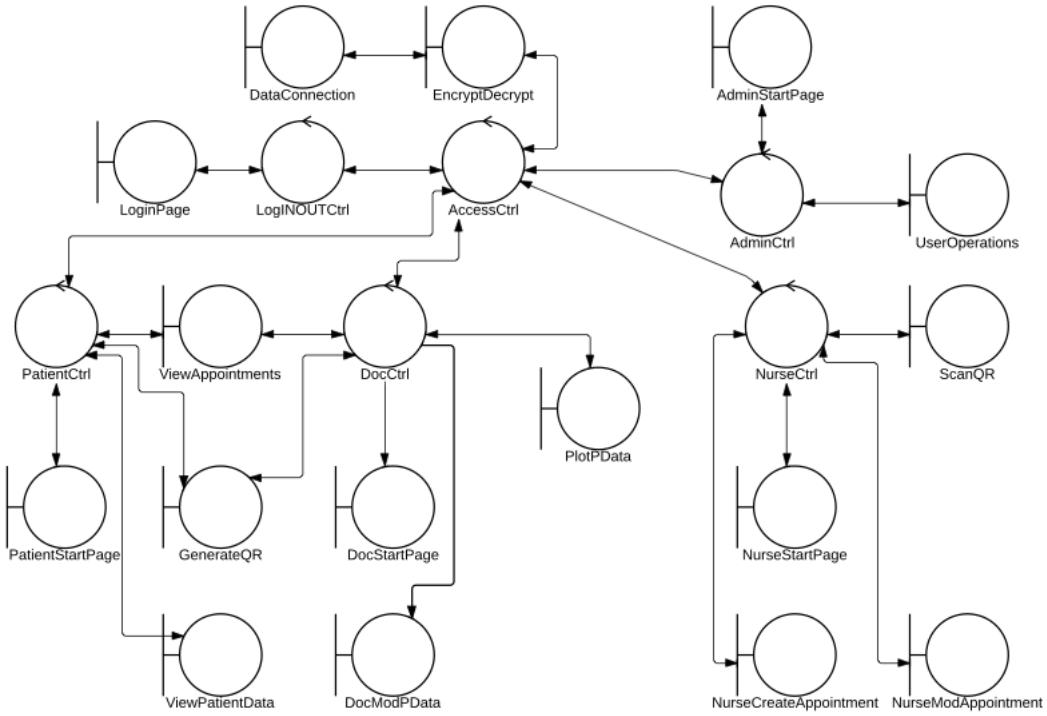

Figure 5: Analysis Class Diagram

4 Architectural Design

4.1 System Architecture

The overall system is best thought of as a combination of three sub-systems; the Data Store, the Security Subsystem, and the User Interface. Each of these sub-systems will require different Software Architectural Styles to fulfill their unique requirements as detailed in the following pages. The only generalization that can be made to the larger system as a whole is that it will be a predominantly layered architecture.

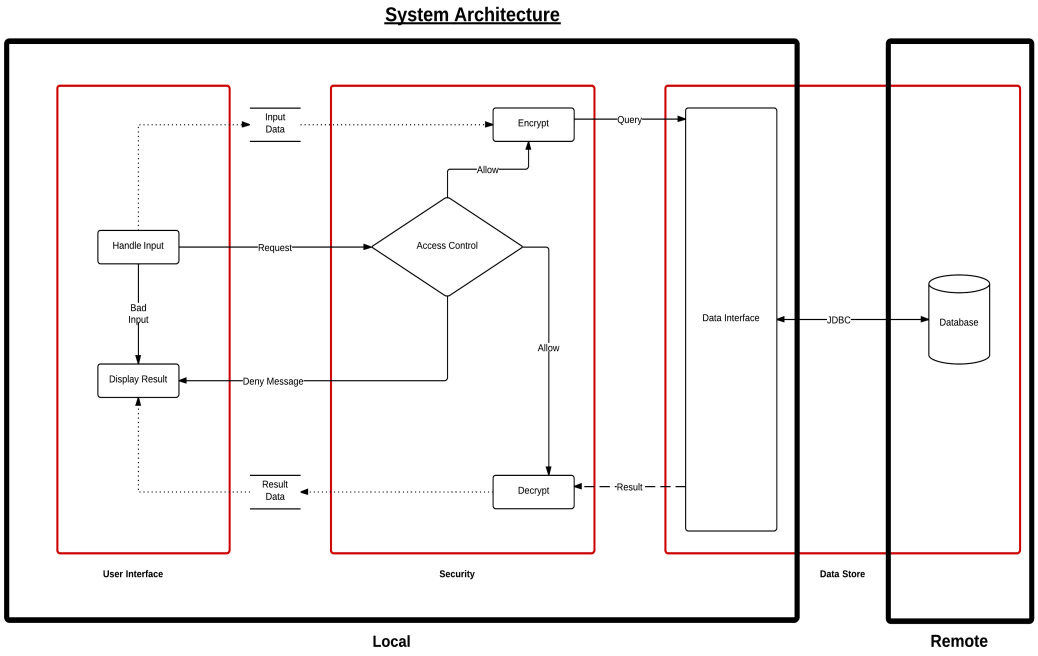

Figure 6: System Architecture

4.2 Subsystems

1. Data Store

   - Description: The Data Store system will include the physical data storage mechanism that stores the medical information this system is required to manage (see SRS Document), as well as the interface mechanism that provides an implementation-independent API between the aforementioned storage mechanism and the rest of the system. In our analysis class diagram, this subsystem is encompassed by the "DataConnection" class.
   - Architecture: The Data Store system will be designed adhering to a Repository Architecture style. This decision was made for the following reasons:
     - The role of this sub-system is predominantly the storage,

maintenance and retrieval of data and thus requires a Data-Centric approach to it's design.
  - This sub-system requires a fixed number of data storage elements and is required to interface with a variable number of actors placing simultaneous requests (see class diagram).

2. Security

   - Description: The Security sub-system acts as a transaction system between the User Interface and the Data Store sub-systems. In our analysis class diagram, this subsystem is encompassed by the "EncryptDecrypt" and "AccessCtrl" classes.There will need to exist a local instance of the Security sub-system for each user session. The Security sub-system has two main roles:
     - To selectively allow and dis-allow communication to and from the Data Store based on: the clearance level required to access the data that is being requested, and the clearance level of the user that has requested the data (based on the user's group designation).
     - To encrypt data all sent to the Data Store sub-system, and to decrypt all data received from the Data Store sub-system.
   - Architecture: This sub-system will be designed adhering to a Batch-Sequential software architecture style. This decision was made for the following reasons:
     - This is a Data-Centric sub-system.
     - All data that passes through the Security sub-system is encrypted/decrypted as a whole before it exits.
     - Only one unit of data will need to be processed by this system at a time (an instance of this system is created for each individual user session, thus only one request will be made at a time, and user will wait for the result) therefore a sophisticated form of pipe-lining is not required.

3. User Interface

   - Description:The User Interface sub-system handles all user requests entered into the system and displays the results of those

actions back to the user. In our analysis class diagram, this subsystem is encompassed by the all of the classes, with the exception of the "DataConnection", "EncryptDecrypt" and "AccessCtrl" classes.

- Architecture: The User Interface sub-system will be designed adhering to an Event Driven software architecture. This decision was made for the following reasons:
  - Although this system handles the same data as the other subsystems (entry and display), the primary role of this subsystem however is to handle user requests.
  - Since user requests are asynchronous and highly unpredictable. A design methodology adept at producing a system capable of handling these types of requirements reliably is required.

5 Class Responsibility Collaboration (CRC) Cards

Class: AdminStartPage	
Responsibility	**Collaborator**
Know AdminCtrl	
Handle button-click event of *Modify User List* button	AdminCtrl
Handle button-click event of *Log Out* button	AdminCtrl

Class: UserOperations	
Responsibility	**Collaborator**
Know username of new user	
Know password of new user	
Know personal information (ie. first name, last name, phone number, email) of new user	
Know group of new user	
Know username of account to modify	
Know new information of account to modify	
Know AdminCtrl	
Handle button-click event of *Add* button	AdminCtrl
Handle button-click event of *Update* button	AdminCtrl
Handle button-click event of *Delete* button	AdminCtrl
Display an error message if the username of a new user already exists in the system	
Display an error message if the username of an existing user doesn't exist in the system	

Class: AdminCtrl	
Responsibility	**Collaborator**
Know AccessCtrl	
Know AdminStartPage	
Know UserOperations	
Know LoginPage	
Handle logout request	DataConnection
Add a user to the system	AccessCtrl, DataConnection
Delete a user from the system	AccessCtrl, DataConnection
Modify an existing user in the system	AccessCtrl, DataConnection

Class: NurseStartPage	
Responsibility	**Collaborator**
Know NurseCtrl	
Handle button-click event of *Add New Appointment* button	NurseCtrl
Handle button-click event of *Modify Apppointment* button	NurseCtrl
Handle button-click event of *Log Out* button	NurseCtrl

Class: NurseCreateAppointmentPage	
Responsibility	**Collaborator**
Know NurseCtrl	
Know information of the appointment to be scheduled	ScanQR
Handle button-click event of *Add Appointment* button	NurseCtrl
Display an error message if the time of the appointment to be scheduled is in the past, or if the appointment to be scheduled conflicts with another appointment	

Class: NurseModifyAppointment	
Responsibility	**Collaborator**
Display information of appointment to be modified	
Know new information of appointment to be modified	
Handle button-click event of *Modify* button	NurseCtrl
Handle button-click event of *Delete* button	NurseCtrl
Display an error message if the new time of the appointment to be modified is in the past, or if the new appointment information conflicts with another appointment	

Class: ScanQR	
Responsibility	**Collaborator**
Decode a QR code	

Class: NurseCtrl	
Responsibility	**Collaborator**
Know AccessCtrl	
Know NurseStartPage	
Know AddAppointmentPage	
Know NurseModifyAppointment	
Know ViewPatientData	
Handle logout request	DataConnection
Add an appointment to the system	AccessCtrl, DataConnection
Delete an appointment from the system	AccessCtrl, DataConnection
Modify an appointment in the system	AccessCtrl, DataConnection
Retrieve an appointment from the system	AccessCtrl, DataConnection

Class: LoginPage	
Responsibility	**Collaborator**
Know username of user	
Know password of user	
Know LogINOUTCtrl	
Handle button-click event of *Submit* button	LogINOUTCtrl
Display an error message if the user's username-password pair does not exist in the system	

Class: LogINOUTCtrl	
Responsibility	**Collaborator**
Handle login request	
Know LoginPage	
Know AccessCtrl	
Create AdminCtrl	
Create PatientCtrl	
Create NurseCtrl	
Create DocCtrl	

Class: AccessCtrl	
Responsibility	**Collaborator**
Determine access permissions of user	

Class: EncryptDecrypt	
Responsibility	**Collaborator**
Encrypt data	
Decrypt data	
Know encryption algorithms	
Know decryption algorithms	

Class: DataConnection	
Responsibility	**Collaborator**
Verify the existence of a username in the database	
Add a user's username, password, group and personal information to the database	
Update a user's password, group or personal information in the database	
Delete a user's username, password, group and personal information in the database	
Add a patient's chart to the database	
Update a patient's chart in the database	
Delete a patient's chart from the database	
Add an appointment to the database	
Update an appointment in the database	
Delete an appointment from the database	

Class: DocStartPage	
Responsibility	**Collaborator**
Know DocCtrl	
Handle button-click event of *View Patient Chart* button	DocCtrl
Handle button-click event of *Modify Patient Chart* button	DocCtrl
Handle button-click event of *View Appointments* button	DocCtrl
Handle button-click event of *GenerateQR* button	DocCtrl
Handle button-click event of *Plot Patient Data* button	DocCtrl
Handle button-click event of *Log Out* button	DocCtrl

Class: DocModPData	
Responsibility	**Collaborator**
Know DocCtrl	
Know patient	ScanQR
Know updated information for patient chart	ScanQR
Handle button-click event of *Modify* button	DocCtrl

Class: PlotPData	
Responsibility	**Collaborator**
Know DocCtrl	
Know patient	ScanQR
Plot patient's height and weight over time	

Class: GenerateQR	
Responsibility	**Collaborator**
Know user	
Generate a QR Code encoding the user's username	

Class: DocCtrl	
Responsibility	**Collaborator**
Know AccessCtrl	
Know DocStartPage	
Know PlotPData	
Know ViewPatientData	
Know DocModPData	
Know ViewAppointments	
Handle logout request	DataConnection
Update a patient chart in the system	AccessCtrl, DataConnection
Retrieve a patient chart from the system	AccessCtrl, DataConnection
Retrieve a doctor's appointments from the system	AccessCtrl, DataConnection

Class: ViewPatientData	
Responsibility	**Collaborator**
Know patient	ScanQR
Display patient's chart	

Class: PatientStartPage	
Responsibility	**Collaborator**
Know PatientCtrl	
Handle button-click event of *View Prescriptions* button	PatientCtrl
Handle button-click event of *View Appointments* button	PatientCtrl
Handle button-click event of *View Reminders* button	PatientCtrl
Handle button-click event of *Generate QR Code* button	PatientCtrl
Handle button-click event of *Logout* button	PatientCtrl

Class: PatientCtrl	
Responsibility	**Collaborator**
Know AccessCtrl	
Know PatientStartPage	
Know ViewPatientData	
Know ViewPrescriptionsPage	
Know ViewAppointmentPage	
Know ViewRemindersPage	
Handle logout request	DataConnection
Retrieve a patient's prescriptions from the system	AccessCtrl, DataConnection
Retrieve a patient's appointments from the system	AccessCtrl, DataConnection
Retrieve a patient's reminders from the system	AccessCtrl, DataConnection

Class: ViewAppointments	
Responsibility	**Collaborator**
Display patient's appointments	

Appendix: Contributions

Work Objectives:

1. System Functionality (Figuring out how the system will work)
2. LaTeXFormatting
3. Document Editing
4. Section 1: Introduction
5. Section 2: Use Case Diagrams
6. Section 3: Analysis Class Diagrams
7. Section 4: Architectural Design
8. CRC Cards

Member Contributions:

Matt Paine: Use Case Diagrams (13%), Analysis Class Diagram (13%), CRC (15%)

Josh Barkovic: Use Case Diagrams (13%), Analysis Class Diagram (13%), Architectural Design (100%), CRC (15%)

Jeff Buscarino: Use Case Diagrams (13%), Analysis Class Diagram (13%), CRC (15%)

Thomas Woudenberg Use Case Diagrams (13%), Analysis Class Diagram (13%), CRC (15%)

Shane Noormohamed: Use Case Diagrams (13%), Analysis Class Diagram (13%), CRC (15%)

Jonathan Beadle: Use Case Diagrams (13%), Analysis Class Diagram (22%), CRC (15%)

Philip Gabardo: Use Case Diagrams (22%), Analysis Class Diagram (13%), Introduction (100%), LaTeXFormatting (100%), CRC (15%)

Note: The amount of entries in each members contribution list is not necessarily a good depiction of how much work they have completed. As some work objectives were much larger than others.