Department of Systems and Computer Engineering
Faculty of Engineering and Design

Security Threat Modeling for IoT-Enabled e-Health Systems

SYSC 4907 Fourth Year Project

Supervising By:
Professor
Mohamed Ibn Khala

Performed By:
Yohannes Kussia
Tinsae Debele
Mohammed Ismail
Kamran Sagheir

4/14/21

Abstract

Health care is one of the most important sectors in the world, with many confidential information transmitted and stored in the system. A security system that is thoroughly designed and implemented, will supply confidentiality, integrity and availability of information in an E-health system. The lack of an adequate security system that is vulnerable to threats coming from adversaries, can cause devastating consequences. Considering that health-care systems have sensitive information and the consequences that would arise from poor security, the group designed and implemented the project titled “Threat modelling for E-health systems". The group’s objective was to implement a plug-in on java eclipse, using a platform called OM2M, which will be discussed in the later sections of the report. The report will discuss, the importance of security for an E-health system, the design of a security system plug-in and its implementation. The group performed excellent research in figuring out the right devices and system design to implement the security plug-in. The group also researched about the software platform being used and the software designs that would be proper for the project. This project was designed based on the knowledge gained from prior engineering and security knowledge as well as research done mainly on the eclipse website, om2m forums and other plug-ins made in the past. Use cases were made for the scenarios that would occur in a health care system, such as registration, logging in the system and supplying access control. Based on this use-cases the group designed the system using architecture diagrams for the hardware and UML diagrams for the software being implemented. The group then implemented a security plug-in that has capabilities of providing confidentiality, through encryption and decryption of data, authentication, and access control. The results were tested using an interface, which was

also implemented by the group. The data transmission of the Restful API was tested using Postman, which shows the content of the headers and body being transmitted. A significant project and scope management was done to successfully complete the project. The project is believed to provide confidentiality, authentication, and access control as proposed in the beginning. It can be added on top of any system, particularly an E-health system. This will give the staff and patients, confidence on the security of information and the medical workers can focus on what they do, which is helping and saving lives of many patients.

Acknowledgment

We would like to thank Professor Mohamed Ibn Kahla as well as Abdelrahman Eldosouky and Ali Farhat for helping us though out the semester and guiding us when we needed it.

Table of Contents

Abstract 1
Acknowledgment 3
Table of Figures 7
List of Tables 7
1. Introduction 8
2.1 Kamran Sagheir 10
2.2 Tinsae Debele 12
2.3 Mohammed Ismail 13
2.4 Yohannes Kussia 14
3. Background Information 17
4. Use Cases 19
4.1 Use case diagram 19
4.2 System sequence diagram 22
5. Design Requirements 24
5.1 Functional requirements 24
5.2 Non-functional requirements 24
5.3 Interface requirements 25
6. System Architecture 26
6.1 Architecture diagram 26
6.2 Mapping Subsystems to Hardware 28
6.3 Control flow and Network protocol 29
7. Software Design 34
7.1 UML with description 34
7.2 Interface 36
7.2.1 Logging Interface 36
7.2.2 Registration interface 37
7.2.3 Dashboard interface for patient Health History 38

7.3 Integrations of security system with E-health System 39
8. Security 40
8.1 Authentication design 40
8.1.1 Authentication implementation 40
8.1.2 Encryption and Decryption 42
8.2 Access Control 45
8.2.1 Security Policy 45
8.2.2 Access control design 47
8.2.3 Access control implementation 49
8.3 Database management and sensitive data storage 52
8.3.1 Patient's Personal Information 52
8.3.2 Patient Medical Information 52
8.3.3 Staff Information 53
8.3.4 Login Credentials 53
9. Testing and Results 54
10. Project management 59
10.1 Project Network Diagram 59
10.2 Schedule/timetable description 59
11. Conclusion 60
References 61
Appendix A 66
Project Code 66
Add Info Panel 66
Dashboard 69
Edit Info Panel 72
Get Info Panel 73
Get Info Panel Ph 76
Login Frame 79

Dashboard Ph 83
Signup Frame 87
Activator 90
Data Phaser 91
ObixUtil 93
Operations 95
Password 96
Read XML File 98
Request Sender 99
Smart Constraints 100
Smart Controller 101
Smart Cycle Manager 103
Smart Router 105
Status Record 106
Trial 109
Access Controller 110
Authentication Controller 111
Encryption and Decryption 112
LinkedList 113
Authentication 115
Access Control 117
Appendix B – Project Proposal 118

Table of Figures

Figure 1: Registration on the Om2m Security System....19
Figure 2: Use case showing Authentication on the om2m security system. 20
Figure 3: Use case showing Access Control in the System. 21
Figure 4: System Sequence Diagram....23
Figure 5: Smart E-health Architecture Diagram....26
Figure 6: UML diagram for Smart E-Health 28
Figure 7: Om2m componet Diagram [25] 30
Figure 8: Resource Creation Scenario [24] 31
Figure 9: Smart E-health Resource Tree 32
Figure 10: Hardware Requirements and set up 33
Figure 11: UML with description 35
Figure 12: Login interface 36
Figure 13: Registration interface 37
Figure 14: Dashboard interface for patient registration 38
Figure 15: Patient information Retrievable interface 38
Figure 16: UML diagram of smart e-health security in OM2M....39
Figure 17: Creating a salt key. 41
Figure 18: Hashing the password and the salt. 41
Figure 19: Difference between ECB and CBC [29]. 44
Figure 20: Access Controller Code....46
Figure 21: LinkedList Code 50
Figure 22: MVC design of security system. 51
Figure 23: Adding information about a patient. 54
Figure 24: retrieving and editing patient information. 55
Figure 25: Registration GUI 56
Figure 26: Login GUI 57
Figure 27: Project Network Diagram 59
Figure 28: Schedule/timetable description 59

List of Tables

Table 1: Difference between AES and DES [28]. 42
Table 2: Access control and capability list....48

1. Introduction

The domain of Internet of Things (IoT) deals with Machine-to-Machine (M2M) communication. The goal of M2M communication is to form links among billions of devices capable of interacting with each other. For M2M communication, the European Telecommunications Standards Institute (ETSI) released specifications. An open-source project named, “OM2M” was created. OM2M supplies a platform for M2M communication that is ETSI-compliant.

One of the most prominent areas of the IoT is Electronic Health Systems (E-Health Systems). OM2M is widely used for communication, among the IoT-enabled E-Health Systems [1]. The key role of IoT in E-Health Systems is the ability to monitor, record and transmit data related to patients, doctors, nurses, and family members [2]. The data is transmitted from E-health systems to hospital servers over wireless networks. The transmitted data, makes it easy for health care professionals to respond to patients in case of emergencies.

This prompt response to emergencies, can save the lives of the patients, during the time patients are dealing with a life-threatening situation [2].

One of the major concerns in the field of IoT, is the security of data. The data of E-Health Systems can be lost and tampered with during the transmission. The data can lose its validity, confidentiality, and authenticity. As a result, during emergencies and under normal circumstances, the data is delayed. This can result in denial of service, data loss and other threats. This can also lead to elevated risks, such as the mortality of patients. In this project, the team has devised a solution to reduce such security threats, by creating security tools as a plugin that can be deployed in any environment to supply security features for

communications over OM2M. The plugin is generic in its nature and it supplies a security layer for the OM2M platform. The security layer makes sure that the data being transmitted and retrieved by the IoT enabled E-Health system, is secure and trustworthy.

A RESTful API is used to carry out the communication. The communication happens over the Hypertext Transfer Protocol Secure (HTTPS). HTTPS is composed of Hypertext Transfer Protocol (HTTP) along with the Transport Layer Security (TLS) protocol, hence is a secure protocol. The security layer supplies tools that allow for the encryption of data before it leaves the E-Health device and decryption of data when it reaches the destination. The security layer also provides Role Based Access Control (RBAC). The RBAC allows machines and users with various levels of authorization to get registered with the system. Based on this level of authorization, they get access to different information that is relevant to them. The distribution of data based on the level of authorization makes sure that only the people with correct authorization get access to the data. This prevents from illegal access to data and data tampering.

2. Individual contributions

All the team members worked on understanding the architecture, design and implementation of the OM2M platform. The scope of this research involved understanding the OM2M terminology, the functionalities and tools that are provided by the OM2M platform by default and designing and implementation of OM2M gateways (MN-CSE), and server (IN-CSE).

2.1 Kamran Sagheir

The research, designing and implementation was done in collaboration with Tinsae Debele.

2.1.1 Research

In addition to the research on the OM2M platform, I conducted research about addition of plugins to the OM2M platform, in the form of a library that will be completely generic in its nature. This research led me towards having a better understanding of how the team needs to design and implement the solution so that it is compliant with the OM2M platform. I also conducted research on forming connections among different IoT enables e-Health systems using the OM2M platform and transmission of data from one node I.e., device to another. This led me towards better understanding of maven (a tool for java projects that automates the build process), RESTful API's, Postman (an open-source application that automates the manual testing for the RESTful APIs), the gateways (MN-CSE's) and the server (IN-CSE) and the tools that can be used for the testing of plugin.

2.1.2 Design

During the design phase, key emphasis was put on making sure that system for the data transmission and setting up connections was completely generic and irrespective of the underlying device. To achieve this goal, I used RESTful APIs as underlying structure to send data. RESTful APIs were used as they are generic in their nature and are compliant with the OM2M platform. To form connections among different devices connected via OM2M platform, I made the plugin in such a way that it could be added as a maven module to the OM2M parent project. This allowed for the flexibility and genericity that team was looking to achieve with the plugin. The plugin can be started and stopped , just by running a simple command on the command line interface.

2.2.3 Implementation

For the implementation phase, I started by making the plugin using maven and added it to the OM2M parent project as a module. This made it significantly easy for the E-Health devices to use the plugin, as it became part of the parent OM2M project and every device that used the OM2M platform, could use the plugin by default. Next part was the transmission of data. For this part, RESTful APIs were used. The code was written in such a way that when the data was ready to be transmitted from the E-Health device (node) to the server (IN-CSE), after the plugin performed its security features, RESTful API did a POST request. As a result, the data got transmitted to IN-CSE and was processed. For the testing part, postman was used that allowed for the testing of the gateways and the server to make sure that the plugin was working properly.

2.2 Tinsae Debele

The research, designing and implementation was done in collaboration with Kamran Sagheir.

2.2.1 Research

The main objective of my research was to determine the best design and methods of implementing a secure H-health System using a horizontal platform, OM2M. Also, my research covers M2M service requirements, the functional architecture, communication interfaces and how to interwork with existing OM2M standard and technologies. The first stage of the research focused on the OM2M platform implementation, which leads to diverse ways of implementation methods. As result, The E-Health System can be implemented using a plugin to on the OM2M platform. The resource creation and allocation, begin by using the fundamental CORE plugins of the OM2M platform. As a result, I researched how OM2M provides a RESTful Capability Layers accessibility via an open interface to enable the development of service and application independently of the underlying network. Furthermore, I focused on how the security system can be implemented independently of the E-Health resource.

2.2.2 Design

Based on my research above, I took the responsibility of design the E-Health system using the CORE OM2M plugin as a dependence. As result, I start to design an E-health system plugin that uses SCL work as an interface between access and the application domain. Besides, to exchange information between the device, we come up with a design that allows a device or a Gateway application to access the infrastructure node. This design allows the interface well-

defined network management function enables to manage the access and core network. Finally, I designed a GUI (Graphical user interface) that allows user to interact with the system.

2.2.3 Implementation

The first step of the implementation starts with build the system requirement for OM2M, which leads me to implement the resource Required for our E-health system. I used the OM2M core system and build the E-Health system based on the design above. Implemented a SCL_Service interface to handle generic RESTful request. Besides, I integrated the security system plugin during the same time. The implementation led me to implement a GUI interface of efficient information exchange between the servers and the gateway, device.

2.3 Mohammed Ismail

The research, designing and implementation was done in collaboration with Yohannes Kussia.

2.3.1 Research

In addition to the research on the OM2M platform, I research on the type of authentication and database is needed and works best for this platform. Part of the research involved knowing how to encrypt passwords using salt and the encryption of the communication between the user and the server. Second part of that research was about finding out which database is best suited for om2m and our requirements for a database, which involved a couple of things like it being a pure java database, being automatically encrypted for more security, and being compatible with ODBC driver.

2.3.2 Design

When designing the authentication of the system I wanted to make sure it would work with om2m plugin we are constructing, since if it was built without it being compatible, I would need to build the system again which would take up more significant amount of time. To help solve that problem I made a script that would be compatible with almost any function. For the database I decided to go with the H2database since it works best with our projects needs and has been proven effective in plugins we already saw built by others.

2.3.3 Implementation

When I was implementing, I did the authentication section then moved to building the encryption and decryption part together and finally moved on to working at the database part, after finishing all my sections I helped my other teammates incorporate it with the plugin of om2m by explaining any part that is not easily understood or suggesting on where to put it in the om2m system. I also did changes to the om2m security code when needed so that it would fit best with our system, after finishing that part I mainly helped building the plugin and other security part sections when my teammates needed my help since I had more previous knowledge in java.

2.4 Yohannes Kussia

The research, designing and implementation was done in collaboration with Mohammed Ismail.

2.4.1 Research

In addition to the research on the OM2M platform, I also contributed by researching about OM2M's access control policy to understand how to design and implement the access control on an OM2M infrastructure node. Not only that but also, I researched about the right data structures such as access control lists and capability lists needed to use to store the actors, resources, and policies as well as about Role Based Access Control (RBAC). I also researched about OM2M built in access control policy and how to use it side by side with the access control to be implemented.

2.4.2 Design

In the design phase my task was to apply the research I did and my design as well as my organizational skills. I was heavily involved in the software design than the system design. I chose an MVC software design pattern in order to be able to implement the security plug-in. The choice was due to the requirement of the use of an interface. The view will implement the interface, the model will have the data of the security system and the controller updates the view. I also designed the sequence of the system and the software. I designed a linked list data structure to provide functionalities as an access control list. I did a high-level design of the RBAC to be implemented using diagrams and models such as the bell lapadula model which provides confidentiality and access control.

2.4.3 Implementation

I was involved in the access control part of the implementation phase. I implemented the access controller.java, Linkedlist.java, access control.java and the view that is updated based on

the role of the user. I implemented the data structure for the access control and the RBAC logic, in order to grant access based on the role of the user. The access control policy was also implemented in this part of the code as a form of data structures stored in a list. I added the configuration of the access control policy in the resources as part of a backup access control. This was done by understanding the horizontal platform and designing a modular scheme that can incorporate a third-party access control method to go hand in hand with the one that I already implemented.

3. Background Information

The security of E-Health Systems is very crucial. There were 417 health security breaches in the year 2016 that were publicly disclosed. The breaches exposed healthcare records of more than 3.2 million people according to Privacy Right Clearinghouse [3]. In the United States alone in 2017, records of 5.579 million patients were affected by various data breaches [4]. In the case of a security breach, electronic and other digital information related to the health of the patient is compromised. Hence, privacy and security are major concerns in electronic healthcare [5-7]. According to Cisco, 50 billion devices will be connected to the Internet by the year 2020 and this will outnumber the projected human population of 7.7 billion, by five to one. This makes security threats even higher [8-9]. Fortunately, that projection was not accurate. However, there are more connected devices today, as compared to a decade ago. Information Handling Services (IHS) Markit Ltd. predicted that there will be 17.6 billion connected devices and managing users 'privacy is a major challenge for such a number of connected devices [10]. Some of the solutions that have been put for the IoT enable E-Health Systems threat management and prevention, are limited due to reasons such as less power availability for the devices, low memory, and low processing speed [11-12]. The threats related to IoT security include eavesdropping, masquerading, routing attacks, denial of service, data modification, man-in-the-middle, user compromise, hard-ware compromise, software compromise, data replay, data fabrication, data privacy, and authentication attacks [13-18]. As a result of these attacks, the patient's private health-related information is exposed to the public. This information can be used for insurance-related frauds and can also cause the death of the patient if the patients' medical data is manipulated [13][15][19]. The IoT-enabled E-Health Systems such as

smartwatches, arable electroencephalography (EEG) headsets and wireless electrocardiogram (ECG) necklaces are usually sensor-based and communicate over a wireless network [13] [15] [17] [20]. According to research, the IoT-enabled e-Health systems are mostly used by the elderly, and chronically ill patients having diseases such as blood pressure and heart issues, among others [14] [15] [17] [20] [21]. “A systematic and structured security technique used to identify the security objectives, threats and vulnerabilities of an application, to help make design and engineering decisions, and determine where to prioritize efforts in designing, developing, and deploying secure Applications” is called Threat Modeling [22]. For the determination of security and privacy threats at the initial stages, understanding the requirements to secure the system, and developing better security systems, the threat modelling has proven to be phenomenally successful [23]. All this information indicates that any breach in data, loss of data or tampering with the data for IoT-enabled E-Health Systems can lead to severe consequences. Therefore, a security system that compromises between the security, cost and efficiency of the system need to be designed and implemented to provide sufficient security without altering or affecting the system negatively in a cost-effective manner.

4. Use Cases

The following section shows the different use cases for the IoT security system design. It has use case diagrams for different scenarios of the system which include registration, authentication and access control.

4.1 Use case diagram

This shows a use case diagram for the registration phase. The first part is the user inputting their credentials and role on their device which will run on the Middle Node (MN-CSE). After that, the information is sent to the Server (IN-CSE) through a secure IPsec. The server encrypts the data and other information and stores it into the database.

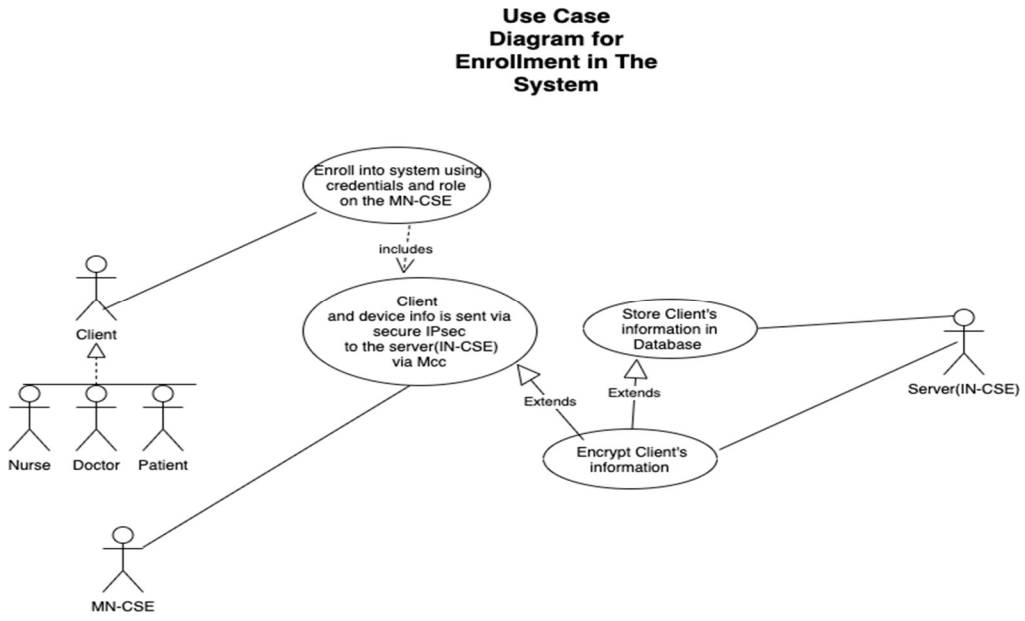

Figure 1: Registration on the Om2m Security System

The following use case diagram shows the authentication phase of the security system. The client logs into the system using the credentials and role. This information is sent via a secure

IPsec to the server. The Server compares the encrypted password with the stored credential in the database. The database uses a salt to enable clients to have the same passwords for different users. Using the salt and the password the server evaluates the credentials and sends the grant access notification to the MN-CSE. The MN-CSE will then grant access to the client.

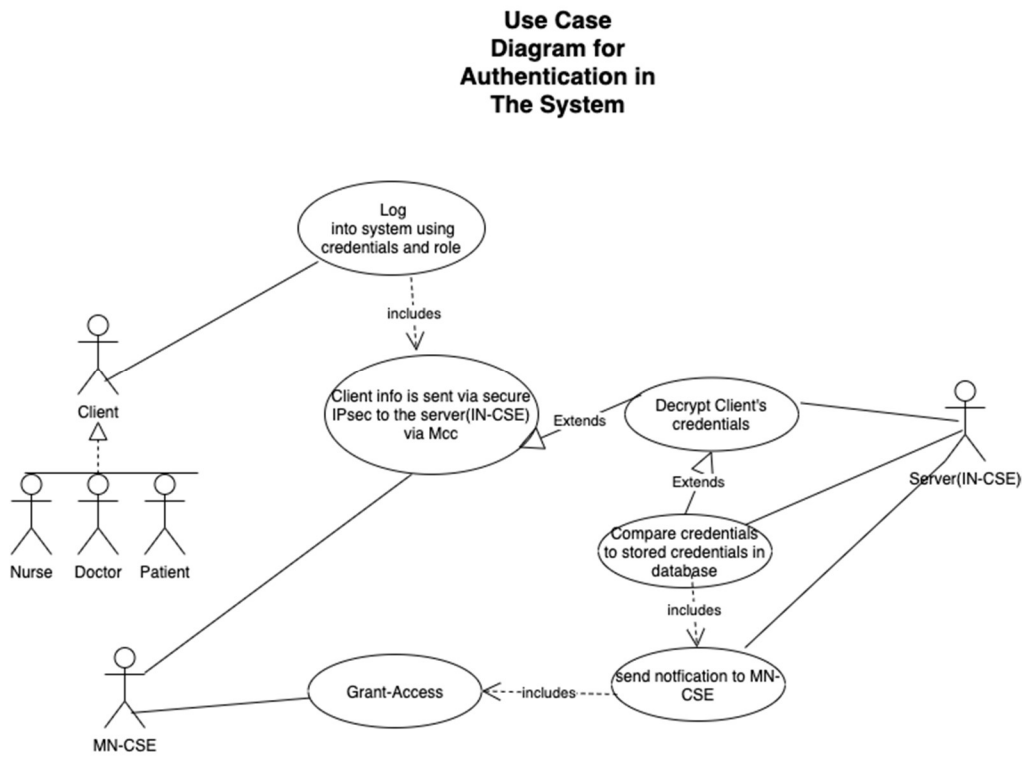

Figure 2: Use case showing Authentication on the om2m security system.

The following use case diagram shows the access control phase of the security system. After the client logs in and is granted access, since the client initially registered using their role, their credentials and role are stored together in the database. Therefore, when a client requests a resource, the request is sent via IPsec to the server. The server checks the role and performs a

role-based access control using the access control policy in the server. The resource is sent towards the MN-CSE and then towards the device/client requesting the resource.

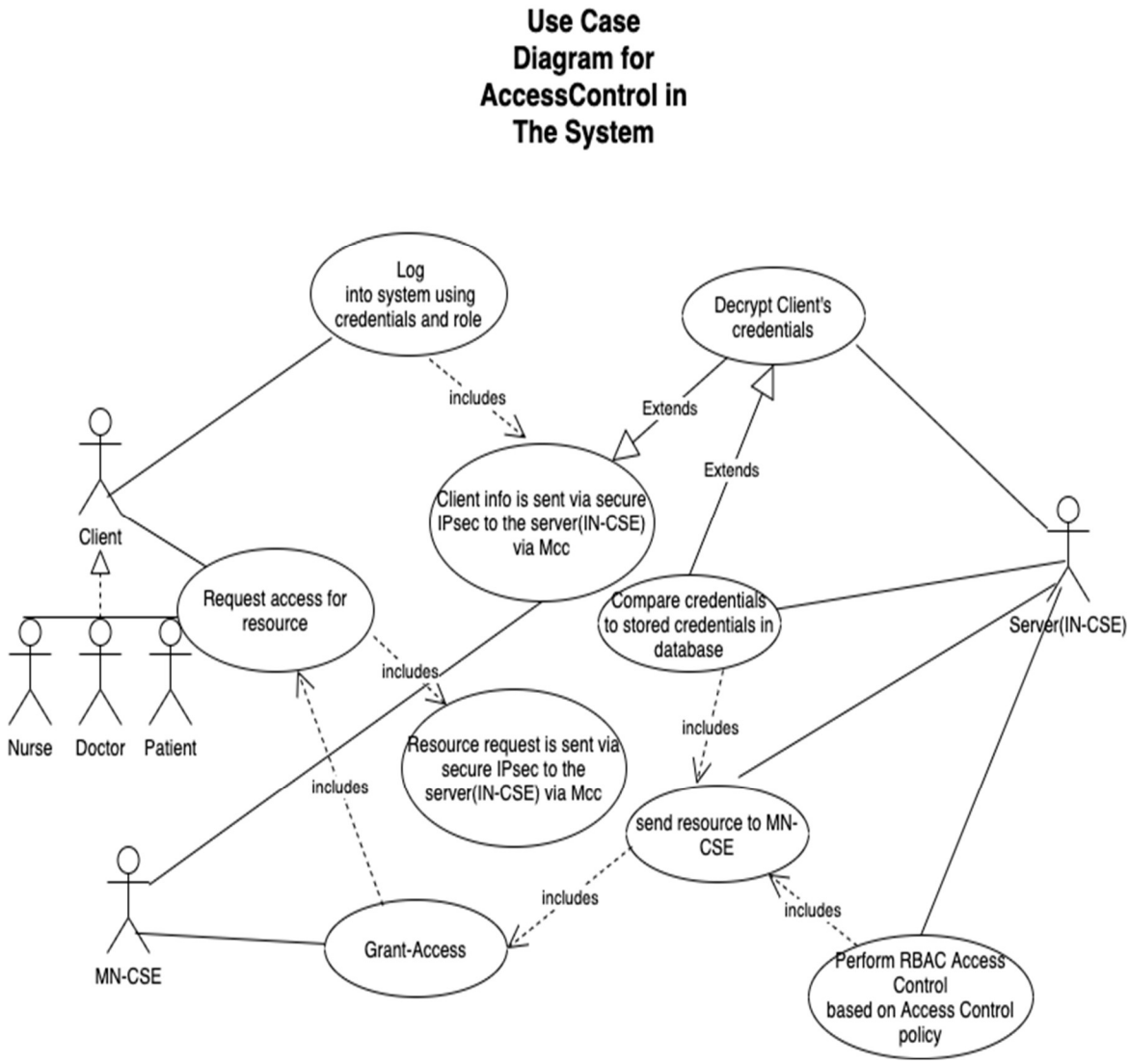

Figure 3: Use case showing Access Control in the System.

4.2 System sequence diagram

The following diagram shows the registration, authentication and access control processes. A patient is on an application entity registering at a clinic via raspberry pi 1. Then the credentials are sent to the IN-CSE server via Mcc. This will then be stored in the database. In another stage a nurse that is located at another hospital and already registered previously, can log in the system via the MN-CSE. The credentials are compared to the one stored in the database. If the encrypted password matches the one that was input, then the nurse is successfully authenticated.

In another sequence, a doctor that is already logged in will try to access a patient's medical information. The resource is located in the IN-CSE in the database. The server then forwards the information via a secure IPsec, to the MN-CSE that the doctor's device is registered at. The following is the sequence diagram described below:

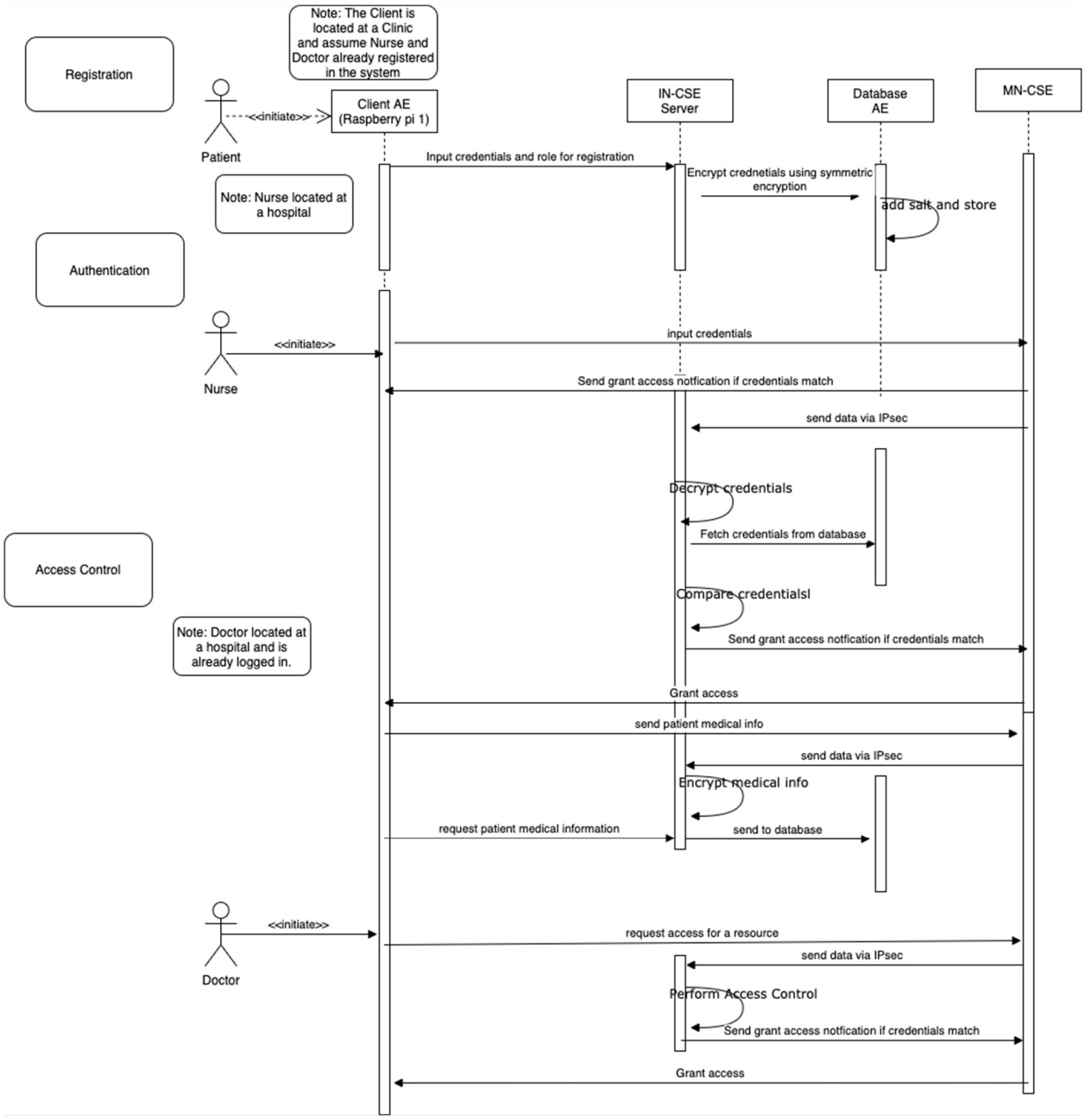

Figure 4: System Sequence Diagram

5. Design Requirements

For Security Threat Modeling of IoT-Enabled E-Health Systems, after researching and designing the use cases for the project, the group came up with the following requirements for the system that must be met to achieve the goal. The requirements are as follows:

5.1 Functional requirements

The functional requirements for the system are as follows:

5.1.1 The security tools shall be generic.

5.1.2 The system shall register devices with different interfaces.

5.1.3 The system shall create a profile for each user.

5.1.4 The system shall register users in different roles.

5.1.5 The system shall have different levels of trust based on their roles.

5.1.6 The system shall have security policies.

5.1.7 The devices shall have open ports for communication.

5.1.8 The system shall be integrated with om2m.

5.1.9 The system shall have a database.

5.2 Non-functional requirements

The non-functional requirements of the system are as follows:

5.2.1. The security tools shall be implemented in the form of a plugin.

5.2.2. The plugin shall be capable of deployment on any device.

5.2.3. The system shall perform encryption and decryption.

5.2.4. The system shall perform authentication.

5.2.5. The system shall use om2m implementation by Eclipse.

5.2.6. The system shall be java 1.8 compatible.

5.2.7. The system shall use maven as a build tool.

5.2.8. The system shall have Tycho installed.

5.2.9. The system shall have a graphical user interface.

5.2.10. The system shall be turned on and off from the command prompt.

5.3 Interface requirements

The interface requirements of the system are as follows:

5.3.1. The interface from the IoT-enabled e-Health devices to the om2m gateway shall be generic.

5.3.2. The interface shall accept data in any format.

5.3.3. The interface shall accept data from multiple IoT-enabled e-Health systems at once.

6. System Architecture

The architecture of the system is as follows:

6.1 Architecture diagram

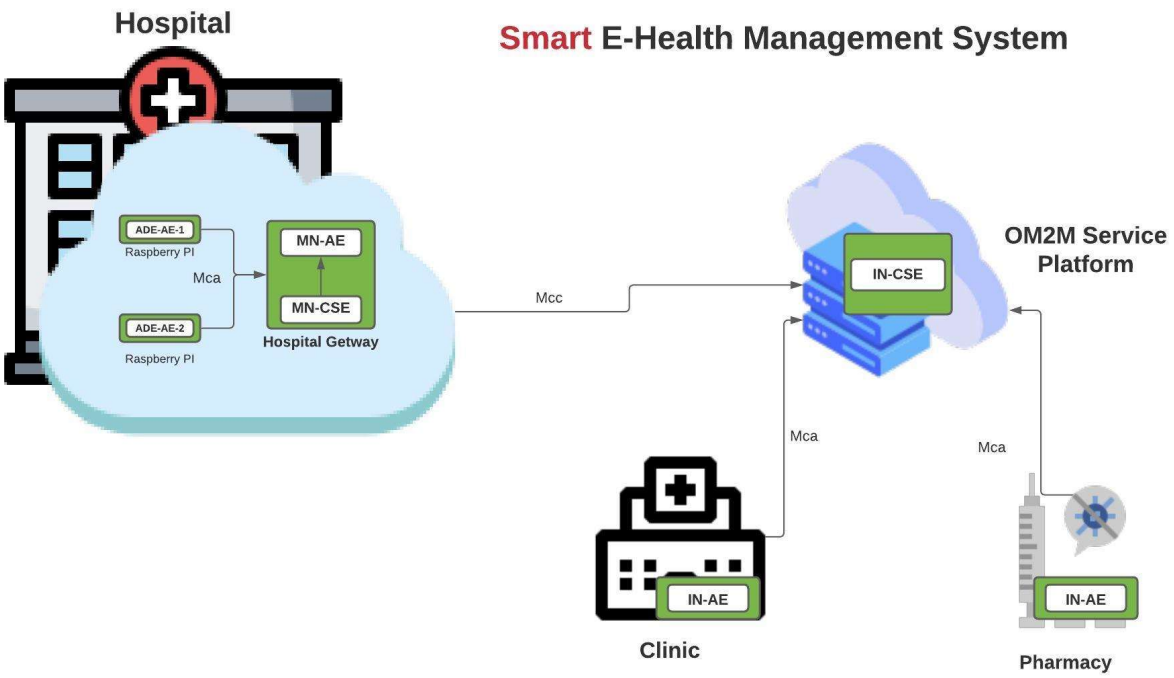

Figure 5: Smart E-health Architecture Diagram

The OM2M project is an open-source implementation of the ETSI M2M. The smart E-health system provides a framework for developing service independently of the underlying network and aims to facilitate deployment of information exchanging system between heterogeneous devices. The basic types of entities for OM2M are Application Entity (AE) and Common service entity (CSE) [27]. The system developed by the team is shown above in the diagram. The description is as follows:

- The hospital, the clinical and the Pharmacy host an application Entity on their own. The AE is placed on an Application Dedicated Node, which is called ADN-AE.
- The Infrastructure Node CSE (IN-CSE) hosts the server, and the middle Node CSE(MN-CSE) hosts the gateway as indicated on the hospital interface.
- The Mca refers to the reference point between an AE and CSE. The Mcc acts as a reference point between the CSEs. In the diagram above, The Mca is used between the ADE-AE 1, ADE-AE-2 and the Hospital Gateway MN-CSE. Also, it is used between Clinic IN-AE, Pharmacy IN-AE and the IN-CSE [27].

In general, the applications and components used are explained as follows:

- ADE-AE-1 and ADE-AE-2 are eclipse plug-ins and have interface that allows users to create, enter and modify information. It interacts with the gateway MN-CSE through Mca reference point and creates a resource.

- IN-AE (Pharmacy and Clinic) applications interact directly with the OM2M server platform

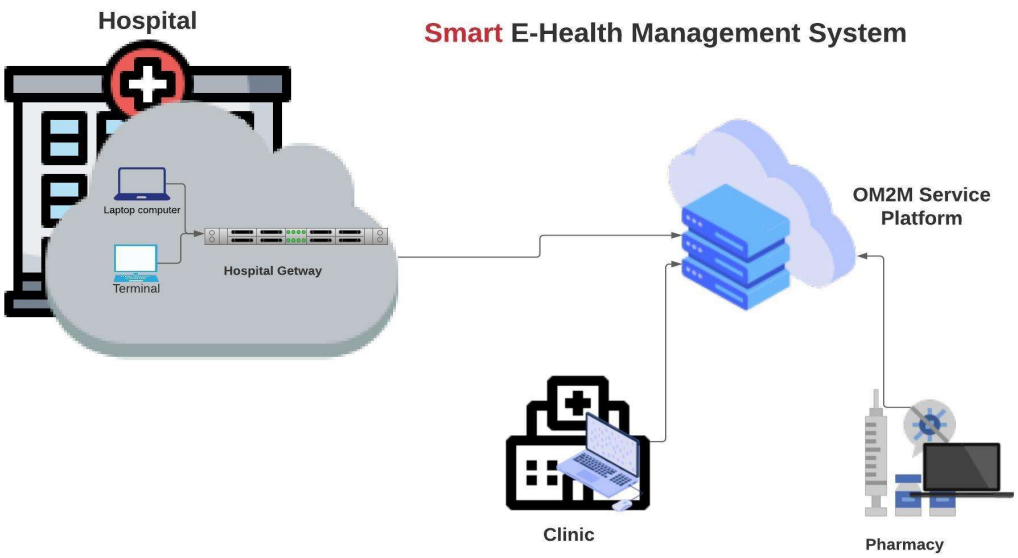

Figure 6: UML diagram for Smart E-Health

(IN-CSE) thought Mca reference point. They are used to get information such as referrals to the hospital or receive prescriptions [26].

6.2 Mapping Subsystems to Hardware

The described use cases enable the E-health management system to incorporate different health care providers under one umbrella management system, which are granted access to the OM2M service platform. The main components of the system underline the overall system of management system [26]. The management system will be deployed in the health care provider system, which inter-connect each system. The main components include:

- The Hospital management systems, which are deployed in any place as needed, such as the doctor's office, front desk. They are connected to the hospital gateways and used to manage patient medical reports, such as patient information, treatment, and prescription.

- The hospital Gateways relate to a cloud or a remote server to provide the necessary communications technology allowing to exchange and store information.
- The OM2M service platform provides a set of services to enable the different platforms to manage the patient such as collecting the data, modify and add treatment.
- The clinic and pharmacy host an application to manage the patient information and data. The clinic uses the system to write a referral, or request any X-RAY, Ultrasound etc. The pharmacy uses it to manage prescriptions from the hospital and clinic.

6.3 Control flow and Network protocol

OM2M Provides a horizontal service platform that facilitates the deployment of vertical applications and leads to innovations in the domain of interoperability.It provides a RESTful Service Capability layer (SCL) accessible via an open interface to enable the development of service and application independently of the underlying network [26]. SCL works as an interface between the network access and the application domain. Each device resource is register to the OM2M standard platform. The SCL can be deployed on an M2M network (IN-CSE), a gateway (MN-CSE) and a Device (IN-AE). OM2M proposes a modular architecture running on top of an OSGI Equinox runtime [25][11]. The core is the main plugin that is deployed in each SCL. It provides a protocol-independent service for handling RESTful Requests. HTTP communication mapping plugin is added to be used as protocol binding with the support of jetty plugin [24]. The OM2M component diagram below explains the following:

- The ***CORE*** plugin implements the SCL_Service interface to handle generic RESTful requests. It receives a protocol-independent response confirmation.

- The **router** defines a single route to handle each request in a resource controller, simply using the request URL and the required method [24].
- The r**esource_controller** implements CRUD methods to create, retrieve, update, delete for each resource. It is responsible for checking the operation such as access right authorization and resource syntax verification [25].
- The ***interworking _proxy*** holds discovered the interworking proxy unit implementing the IPU service.
- The ***Request_sender*** hold discovered protocol-specific clients implementing the client_service interface [24].
- The ***HTTP_Mapping is*** responsible for handling and Mapping the http protocol. It is composed of *HTTP_client* and *HTTP_servlet* request.

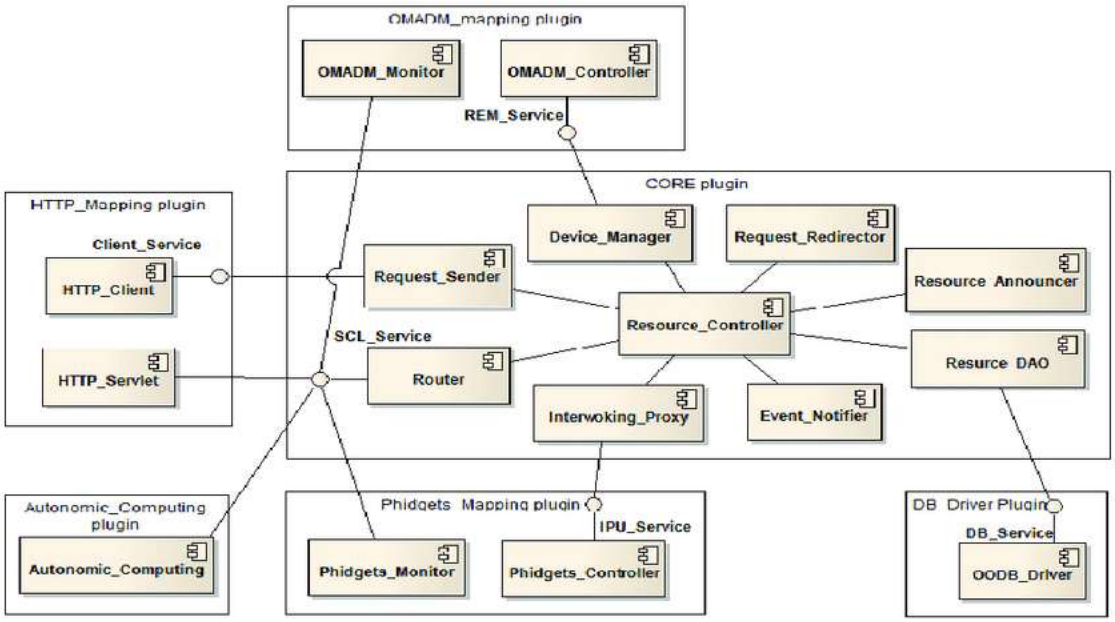

Figure 7: Om2m componet Diagram [25]

As indicated on the resource creation scenario diagram below, the process of creating the application resource implement with an http request with username/password as authorization

header includes the application XML data as the body to the URL [24]. As indicated above, the *http_Mapping* plugin builds a protocol-independent *request indcation* object from the received request and uses it to call the router. It creates an application object form the received XML representation and store it on the database and notifies all interested subscriber via the *even_notifier* and announce the application resource to preselected remote SCLs via the *Resource_Announcer [26]*.

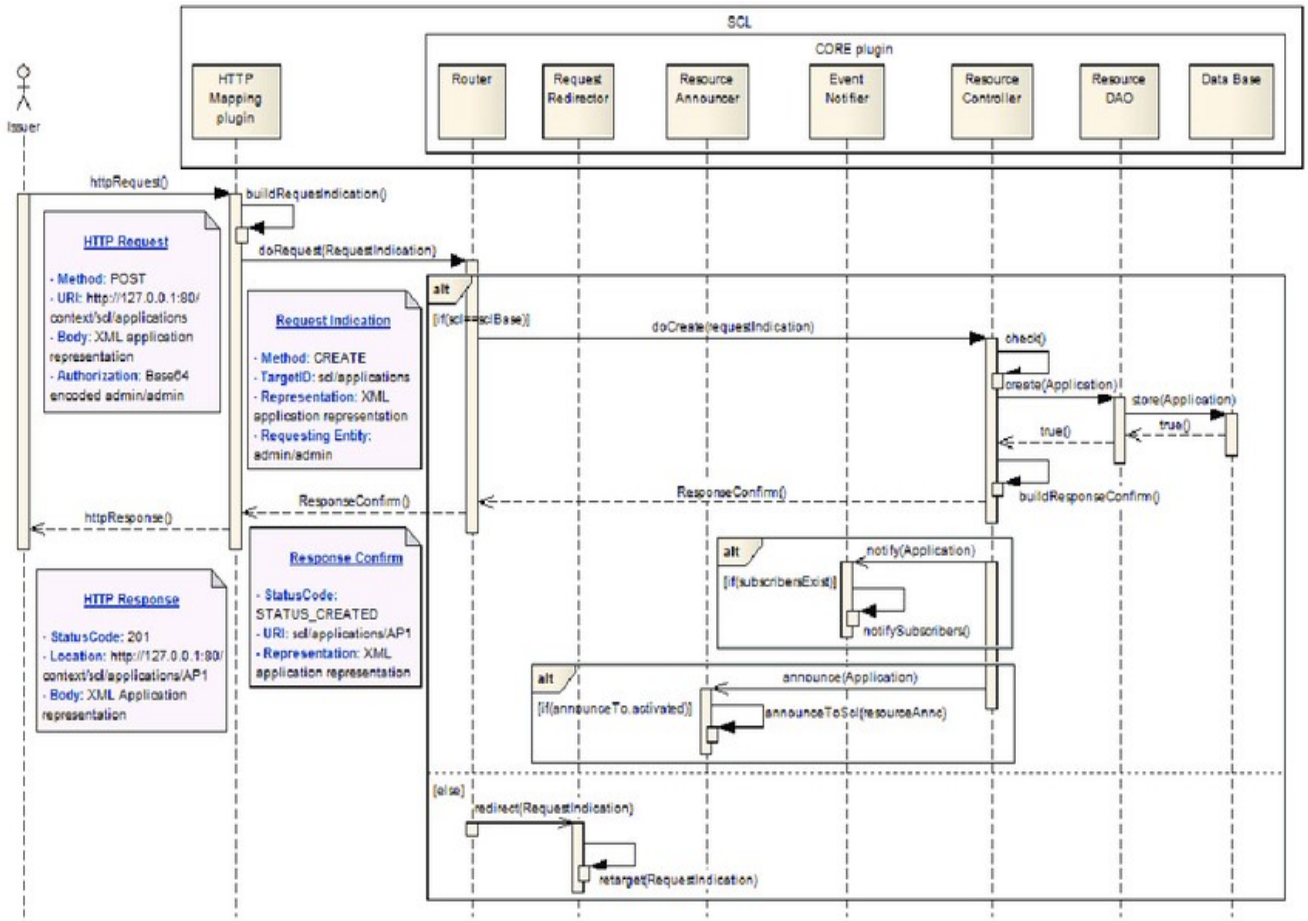

Figure 8: Resource Creation Scenario [24]

The OM2M architecture follows a RESTful style of with a hierarchical namespace, an AJAX web interface based is provided to enable exploring SCLs resource tree, monitoring a patient and medical report. For the project, a smart E-health system is registered and creates a resource tree under the MN-CSE. The MN-CSE is registered and authenticated to the infrastructure node CSE (IN-CSE) [27]. The smart E-health system SCL resource trees have three sub-resource attributes: namely DESCRIPTOR DATA and ACCESSCONTROL. In the data section, each patient has an identification number, and their information is updated or accessed using the patient identification number. Once the Smart E-health system plugin discovers new information is added, it creates SCL that corresponding to the patient id information and data containers.

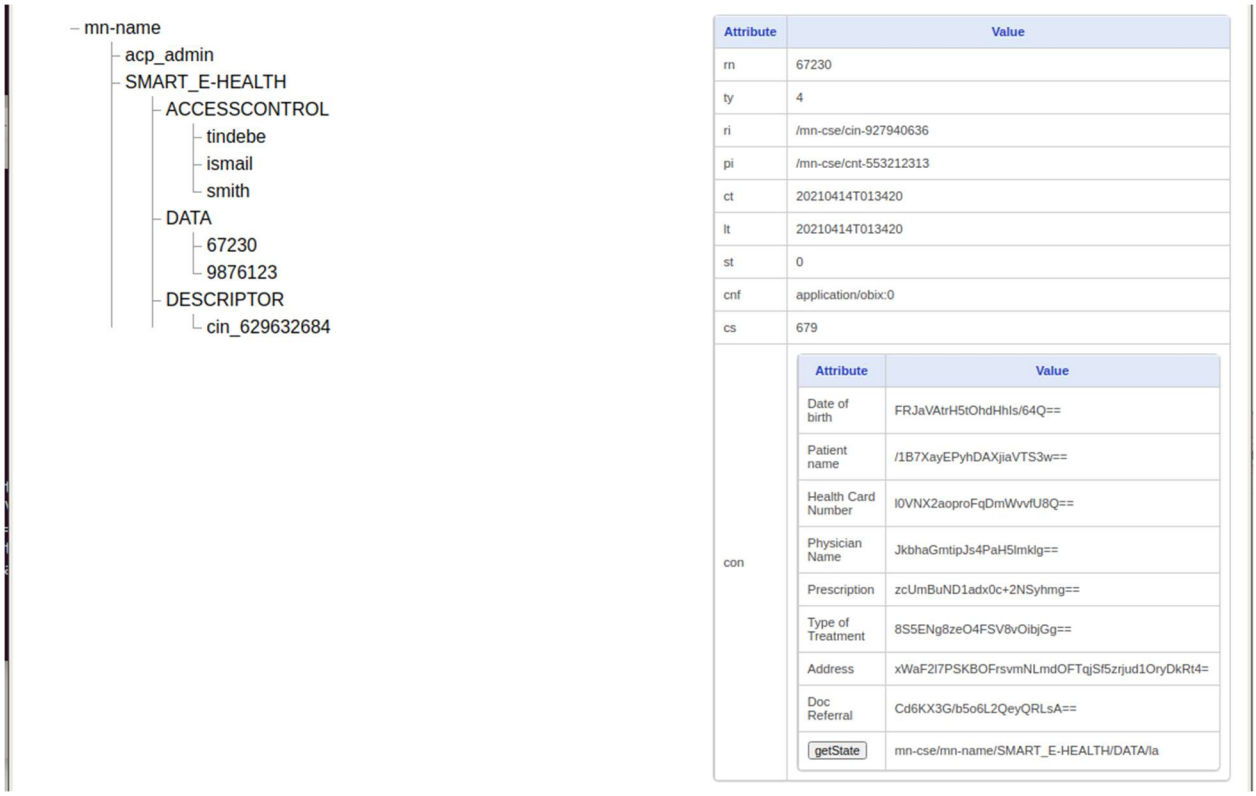

Figure 9: Smart E-health Resource Tree

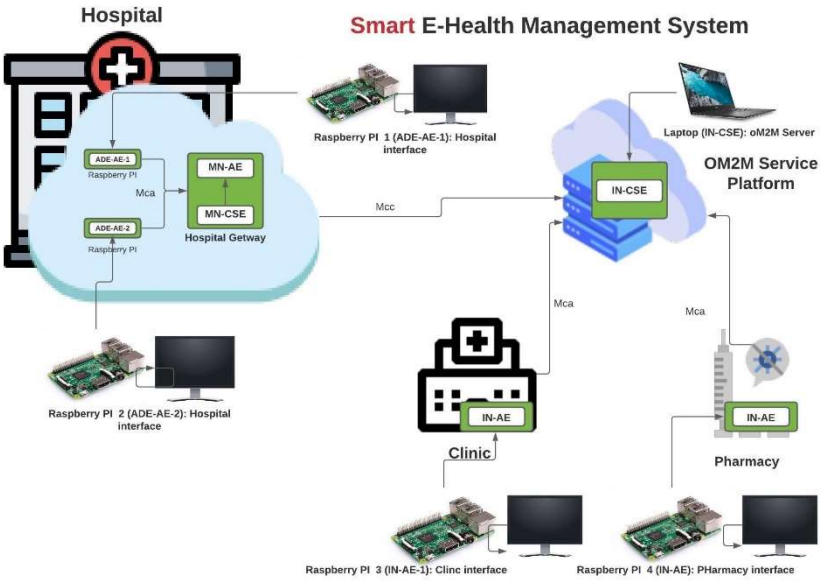

Figure 10: Hardware Requirements and set up

The hardware requirement is specified based on the traditional deployments and driven by the complexity of the om2m service platform [26]. The main goal of the hardware specification is to increase compatibility, run as fast as possible and reliable system.

Hardware Requirements for Deployment:

- The Application entities and MN-CSE are implemented on Raspberry PI 3, which has 4 x ARM Cortex-A53 1.2GHz with a networking 2.4GHZ 802.11n wireless.
- The infrastructure common service entity (server om2m) is implemented on a Laptop, which has a core i7 process.

7. Software Design

7.1 UML with description

The Smart E-heath Plugin takes the CORE OM2M plugins implementation and the modular architecture running on the of an OSGI Equinox runtime. The Core system is the main plugin that deployed in the SCL. The Smart E-health plugin use a RESTful Request as means of communication between each device(gateway), which supposing that communication happened by means of HTTP request. As indicate Figure 11, Each class are responsible for creating the resource tree, required attributes and established communication for sending request back and forth.

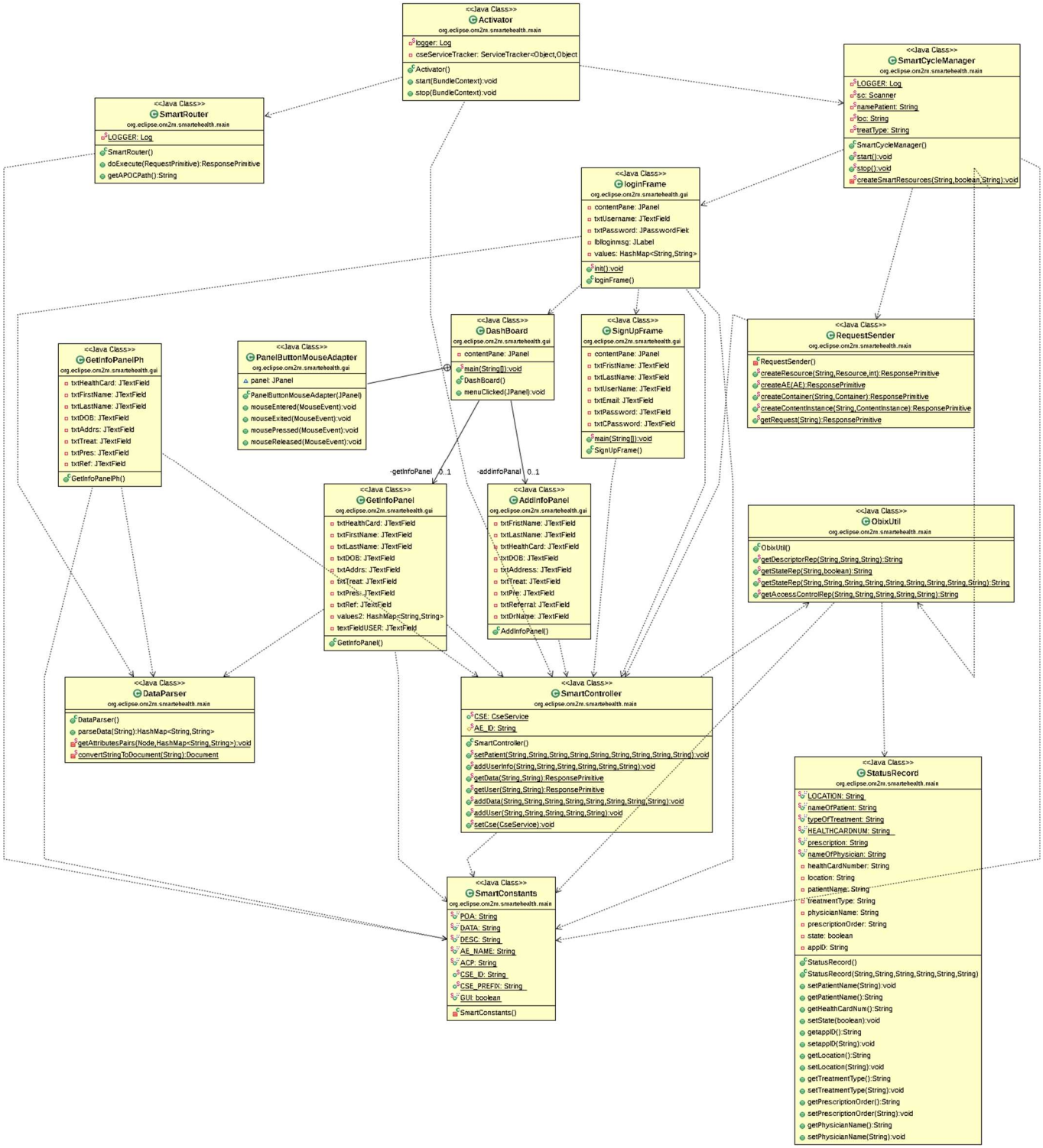

Figure 11: UML with description

7.2 Interface

To create a universal system that allows health care to share, store and exchange information without compromising the security and efficiency of the communication system computer through the use of the Smart E-health Interference. The sign-up and logging screen consider the efficiency and speed of the login process while maintaining simplicity, privacy and data security.

7.2.1 Logging Interface

Smart health system login GUI is a simple and efficient design that allows user to differentiate between logging in and signing up. The design makes a play for the contrast between a colourful. The existing user can easily log into their profile by just using their username (Uniquely identifier a user) and password. However, if it is the user is a new employee, they have an option to sign up and register their credential to use or access the E-Health system. When the user is selects SING UP Butten, it will direct to a new registration form.

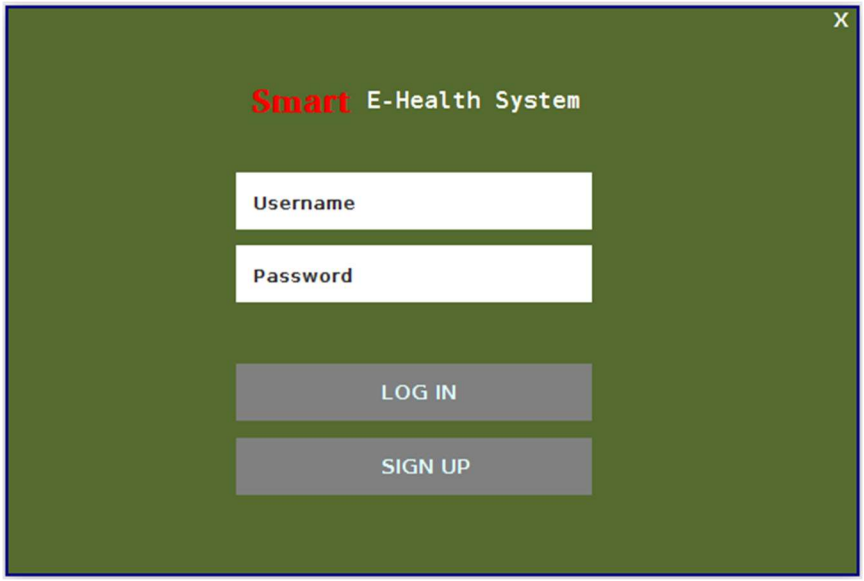

Figure 12: Login interface

7.2.2 Registration interface

To perform any operation or access information for Smart E-Health, users are required to register using their credential. As indicated in figure 13, After the user selected the signup option on the logging, he/she will be directed to the registration form below. The registration form requires some sensitive information. After the user filled the form and select Register Now, the information will be encrypted and store on the server-side as a password file, which will be used for authentication, and access control.

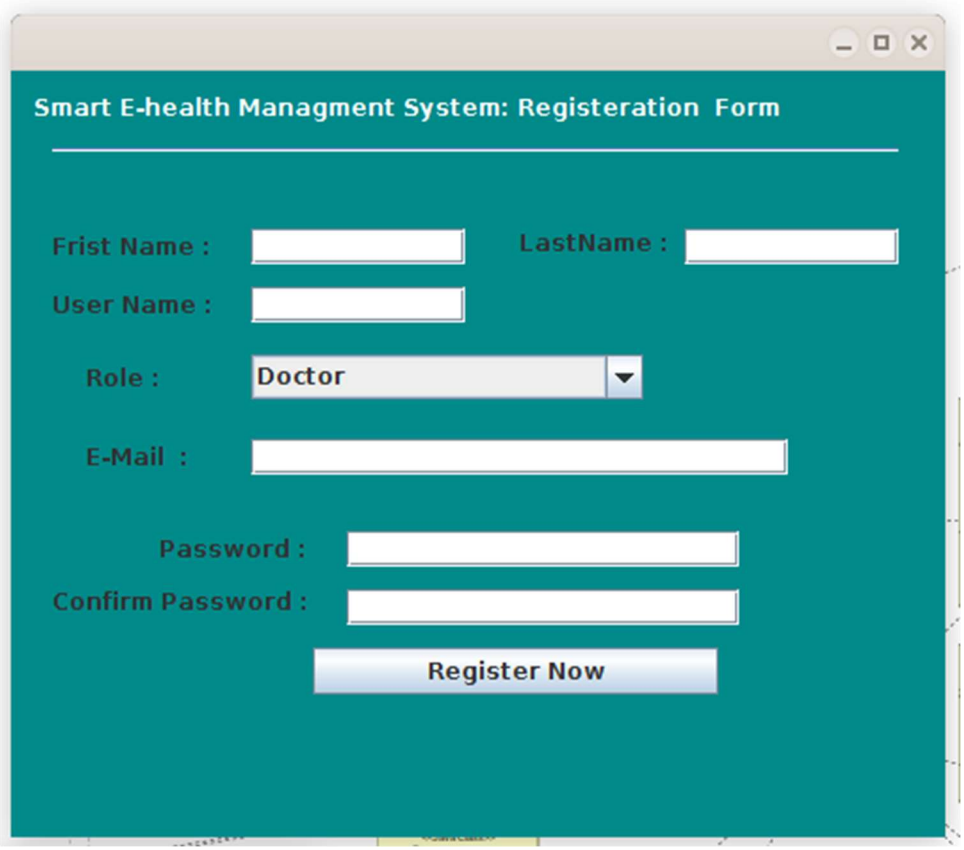

Figure 13: Registration interface

7.2.3 Dashboard interface for patient Health History

With a successful registration and logging, The Patient registration interface is designed to allow the use of concurrently access information while considering simplicity, privacy and data security. It provides an option of adding a new record (ADD) as indicated in Figure 14 or get existing patient information as shown in Figure 15..

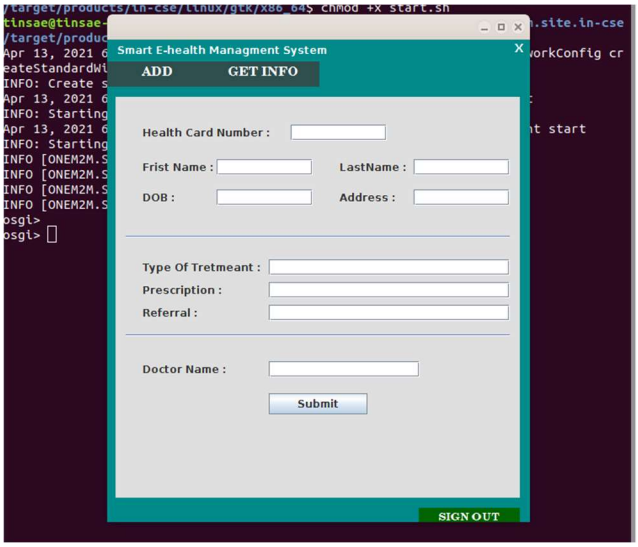

Figure 14: Dashboard interface for patient registration

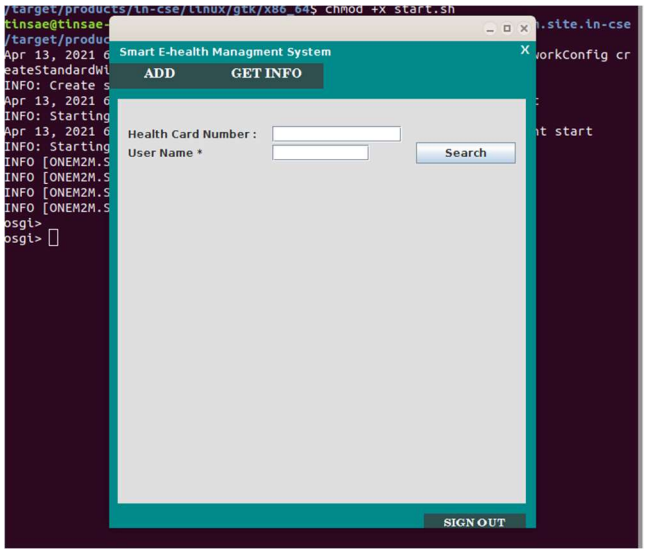

Figure 15: Patient information Retrievable interface

7.3 Integrations of security system with E-health System

The security system of E-health is designed and implemented using the same mechanism as the Smart E-health plugin. The security system is designed as an independent plugin for any modification and reusable for the future. The E-health system takes the security plugin as a dependent, which uses it for encryption, decryption, authentication and access control. Every sensitive information is encrypted before being stored into the infrastructure node (Server) and checked for proper credentials, before decrypting the patient information. The Smart E-Health plugin uses class from the security system plugin such as Authentication.java, AccessControl.java and EncrypandDecrypt.java.

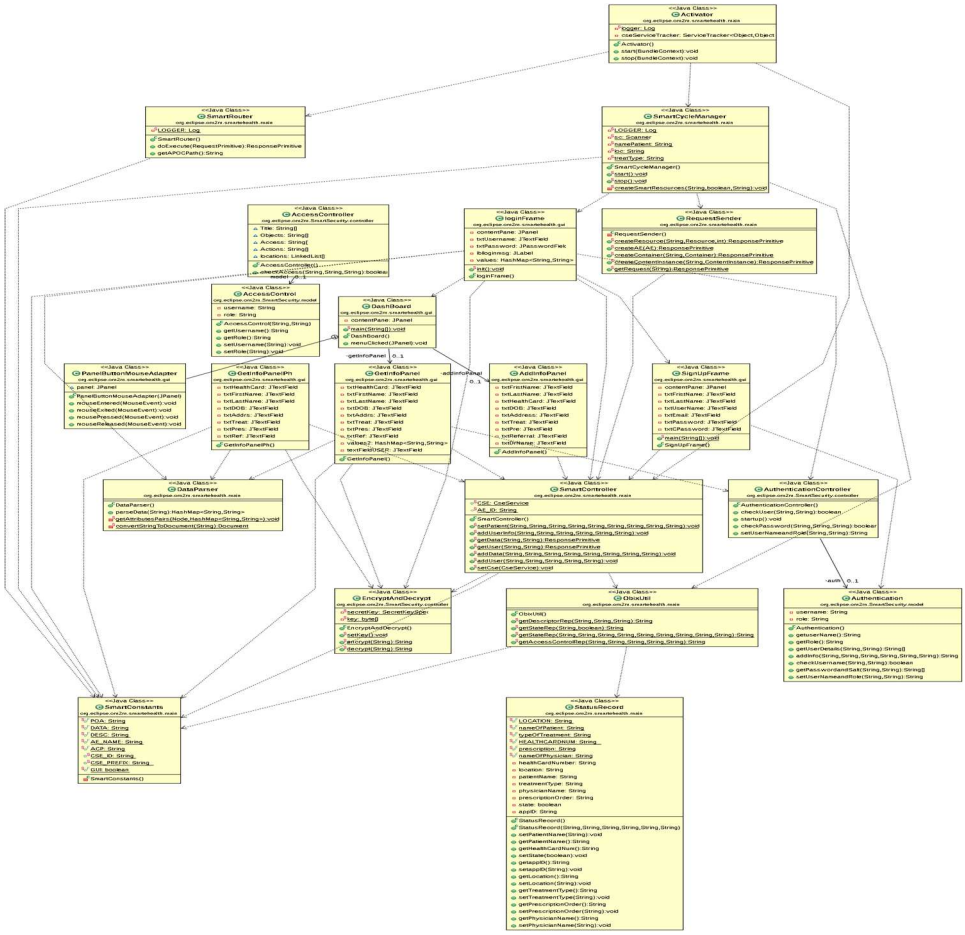

Figure 16: UML diagram of smart e-health security in OM2M

8. Security

8.1 Authentication design

Whenever a user is attempting to access the system, a certain code or program will need to run, check, and verify that the person trying to enter is who they claim they are. The interaction between the server and the user also needs to be as secure as it can be, so that no one else is able to take the information being sent and use it to gain access in the system at another point.

To be able to have a secure access, a password verification is set in place along with an encryption and decryption script when communicating to the server, so only the real user and the system itself know what the password is.

This section will be split into two parts: The first section will explain how the system takes in the password as well as how it stores it in the database and the type of security added that acts as a second line of defense if the system was breached. The second section will discuss the encryption and decryption that was added so that the server can have a secure communication with any user. The section will also discuss the type of encryption used including the type of key, the block size etc.

8.1.1 Authentication implementation

When a user first registers in the system/application, they will need to provide their personal information to the system, like their username, password, health card number and so on. The system will be able to store it in the database for later use when needed.

When the user inputs the password at account registration, a script we wrote will activate and create a secure random number which will be called a salt key, as seen in figure 17 below.

```
SecureRandom rand = new SecureRandom();
byte[] salt = new byte[64];
```

Figure 17: Creating a salt key.

The salt, is a key that will used as a second input to a hash algorithm, to safeguard the passwords.Since hashing two passwords that are the same without any random number added along with them will result in the same exact hashed password, which will not make a system as secure as needed.

The salt key along with the password is used in a SHA-256 hash algorithm to result in a secure hashed password which will then be stored with the salt in the database as seen in the figure 13 below.

```
MessageDigest hashAlg = MessageDigest.getInstance("SHA-256");
rand.nextBytes(salt);
hashAlg.update(salt);
Password = hashAlg.digest(password.getBytes());

Out = Username + ":" + Arrays.toString(Password) + ":" + Arrays.toString(salt)
+ ":" + role+ ":" + contactinfo+ ":" + address+ ":" + healthCardNumber;
```

Figure 18: Hashing the password and the salt.

8.1.2 Encryption and Decryption

In order to secure any communication done between the server and a user, an encryption will be done to each message that leave either end. There are many diverse types of encrypting algorithms and modes alongside key sizes can be used when encrypting information, we will explain which we picked and the reasons behind each choice.

Table 1: Difference between AES and DES [28].

Sr. No.	Key	AES	DES
1	Definition	AES stands for Advanced Encryption Standard.	DES stands for Data Encryption Standard.
2	Key Length	Key length varies from 128 bits, 192 bits to 256 bits.	Key length is of 56 bits.
3	Rounds of Operations	Rounds per key length:128 bits - 10;192 bits - 12;256 bits - 14.	16 rounds of identical operations.
4	Network	AES structure is based on substitution-permutation network.	DES structure is based on feistal network.
5	Security	AES is de-facto world standard and is more secure than DES.	DES is weak and 3DES(Triple DES) is more secure than DES.
6	Rounds	Byte substitution, Shift Row, Mix Column and Key Addition.	Expansion, XOR operation with round key, Substitution and Permutation.
7	Size	AES can encrypt 128 bits of plain text.	DES can encrypt 64 bits of plain text.
8	Derived from	AES derives from Square cipher.	DES derives from Lucifer cipher.
9	Designed By	AES was designed by Vincent Rijmen and Joan Daemen.	DES was designed by IBM.
10	Known attacks	No known attack.	Brute-force, Linear cryptanalysis, and Differential cryptanalysis.

First thing that will be explained is the cipher choice. When picking a symmetric cipher, we will need to see the benefits each provide. There are three types of ciphers: DES, 3DES, and AES. The difference between DES and AES can be seen in table 1 above.

As seen in row 10 of the table, AES is better than DES, since DES uses the old network called feistal structure. This has a few vulnerabilities rather than the new model which is called the substitution-permutation network. 3DES is almost the same as DES, but with a small difference, where the message goes though the cipher three times using three keys. If the keys are different, it adds more protection, but even after that fact AES is still a better choice because it doesn't have any known attacks that can be used against it at this time. For the key size we decided to go with the 64-bit size just for demonstration purposes although a 128-bit size would be recommended, since it would take significantly longer to process.

Second thing we need to look at is the encryption mode, there is two main types of encryption modes, either a block cipher or a stream cipher. For our design, we decided to go with the block cipher since it is less complex to build as well as making it hard to reverse encrypt a message. The block cipher we decided to go with, is the Cipher Block Chaining (CBC), since it provides addition protection against repetitiveness in messages. For example, as seen in the figure 14 in the next page ,Electronic Code Book (ECB) shows patterns while CBC does not.

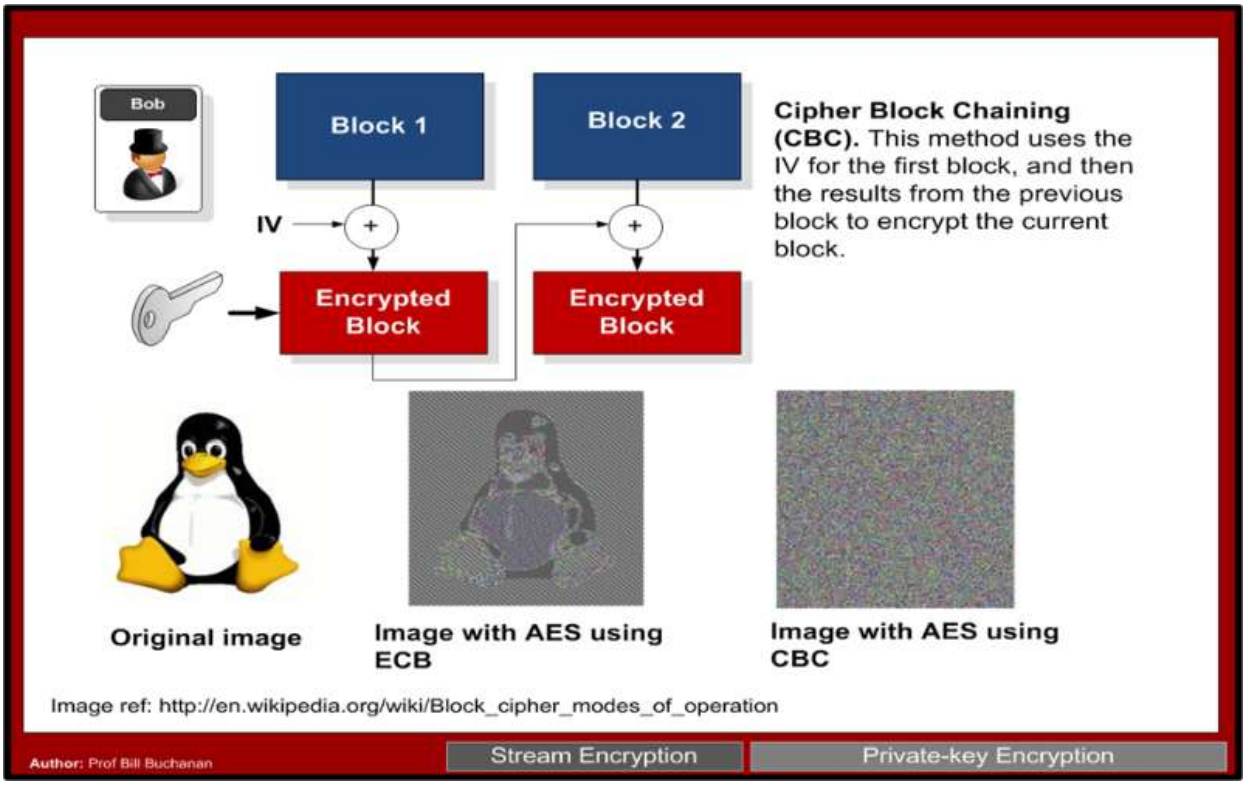

Figure 19: Difference between ECB and CBC [29].

At this point we need to pick a hashing algorithm and the one we decided to go with is SHA-256. Since unlike the latest one, SHA3-256 which is yet to be standardized SHA 2 with a key size of 256 (SHA-256) is the latest version to be fully standardized and offers better protection that the previous versions with bigger key sizes.

Last thing to pick when building the encryption and decryption code is to pick the padding block size, we decided to go with the Password Based Encryption Standard 5 padding (PKCS5padding) which has an 8-byte block size. This is so because, the messages will have a constantly small size rather than use a 1-255 byte block size which is done in the Password Based Encryption Standard 7 padding (PKCS7padding).

8.2 Access Control

Part of the security layer is access control which is designed and implemented based on a role-based access control scheme. Role based access control, is an access control mechanism that supplies access control based on the role of the user or subject. In a trivial system, there is an actor/subject like a doctor and patient, there is resources to access such as patient's information and there are access rights in the case of the project, we use CRUD. CRUD, which is create, retrieve, update and delete. The associated subjects, the resources and access rights they can have, are all listed in a security policy. This policy is a documented agreement to design and implement security, based on the specifications written in it. The security policy is as follows in the next section.

8.2.1 Security Policy

In the security policy the subjects are Doctors, Nurses, Patients, Technical support. The resources are patient personal information, patient medical information and staff information. Access rights are seek/view, get/retrieve, modify/update and delete. The security policy list is as follows:

1) A technical support can get patients account info except password

2) A doctor can access patient personal and medical information.

3) A nurse can access patient's personal information

4) A patient can access both personal information and self-medical results.

The following code shows the lists for the subjects, objects and access rights given via Access. This is done by giving the integer values for each subject and objects, the proper access listed in the above policy. The policy is modular and subjects, resources and access right can all be added on to the underlying data structures.

```
1 package org.eclipse.om2m.SmartSecurity.controller;
2
3 import org.eclipse.om2m.SmartSecurity.model.AccessControl;
4
5
6 public class AccessController {
7     private AccessControl model;
8     //private AccessUI view;
9
10    String [] Title = {"technical support","Patient","Nurse","Doctor"};
11    String [] Objects = {"PatientPersonalInfo","PatientMedicalInfo","StaffInfo"};
12    String [] Access = {"0211","0112","2111"};
13    String [] Actions = {"seek","get","modify"};
14    LinkedList [] locations = new LinkedList[3];
15    public AccessController() {
16
17        //this.view = view;
18
19        for(int i = 0; i < locations.length;i++) {
20            locations[i] = new LinkedList(Objects[i],Title,Access[i]);
21        }
22    }
23
24    public boolean checkAccess (String requester, String obj, String action){
25        for(int i = 0; i < locations.length;i++) {
26            if(locations[i].listname.equalsIgnoreCase(obj)) {
27                if(locations[i].GrantAccess(requester,action)) {
28                System.out.println("ACCESS GRANTED");
29                return true;
30                }
31                else {
32                    System.out.println("ACCESS DENIED");
33                    break;
34                }
35            }
36        }
37        return false;
38    }
39 }
```

Figure 20: Access Controller Code

8.2.2 Access control design

The design of the access control is based on a role-based scheme as indicated above. The role is part of the user's account and it will be the criteria needed to provide access control. There are three primary rules to consider for the RBAC mechanism.

1. Role assignment – an actor/subject can have the right of access only if he has been given a role.
2. Role authorization – the subject's role must be authorized
3. Permission authorization – A subject can be granted permission only if the subject has authorized access right ensuring rule 1 and 2.

The advantages and disadvantages of RBAC were analyzed and compared before implementation of the access control. RBAC provides:

1. Security in a way that reduces administrative and technical support by reducing the need for password changes. When there is a new subject or change in a role. Through RBAC you can add and switch roles easily and quickly via access control lists which will be discussed later.
2. It is implemented in the back end by aligning roles with the organization's structure so users can work efficiently and autonomously.
3. For a health care system that stores and handles sensitive data it is important to meet regulatory requirements for privacy and confidentiality. RBAC improves compliance by giving the programmers the ability to manage the access and usage of data.

The data structure used to implement RBAC is called an access control list. This data structure acts as a form of a token given to a subject that is made up of an object-access right pair. Given a set of subjects, objects, and access rights an access control list is associated to a subject and it maps the subjects access right towards an object or resource. It is originated from an access control matrix which is a 2D array with the subject and object mapped to each other via an access right. Access control lists can be analogous to keeping a list of users that can access a bank account or safe. The other data structure that can be in comparison to an ACL is capability list. The differences, advantage, and disadvantages of these two data structures are listed in the following table.

Table 2: Access control and capability list.

Access Control Lists (ACL)	**Capability Lists**
Table or matrix that tells a subject the access rights it has for an object.	It is a token or key that gives the subject of the key the access right for a resource.
It's like maintaining a list of people that can access a bank account.	It's like giving the owner of the bank account multiple cards or passes.
The bank is involved in adding a new user, verifying user and storing the list.	No need to for the bank in any transactions, to add a new person the owner gives the card.
Bank must safely guard the list. User can't extend privileges. Revoking can be done by the owner by removing friends from the list.	Hard to forge the card. User can delegate his/her privilege. Revoking key is hard because it's hard to identify if the key was copied.
Have to traverse the list looking for access right of a user. Less efficient.	The system knows based on the token about the access right. More efficient

Availability and accountability is better because all the access rights of subjects for objects is easy because it's at one location.	To find who are all the users that have a capability to access a resource we need to check all the tokens.

Based on the above differences the ACL provides better accountability and availability, revocation of resources as well as easier implementation. Although capability lists have better efficiency ACL is chosen by compromising for better security design and ease of implementation.

8.2.3 Access control implementation

The above capability list was implemented using a linked list. The linked list was a user defined linked list and not the one from Java's API. This was done to have a suitable list that can mimic the functionalities of a capability list. The linked list is implemented in the following code: The linked list is chosen because it does not have fixed size and updating it through insertion and deletion is very efficient compared to arrays.

```java
package org.eclipse.om2m.SmartSecurity.controller;

public class LinkedList {
    Node head; // head
    String listname;
    String [] Actions = {"seek","get","modify"};
    // the node class I will need for the solution
    static class Node {

        String name;
        int number;
        Node next;

        Node(String n,int d)
        {
            name = n;
            number = d;
            next = null;
        }
    }

    // Initializes the LinkedList into the sets I need exactly
    public LinkedList(String list,String [] name,String codes) {
        this.listname = list;
        for(int i = 0; i < name.length; i++) {
            int a = Character.getNumericValue(codes.charAt(i));
            if(a != 4) {
                this.insert(name[i], a);
            }
        }
    }

    // inserts a new node into the list
    public void insert(String name, int number) {
        Node new_node = new Node(name,number);
        new_node.next = null;

        if (head == null) {
            head = new_node;
        }
        else {
            Node last = head;
            while (last.next != null) {
                last = last.next;
            }
            last.next = new_node;
        }

    }

    //grants access to the user if the request is with the limit of the access of said title
    public boolean GrantAccess (String name, String access){
        System.out.println("Checking Access to "+ listname +" for a " + name + " wanting to " +
access );
        for(int i = 0; i < Actions.length;i++){
            if(Actions[i].toLowerCase().contentEquals(access.toLowerCase())) {
                int a = this.GetAccess(name);
```

Figure 21: LinkedList Code

The linked list holds the access rights that are mapped to the subject and object pair. The constructor creates linked lists with nodes that have a subject as the head and then the access rights for the objects as the nodes. At the end, a list of access rights of objects for a list of subjects will be created according to the security policy and RBAC design. The insert function inserts the nodes while grant access will grant access for a specific user with a name, checking against the security policy on every iteration.

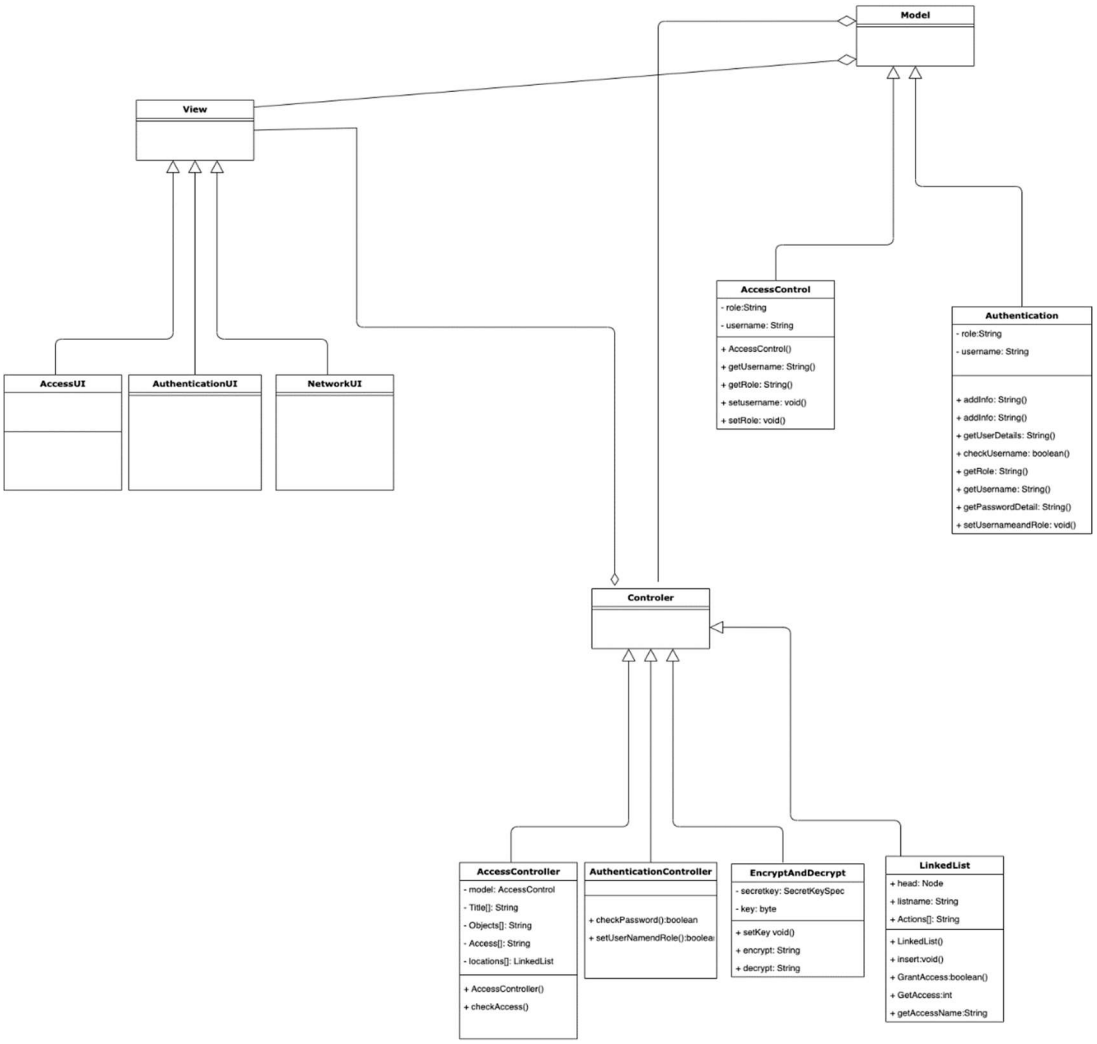

Figure 22: MVC design of security system.

8.3 Database management and sensitive data storage

To store the sensitive data of the hospital, a database was created using H2 database. The database consists of 4 tables. The tables were made in a way that preserved data to avoid data loss, to supply a secure storage using encryption techniques and considering the level of sensitivity of data. The database will hold four tables. The description of each table will be below in detail to supply a brief explanation.

8.3.1 Patient's Personal Information

The first table is associated with the patient's information and it holds the patients name, nickname, email, address, health card details, age, allergies, pre-conditions, current conditions, and family doctor. The table includes anything a doctor would want to know about the patient when diagnosing them, so that there will not be any conflicts in treatments or side effects of existing allergies. The pre-conditions and current conditions will be pointers inside the table, which will be pointing to a row in the patient diagnostic table to save time and space when looking up though each patient.

8.3.2 Patient Medical Information

The second is the patient diagnostic table. it contains the information that represents an instance of time the patient visited the hospital. The table consists of the corresponding doctor, the patient, description/notes of what was diagnosed, the prescription given to the patient, the date of the appointment, and the status of the condition. For example, if a patient enters the hospital to see the doctor, the doctor will examine the patient and either request tests or write a prescription. That information must be recorded for future references in case the patient visits the hospital again. This will prevent medications interfering with each other.

8.3.3 Staff Information

The third table will contain the information about the staff personnel that work in the hospital. The table also contains information for the role of each staff member, so that the RBAC based access control will be able to decide who has the right to access a file, read or change a resource/data. A column was also added in the table which will record which qualifications a staff has in case a certain doctor is needed for an emergency.

8.3.4 Login Credentials

The last table will be the password table, which will be used to determine if the user logging in has the correct credentials. The table has three columns: user ID, salt, and hashed password. The salt will be used in a hashing function to generate a hashed password for increased protection of details in the database. The hashed password will be used to compare with the password inserted, after it goes through the same hashing function to check the similarities of the passwords.

9. Testing and Results

To be able to view how the security works, some websites and software will be needed. It will enable us to test it out and spot the plugins work in progress. To test out the code, we can launch the OM2M website and enter the MN-CSE which will display for us any communication that happens between a user and the system. The MN-case and in-case both have information either retaining to the user or the server itself. When sending and receiving requests the plugin will be encrypting the data before it goes through the network, which gives us the ability to if the requests are encrypted or not in the MN-case. The encryption is as explained in the security part, it is a 64 key AES cipher with a SHA-256 key hashing algorithm which will be used with an 8-bit size padding and a CBC encryption mode. As seen in the figure 22 below, you can see that the information being set is encrypted using the plugin, before it even leaves the system itself, so that it can be as secure as possible.

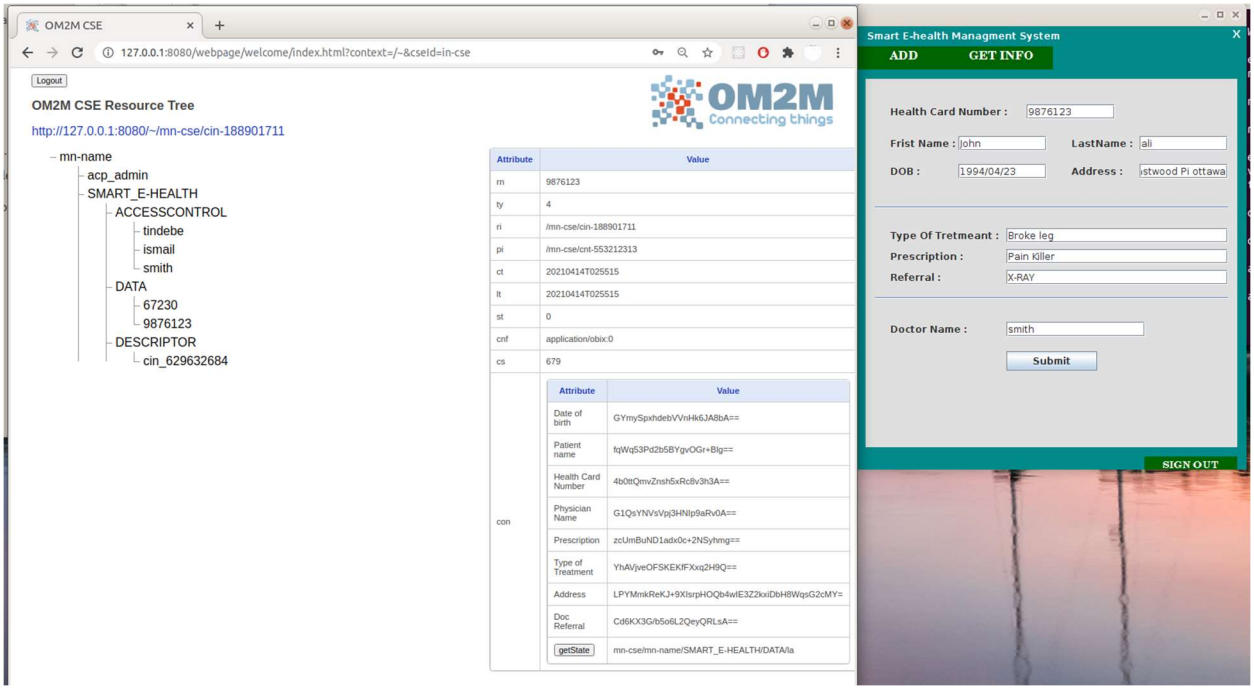

Figure 23: Adding information about a patient.

Another thing figure 22 above show us, is that the GUI and the om2m plugin is functioning correctly, where it does take in all the information, we give it and it sends it in the right manner with the right inputs corresponding to its section. This is important since if there was a problem going on in the system where information is being lost or sent incorrectly it would be hard to diagnose due to how Om2m system is built, so to be able to minimize the field of search would be great when debugging.

After testing the system when adding information now we need to check if it is able to retrieve information from the server as well, in order to do that we will need to use the get info tab in the GUI we built. After filling the form and sending the request to the server, the server will verify the access right for the user requesting and only retrieve what that user is allowed to see in order to maintain confidentiality. As seen in the figure 22 below, the doctor is able to retrieve information about the patient and change it if he sees fit to do so.

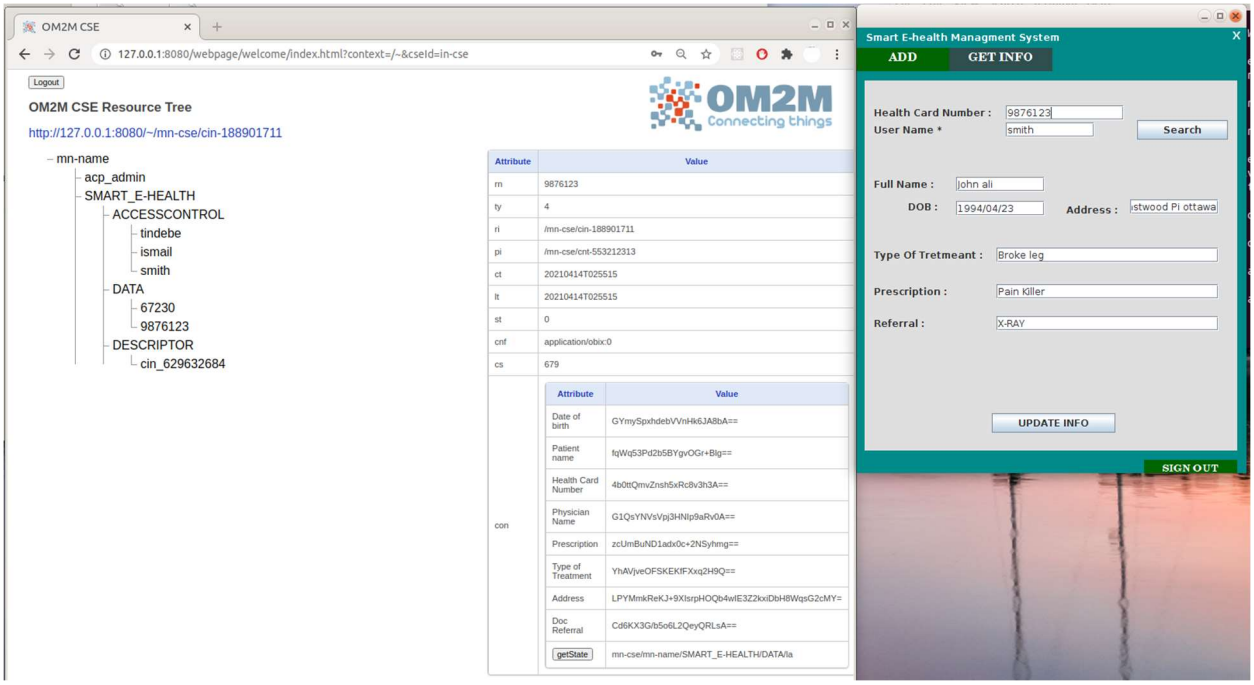

Figure 24: retrieving and editing patient information.

Last two sections to test out will be the registration and login functions to see if they work as intended, first we will look at the registration function. The function itself will need to check if a user with the same name already exists, after doing so it will send all the information that the user added to the server. The password of the user will be hashed alongside a salt value, and the whole thing will be sent to the server as a password, so when a user attempts to login the server will retrieve the salt and hash it with the given password and see if the result will be the same as hashed password saved in the database server. Below is a figure 24 of the registration GUI running.

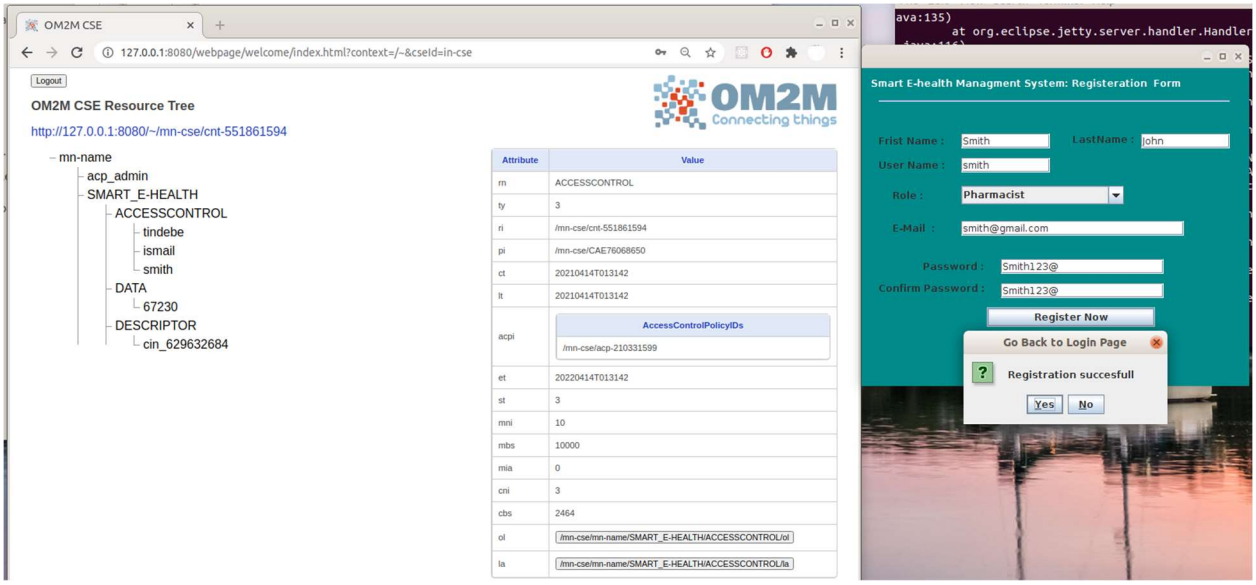

Figure 25: Registration GUI

If you look at the left side of the figure 24 above, you will see a section called SMART_E-HEALTH underneath this section there will be one called ACCESSCONTROL, this section holds all the users whether registered or logged in to the network. As you can see in the right side of figure 24 a person with the family name of smith just registered to the system. Once the

register now button was clicked the information was sent to the server and the user information then appears in the in MN-case, which can be seen in the ACCESSCONTROL list.

Lastly will be logging into the network, when a user attempts to login to the system their password will be hashed and checking with the saved hash password in order to make sure the user is who they claim they are. Once the system verifies the user they will add them to the MN-case just like a newly registered user would be added, when they are added to the MN-case the system will record their privileges as a second type of access control in addition to the one in the script just o add another layer of security. The access control is as explained above in the security section, it is a role-based access control since all the types of interaction that will happen in the server will be role based, for example doctor given prescriptions or pharmacist giving meds to patients. Each person has their set role which will determine what they can or can not do when interacting with the server.

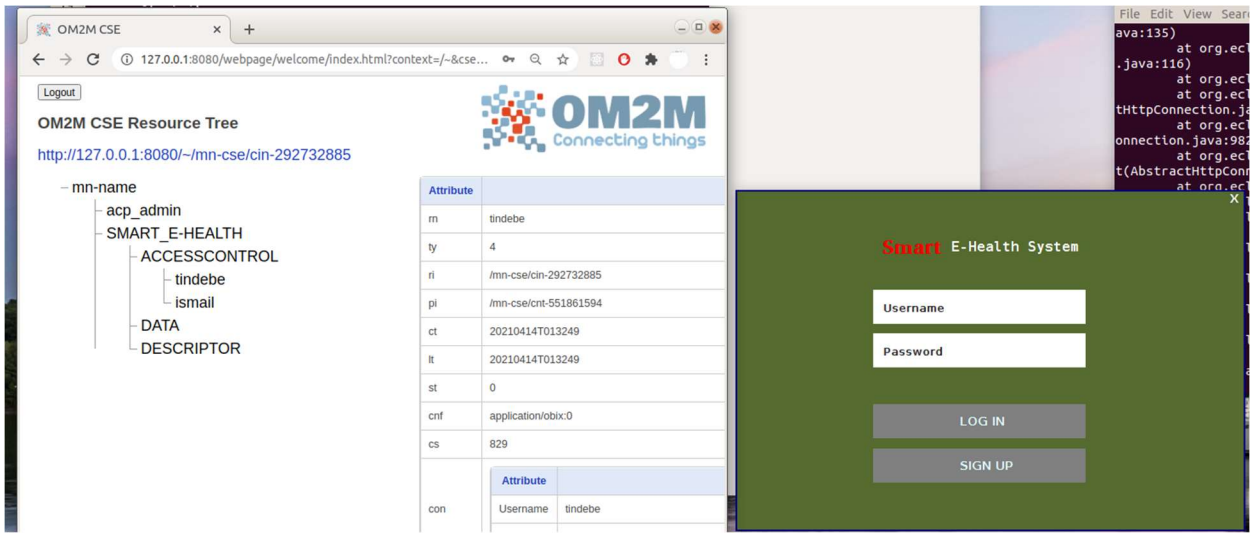

Figure 26: Login GUI

As seen in the figure 25 above, everything is running as expected and the user will be successfully logged in and lead to the dashboard page which can be seen in figure 22 above. All the information related to the user in which the system will need for requests, will be saved underneath the users' family name, and can be seen in the attribute tab on the right side of the website.

After the thorough testing and verifying of the result we got, we can clearly see that the system is functioning exactly as intended and that all the security and communication features work perfectly. As shown above, the protection that the plugin grants the users, and the server is great, and it makes sure that all information is always protected. By using multiple layers of protection as a second line of defense in case someone was able to breach the security. The interface/GUI implemented in the system is quite simple and clear to use, so that anyone in any age bracket can use it without having trouble trying to find something. There is an ease of use and ease of access to the end user.

10. Project management

10.1 Project Network Diagram

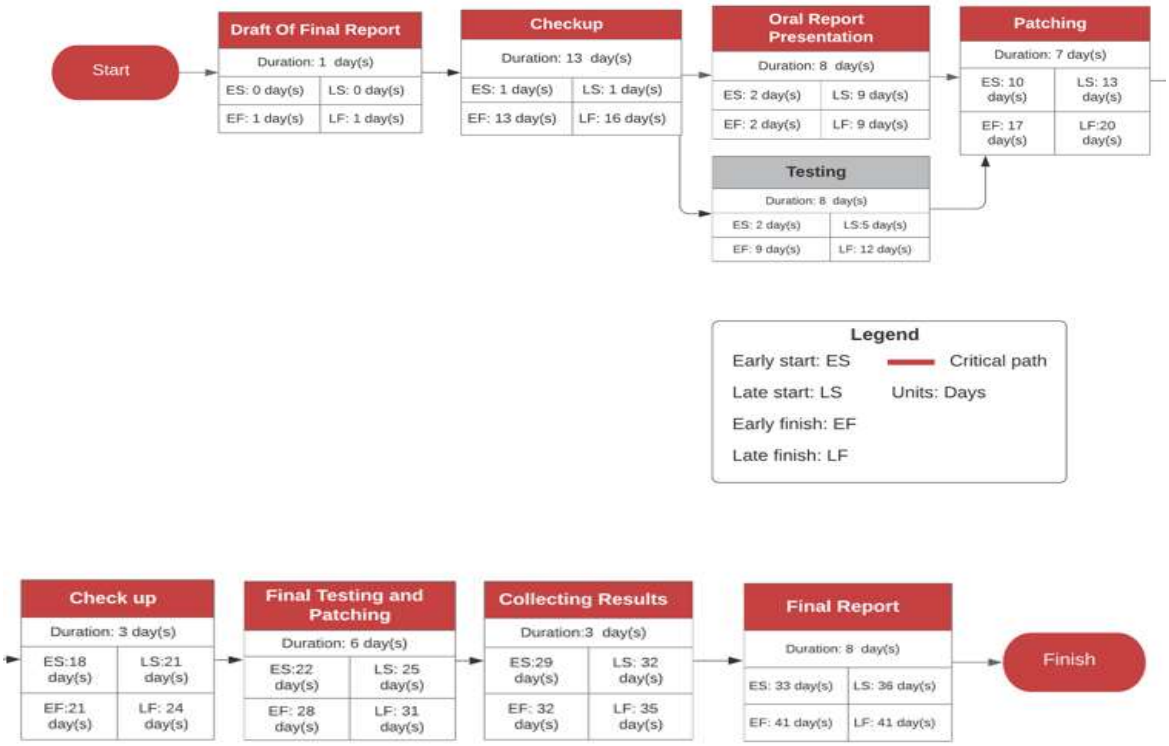

Figure 27: Project Network Diagram

10.2 Schedule/timetable description

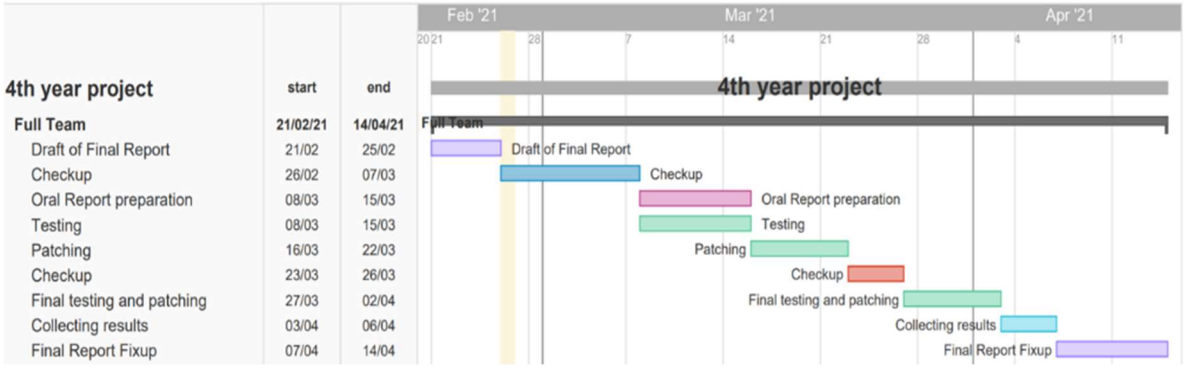

Figure 28: Schedule/timetable description

11. Conclusion

The team has made great progress in carrying out the necessary steps towards the completion and presenting of the project, “Security Threat Modeling for IoT enabled E-Health Systems”. The team had set milestones to achieve the terms and conditions of the proposal and those milestones were achieved successfully on time as per the timetable. The project has proven to be a success, as the team has achieved all the tasks and had learned a lot from it.

The team was able to purchase the necessary Raspberry Pis and accessories within the budget limit to complete the project successfully. The design choice was made to decrease cost, while efficiently building a system that can meet the requirements. In the design phase, the group made a collaborative conclusion to the design of the security layer. When questions or misunderstandings arise, the group was able to listen to each other and come up with a plan that best suited the project needs rather than an individual’s need. This kind of collaboration has been on-going since the start of the project.

In conclusion, the group has brought together their 4 years of knowledge in engineering as well as acquired knowledge from work and projects to be able to progress well in this project. All group members have equally contributed towards the completion of the project. With that being said, the group is delighted to design and implement a security system for an E-Health System that will provide the necessary confidentiality, authentication, and access control. We are hopeful that future teams and students can learn from, use, and improve this project.

References

[1] Sabrina Sicari, Alessandra Rizzardi, Alberto Coen-Porisini, Luigi Alfredo Grieco, Thierry Monteil. Secure OM2M Service Platform. International Workshop on Self-Aware Internet of Things (SELFIOT 2015), Jul 2015, Grenoble, France. 6p. ffhal-01228303

[2] Adebayo Omotosho, Benjamin Ayemlo Haruna & Olayemi Mikail Olaniyi (2019) Threat Modeling of Internet of Things Health Devices, Journal of Applied Security Research, 14:1, 106-121, DOI: 10.1080/19361610.2019.1545278

[3] Omotosho, A., Emuoyibofarhe, J., & Oke, A. O. (2017). Securing private keys in electronic health records using session-based hierarchical key encryption. Journal of Applied Security Research,12(4), 463–477. doi:10.1080/19361610.2017.1354272 Privacy Right Clearinghouse (2016). Data breaches. Retrieved from https://www.privacy-rights.org/data-breaches.

[4] Landi, H. (2018). 2017 Breach report: 477 breaches, 5.6M patient records affected. Retrieved

from https://www.healthcare-informatics.com/news-item/cybersecurity/2017-breach-report-477-breaches-56m-patient-records-affected 120 A. OMOTOSHO ET AL.

[5] Omotosho, A., & Emuoyibofarhe, J. (2014). A criticism of the current security, privacy and accountability issues in electronic health records'.International Journal of Applied Information Systems,7(8), 11–18. doi:10.5120/ijais14-451225

[6] Omotosho, A., Emuoyibofarhe, J., & Meinel, C. (2017a). Ensuring patients'privacy in a cryptographic-based-ehr using bio-cryptography. International Journal of Electronic Healthcare,9(4), 227–254. doi:10.1504/IJEH.2017.085800.

[7] Omotosho, A., Emuoyibofarhe, J., & Meinel, C. (2017b). Securing E-prescription from medical identity theft using steganography and anti-phishing techniques. Journal of Applied Security Research,12(3), 447–461. doi:10.1080/19361610.2017.1315788

[8] Campbell, D. (2016). Internet of things security and privacy –part 1. Retrieved from http://www.macadamian.com/2016/11/15/internet-of-things-security-and-privacy-part-1/

[9] Evans, D. (2011). The internet of things how the next evolution of the internet is changing everything. Retrieved from https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[10] Nordrum, A. (2016). Popular internet of things forecast of 50 billion devices by 2020 is outdated. Retrieved from https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

[11] Abie, H., & Balasingham, I. (2012, February). Risk-based adaptive security for smart IoT in eHealth. In Proceedings of the 7th International Conference on Body Area Networks (pp.269–275). Brussels, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[12] Doukas, C., Maglogiannis, I., Koufi, V., Malamateniou, F., & Vassilacopoulos, G. (2012, November). Enabling data protection through PKI encryption in IoT m-Health devices. In 2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE) (pp. 25–29). Larbaca, Cyprus: IEEE

[13] Agrawal, V. (2015). Security and privacy issues in wireless sensor networks for healthcare. In Internet of Things. User-Centric IoT (pp. 223–228). Cham, Switzerland: Springer

[14] Gonc¸alves, F., Macedo, J., Nicolau, M. J., & Santos, A. (2013, September). Security architecture for mobile e-health applications in medication control. In 2013 21st International Conference on Software, Telecommunications and Computer Networks (SoftCOM) (pp.1–8). Primosten, Croatia: IEEE.

[15] Gope, P., & Hwang, T. (2016). BSN-Care: A secure IoT-based modern healthcare system using body sensor network. IEEE Sensors Journal,16(5), 1368–1376. doi:10.1109/ JSEN.2015.2502401

[16] Islam, S. R., Kwak, D., Kabir, M. H., Hossain, M., & Kwak, K. S. (2015). The internet of things for health care: a comprehensive survey. IEEE Access,3, 678–708.

[17] Joshitta, M. S., & Arockiam, L. (2017). EPC based authentication of devices in the smart health system. International Journal of Electrical, Electronics & Computer Science Engineering,4(4), 6–11. Landi, H. (2018). 2017 Breach report: 477 breaches, 5.6M patient records affected. Retrieved from https://www.healthcare-informatics.com/news-item/cybersecurity/2017-breach-report-477-breaches-56m-patient-records-affected120 A. OMOTOSHO ET AL.

[18] Moosavi, S. R., Gia, T. N., Nigussie, E., Rahmani, A. M., Virtanen, S., Tenhunen, H., & Isoaho, J. (2016). End-to-end security scheme for mobility enabled healthcare Internet of Things. Future Generation Computer Systems,64,108–124. doi:10.1016/j.future.2016.02.020

[19] Sivaraman, V., Gharakheili, H. H., Vishwanath, A., Boreli, R., & Mehani, O. (2015, October). Network-level security and privacy control for smart-home IoT devices. In 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob) (pp. 163–167). IEEE, Abu Dhabi, United Arab Emirates

[20] Savola, R. M., Savolainen, P., Evesti, A., Abie, H., & Sihvonen, M. (2015, August). Risk driven security metrics development for an e-health IoT application. In Information Security for South Africa (ISSA), 2015 (pp. 1–6). Johannesburg, South Africa: IEEE.

[21] Zhang, X. M., & Zhang, N. (2011, May). An open, secure and flexible platform based on internet of things and cloud computing for ambient aiding living and telemedicine. In 2011 International Conference on Computer and Management (CAMAN) (pp. 1–4). IEEE, Wuhan, China.

[22] Jagannathan, V. (2018). Threat modeling: Architecting & designing with security in mind (pp. 1–33). Retrieved from https://www.owasp.org/images/a/a6/AdvancedThreatModeling.pdf.

[23] Shostack, A. (2014). Threat modeling: Designing for security. Indianapolis, IN: John Wiley &Sons.

[24] M. Ben Alaya, Y. Banouar, T. Monteil, C. Chassot, K. Drira, OM2M: Extensible ETSI-compliant M2M Service Platform with Self-configuration Capability, Procedia Computer Science,Volume 32,2014,

[25] Sabrina Sicari, Alessandra Rizzardi, Alberto Coen-Porisini, Luigi Alfredo Grieco, Thierry Monteil. Secure OM2M Service Platform. International Workshop on Self-Aware Internet of Things (SELFIOT 2015), Jul 2015, Grenoble, France. 6p. hal-01228303

[26] Guindon, C. (n.d.). Om2m. Retrieved February 26, 2021, from https://www.eclipse.org/proposals/technology.om2m/

[27] Velez, L. (n.d.). Standards for M2M and the Internet of Things. Retrieved February 26, 2021, from https://www.onem2m.org/tr-0045/architecture

[28] Difference between AES and DES Ciphers. [Online]. Available: https://www.tutorialspoint.com/difference-between-aes-and-des-ciphers#:~:text=AES%20stands%20for%20Advanced%20Encryption,stands%20for%20Data%20Encryption%20Standard.&text=Key%20length%20varies%20from%20128%20bits%2C%20192%20bits%20to%20256%20bits.&text=AES%20is%20de%2Dfacto%20world,is%20more%20secure%20than%20DES. [Accessed: 13-Apr-2021].

[29] P. B. B. OBE, "Electronic Code Book (ECB) and Cipher Block Chaining (CBC)," *Medium*, 11-Jan-2021. [Online]. Available: https://medium.com/asecuritysite-when-bob-met-alice/electronic-code-book-ecb-and-cipher-block-chaining-cbc-e3309d704917. [Accessed: 13-Apr-2021].

Appendix A

Project Code

Add Info Panel

```
 1 package org.eclipse.om2m.smartehealth.gui;
 2
 3 import javax.swing.JPanel;
15
16 public class AddInfoPanel extends JPanel {
17     private JTextField txtFristName;
18     private JTextField txtLastName;
19     private JTextField txtHealthCard;
20     private JTextField txtDOB;
21     private JTextField txtAddress;
22     private JTextField txtTreat;
23     private JTextField txtPre;
24     private JTextField txtReferral;
25     private JTextField txtDrName;
26
27     /**
28      * Create the panel.
29      */
30     public AddInfoPanel() {
31         setBackground(SystemColor.window);
32         setBounds(0, 0, 480,476);
33         setVisible(true);
34         setLayout(null);
35
36         JLabel fristName = new JLabel("Frist Name :");
37         fristName.setBounds(32, 76, 88, 15);
38         add(fristName);
39
40         txtFristName = new JTextField();
41         txtFristName.setBounds(120, 74, 114, 19);
42         add(txtFristName);
43         txtFristName.setColumns(10);
44
45         JLabel lblLastName = new JLabel("LastName :");
46         lblLastName.setBounds(266, 76, 88, 15);
47         add(lblLastName);
48
49         txtLastName = new JTextField();
50         txtLastName.setBounds(354, 74, 114, 19);
51         add(txtLastName);
52         txtLastName.setColumns(10);
53
54         JLabel lblHcard = new JLabel("Health Card Number :");
55         lblHcard.setBounds(32, 35, 158, 15);
56         add(lblHcard);
57
58         txtHealthCard = new JTextField();
59         txtHealthCard.setBounds(208, 33, 114, 19);
60         add(txtHealthCard);
61         txtHealthCard.setColumns(10);
62
63         JLabel lblNewLabel = new JLabel("DOB :");
64         lblNewLabel.setBounds(32, 112, 70, 15);
65         add(lblNewLabel);
66
67         txtDOB = new JTextField();
68         txtDOB.setBounds(120, 110, 114, 19);
```

```
69          add(txtDOB);
70          txtDOB.setColumns(10);
71
72          JSeparator separator = new JSeparator();
73          separator.setBounds(12, 165, 456, 15);
74          add(separator);
75
76          JLabel lblNewLabel_1 = new JLabel("Address :");
77          lblNewLabel_1.setBounds(266, 112, 70, 15);
78          add(lblNewLabel_1);
79
80          txtAddress = new JTextField();
81          txtAddress.setBounds(354, 110, 114, 19);
82          add(txtAddress);
83          txtAddress.setColumns(10);
84
85          JLabel lblNewLabel_2 = new JLabel("Type Of Tretmeant :");
86          lblNewLabel_2.setBounds(32, 195, 145, 15);
87          add(lblNewLabel_2);
88
89          txtTreat = new JTextField();
90          txtTreat.setBounds(182, 193, 286, 19);
91          add(txtTreat);
92          txtTreat.setColumns(10);
93
94          JLabel lblPre = new JLabel("Prescription :");
95          lblPre.setBounds(32, 222, 132, 15);
96          add(lblPre);
97
98          txtPre = new JTextField();
99          txtPre.setBounds(182, 220, 286, 19);
100         add(txtPre);
101         txtPre.setColumns(10);
102
103         JLabel lblReferral = new JLabel("Referral :");
104         lblReferral.setBounds(32, 249, 80, 15);
105         add(lblReferral);
106
107         txtReferral = new JTextField();
108         txtReferral.setBounds(182, 247, 286, 19);
109         add(txtReferral);
110         txtReferral.setColumns(10);
111
112         JSeparator separator_1 = new JSeparator();
113         separator_1.setBounds(12, 282, 456, 2);
114         add(separator_1);
115
116         JLabel lblDrName = new JLabel("Doctor Name :");
117         lblDrName.setBounds(32, 316, 122, 15);
118         add(lblDrName);
119
120         txtDrName = new JTextField();
121         txtDrName.setBounds(182, 314, 179, 19);
122         add(txtDrName);
123         txtDrName.setColumns(10);
124
125         JButton btnNewButton = new JButton("Submit");
```

```
126         btnNewButton.addActionListener(new ActionListener() {
127             public void actionPerformed(ActionEvent e) {
128                 String fristName = txtFristName.getText();
129                 String lastName = txtLastName.getText();
130                 String namePatient = fristName + " " + lastName;
131                 String hcard = txtHealthCard.getText();
132                 String preOrder = txtPre.getText();
133                 String birthOfDate = txtDOB.getText();
134                 String phyName = txtDrName.getText();
135                 String treatType =txtTreat.getText();
136                 String address = txtAddress.getText();
137                 String docRef = txtReferral.getText();
138                 SmartController.addData(hcard,namePatient,birthOfDate, treatType, preOrder,
    phyName,address, docRef );
139
140             }
141
142         });
143
144         btnNewButton.setBounds(182, 352, 117, 25);
145         add(btnNewButton);
146
147     }
148 }
149
```

Dashboard

```
1 package org.eclipse.om2m.smartehealth.gui;
2
3 import java.awt.BorderLayout;
20
21 public class DashBoard extends JFrame {
22
23     private JPanel contentPane;
24     private AddInfoPanel addinfoPanal;
25     private editInfoPanel editinfoPan;
26     private GetInfoPanel getInfoPanel;
27     /**
28      * Launch the application.
29      */
30     public static void main(String[] args) {
31         EventQueue.invokeLater(new Runnable() {
32             public void run() {
33                 try {
34                     DashBoard frame = new DashBoard();
35                     frame.setVisible(true);
36                 } catch (Exception e) {
37                     e.printStackTrace();
38                 }
39             }
40         });
41     }
42
43     /**
44      * Create the frame.
45      */
46     public DashBoard() {
47         setBackground(new Color(0, 128, 128));
48         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
49         setBounds(100, 100, 502, 603);
50         //setUndecorated(true);
51         contentPane = new JPanel();
52         contentPane.setBackground(new Color(0, 139, 139));
53         //contentPane.setBorder(new LineBorder(new Color(0,0, 128) , 2));
54         setContentPane(contentPane);
55         contentPane.setLayout(null);
56         addinfoPanal = new AddInfoPanel();
57
58         editinfoPan = new editInfoPanel();
59
60         getInfoPanel = new GetInfoPanel();
61
62
63
64         JPanel getPnl = new JPanel();
65         getPnl.setBackground(new Color(0, 100, 0));
66         getPnl.addMouseListener(new PanelButtonMouseAdapter(getPnl){
67             @Override
68             public void mouseClicked(MouseEvent e) {
69                 menuClicked(getInfoPanel);
70             }
71         });
72         getPnl.setBounds(119, 26, 133, 30);
73         //getPnl.setBorder(new LineBorder(new Color(0,0, 128) , 2));
```

```
74          contentPane.add(getPnl);
75
76          JLabel lblNewLabel_1_1 = new JLabel("GET INFO");
77          lblNewLabel_1_1.setForeground(Color.WHITE);
78          lblNewLabel_1_1.setFont(new Font("Century Schoolbook L", Font.BOLD, 15));
79          getPnl.add(lblNewLabel_1_1);
80
81          JPanel addPnl = new JPanel();
82          addPnl.addMouseListener(new PanelButtonMouseAdapter(addPnl) {
83              @Override
84              public void mouseClicked(MouseEvent e) {
85                  menuClicked(addinfoPanal);
86              }
87          });
88          addPnl.setBackground(new Color(0, 100, 0));
89          //addPnl.setBorder(new LineBorder(new Color(0,0, 128) , 2));
90          addPnl.setForeground(new Color(0, 0, 0));
91          addPnl.setBounds(0, 26, 120, 30);
92          contentPane.add(addPnl);
93
94          JLabel lblNewLabel_1 = new JLabel("ADD");
95          lblNewLabel_1.setForeground(Color.WHITE);
96          lblNewLabel_1.setFont(new Font("Century Schoolbook L", Font.BOLD, 15));
97          addPnl.add(lblNewLabel_1);
98
99          JPanel edietPnl = new JPanel();
100         edietPnl.addMouseListener(new PanelButtonMouseAdapter(edietPnl){
101             @Override
102             public void mouseClicked(MouseEvent e) {
103                 menuClicked(editinfoPan);
104             }
105         });
106         edietPnl.setBackground(new Color(0, 100, 0));
107         edietPnl.setBounds(250, 26, 120, 30);
108         //panel_1.setBorder(new LineBorder(new Color(0,0, 128) , 2));
109         contentPane.add(edietPnl);
110
111         JLabel lblNewLabel_1_2 = new JLabel("EDIT");
112         lblNewLabel_1_2.setForeground(Color.WHITE);
113         lblNewLabel_1_2.setFont(new Font("Century Schoolbook L", Font.BOLD, 15));
114         edietPnl.add(lblNewLabel_1_2);
115
116         JLabel lblExit = new JLabel("X");
117         lblExit.addMouseListener(new MouseAdapter() {
118             @Override
119             public void mouseClicked(MouseEvent e) {
120                 if(JOptionPane.showConfirmDialog(null, "Are you sure you want to close this
    applicaiton", "Confirmation", JOptionPane.YES_NO_OPTION) == 0)
121                 {
122                     DashBoard.this.dispose();
123                 }
124             }
125             @Override
126             public void mouseEntered(MouseEvent e) {
127                 lblExit.setForeground(Color.RED);
128             }
129             public void mouseExited(MouseEvent e) {
```

```
130                 lblExit.setForeground(Color.WHITE);
131             }
132         });
133         lblExit.setHorizontalAlignment(SwingConstants.CENTER);
134         lblExit.setForeground(Color.WHITE);
135         lblExit.setFont(new Font("Dialog", Font.BOLD, 14));
136         lblExit.setBounds(476, 0, 26, 20);
137         contentPane.add(lblExit);
138
139         JLabel lblNewLabel = new JLabel("Smart E-health Managment System");
140         lblNewLabel.setForeground(Color.WHITE);
141         lblNewLabel.setBounds(12, 5, 411, 15);
142         contentPane.add(lblNewLabel);
143
144         JPanel paneMainContent = new JPanel();
145         paneMainContent.setBounds(10, 68, 480, 476);
146         contentPane.add(paneMainContent);
147
148
149
150         JPanel signOutbtn = new JPanel();
151         signOutbtn.addMouseListener(new PanelButtonMouseAdapter(signOutbtn){
152             @Override
153             public void mouseClicked(MouseEvent e) {
154                 if(JOptionPane.showConfirmDialog(null,"Are you sure you want to sign out ?")
    == 0) {
155                     loginFrame frameLogin = new loginFrame();
156                     frameLogin.setVisible(true);
157                     DashBoard.this.dispose();
158                 }
159             }
160         });
161         signOutbtn.setBackground(new Color(0, 100, 0));
162         signOutbtn.setBounds(370, 556, 120, 20);
163         contentPane.add(signOutbtn);
164
165         JLabel lblNewLabel_1_2_1 = new JLabel("SIGN OUT");
166         lblNewLabel_1_2_1.setForeground(Color.WHITE);
167         lblNewLabel_1_2_1.setFont(new Font("Century Schoolbook L", Font.BOLD, 13));
168         signOutbtn.add(lblNewLabel_1_2_1);
169         paneMainContent.setLayout(null);
170
171         paneMainContent.add(addinfoPanal);
172         addinfoPanal.setVisible(true);
173         paneMainContent.add(editinfoPan);
174         paneMainContent.add(getInfoPanel);
175         menuClicked(addinfoPanal);
176         menuClicked(addinfoPanal);
177     }
178     public void menuClicked(JPanel panel) {
179         addinfoPanal.setVisible(false);
180         editinfoPan.setVisible(false);
181         getInfoPanel.setVisible(false);
182
183         panel.setVisible(true);
184     }
185
```

```
186     private class PanelButtonMouseAdapter extends MouseAdapter{
187         JPanel panel;
188         public PanelButtonMouseAdapter(JPanel panel) {
189             this.panel = panel;
190         }
191         @Override
192         public void mouseEntered(MouseEvent e) {
193             panel.setBackground(new Color(112, 128 , 144));
194         }
195         @Override
196         public void mouseExited(MouseEvent e) {
197             panel.setBackground(new Color(47, 79, 79));
198         }
199         @Override
200         public void mousePressed(MouseEvent e) {
201             panel.setBackground(new Color(60, 179, 113));
202         }
203         @Override
204         public void mouseReleased(MouseEvent e) {
205             panel.setBackground(new Color(112, 128, 144));
206         }
207     }
208 }
209
```

Edit Info Panel

```
 1 package org.eclipse.om2m.smartehealth.gui;
 2
 3 import java.awt.SystemColor;
 6
 7 public class editInfoPanel extends JPanel {
 8
 9     /**
10      * Create the panel.
11      */
12     public editInfoPanel() {
13         setBackground(SystemColor.window);
14         setBounds(0, 0, 480,476);
15         setVisible(true);
16         setLayout(null);
17     }
18
19 }
20
```

Get Info Panel

```
1 package org.eclipse.om2m.smartehealth.gui;
2
3 import java.awt.SystemColor;
21
22 public class GetInfoPanel extends JPanel {
23     private JTextField txtHealthCard;
24     private JTextField txtFirstName;
25     private JTextField txtLastName;
26     private JTextField txtDOB;
27     private JTextField txtAddrs;
28     private JTextField txtTreat;
29     private JTextField txtPres;
30     private JTextField txtRef;
31
32     /**
33      * Create the panel.
34      */
35     public GetInfoPanel() {
36         setBackground(SystemColor.window);
37         setBounds(0, 0, 480,476);
38         setVisible(true);
39         setLayout(null);
40
41         JLabel lblNewLabel = new JLabel("Health Card Number :");
42         lblNewLabel.setBounds(44, 34, 152, 15);
43         add(lblNewLabel);
44
45         txtHealthCard = new JTextField();
46         txtHealthCard.setBounds(214, 32, 152, 19);
47         add(txtHealthCard);
48         txtHealthCard.setColumns(10);
49
50         JButton btnSeach = new JButton("Search");
51
52         btnSeach.addActionListener(new ActionListener() {
53             public void actionPerformed(ActionEvent e) {
54                 String HealthCard = txtHealthCard.getText();
55                 String xmlString =
   SmartController.getData(SmartConstants.AE_NAME,HealthCard).getContent().toString();
56                 DataParser data = new DataParser();
57
58                 HashMap<String, String> values = data.parseData(xmlString);
59
60                 txtFirstName = new JTextField();
61                 txtFirstName.setColumns(10);
62                 txtFirstName.setBounds(118, 124, 114, 19);
63                 add(txtFirstName);
64                 txtFirstName.setText(EncryptAndDecrypt.decrypt(values.get("Patient name")));
65
66                 JLabel fristName = new JLabel("Frist Name :");
67                 fristName.setBounds(12, 126, 88, 15);
68                 add(fristName);
69
70                 JLabel lblLastName = new JLabel("LastName :");
71                 lblLastName.setBounds(250, 126, 88, 15);
72                 add(lblLastName);
73
```

```
74              txtLastName = new JTextField();
75              txtLastName.setColumns(10);
76              txtLastName.setBounds(342, 124, 114, 19);
77              add(txtLastName);
78
79
80              JLabel lblNewLabel_1 = new JLabel("DOB :");
81              lblNewLabel_1.setBounds(56, 155, 70, 15);
82              add(lblNewLabel_1);
83
84              txtDOB = new JTextField();
85              txtDOB.setColumns(10);
86              txtDOB.setBounds(118, 155, 114, 19);
87              add(txtDOB);
88              txtDOB.setText(EncryptAndDecrypt.decrypt(values.get("Date of birth")));
89
90
91              JLabel lblNewLabel_1_1 = new JLabel("Address :");
92              lblNewLabel_1_1.setBounds(260, 159, 70, 15);
93              add(lblNewLabel_1_1);
94
95              txtAddrs = new JTextField();
96              txtAddrs.setColumns(10);
97              txtAddrs.setBounds(342, 153, 114, 19);
98              add(txtAddrs);
99              txtAddrs.setText(EncryptAndDecrypt.decrypt(values.get("Date of birth")));
100
101             JLabel lblNewLabel_2 = new JLabel("Type Of Tretmeant :");
102             lblNewLabel_2.setBounds(12, 217, 145, 15);
103             add(lblNewLabel_2);
104
105             txtTreat = new JTextField();
106             txtTreat.setColumns(10);
107             txtTreat.setBounds(170, 215, 286, 19);
108             add(txtTreat);
109             txtTreat.setText(EncryptAndDecrypt.decrypt(values.get("Type of Treatment")));
110
111             txtPres = new JTextField();
112             txtPres.setColumns(10);
113             txtPres.setBounds(170, 262, 286, 19);
114             add(txtPres);
115             txtPres.setText(EncryptAndDecrypt.decrypt(values.get("Prescription")));
116
117             JLabel lblPre = new JLabel("Prescription :");
118             lblPre.setBounds(12, 264, 132, 15);
119             add(lblPre);
120
121
122             JLabel lblReferral = new JLabel("Referral :");
123             lblReferral.setBounds(12, 305, 80, 15);
124             add(lblReferral);
125
126             txtRef = new JTextField();
127             txtRef.setColumns(10);
128             txtRef.setBounds(170, 303, 286, 19);
129             add(txtRef);
130             txtRef.setText(EncryptAndDecrypt.decrypt(values.get("Doc Referral")));
```

```
131
132             }
133         });
134         btnSeach.setBounds(224, 63, 117, 25);
135         add(btnSeach);
136
137
138
139     }
140 }
141
```

Get Info Panel Ph

```
1 package org.eclipse.om2m.smartehealth.gui;
2
3 import java.awt.SystemColor;
21
22 public class GetInfoPanelPh extends JPanel {
23     private JTextField txtHealthCard;
24     private JTextField txtFirstName;
25     private JTextField txtLastName;
26     private JTextField txtDOB;
27     private JTextField txtAddrs;
28     private JTextField txtTreat;
29     private JTextField txtPres;
30     private JTextField txtRef;
31
32     /**
33      * Create the panel.
34      */
35     public GetInfoPanelPh() {
36         setBackground(SystemColor.window);
37         setBounds(0, 0, 480,476);
38         setVisible(true);
39         setLayout(null);
40
41         JLabel lblNewLabel = new JLabel("Health Card Number :");
42         lblNewLabel.setBounds(44, 34, 152, 15);
43         add(lblNewLabel);
44
45         txtHealthCard = new JTextField();
46         txtHealthCard.setBounds(214, 32, 152, 19);
47         add(txtHealthCard);
48         txtHealthCard.setColumns(10);
49
50         JButton btnSeach = new JButton("Search");
51
52         btnSeach.addActionListener(new ActionListener() {
53             public void actionPerformed(ActionEvent e) {
54                 String HealthCard = txtHealthCard.getText();
55                 String xmlString =
   SmartController.getData(SmartConstants.AE_NAME,HealthCard).getContent().toString();
56                 DataParser data = new DataParser();
57
58                 HashMap<String, String> values = data.parseData(xmlString);
59
60                 txtFirstName = new JTextField();
61                 txtFirstName.setColumns(10);
62                 txtFirstName.setBounds(118, 124, 114, 19);
63                 add(txtFirstName);
64                 txtFirstName.setText(EncryptAndDecrypt.decrypt(values.get("Patient name")));
65
66                 JLabel fristName = new JLabel("Frist Name :");
67                 fristName.setBounds(12, 126, 88, 15);
68                 add(fristName);
69                 /*
70                 JLabel lblLastName = new JLabel("LastName :");
71                 lblLastName.setBounds(250, 126, 88, 15);
72                 add(lblLastName);
73
```

```
74                  txtLastName = new JTextField();
75                  txtLastName.setColumns(10);
76                  txtLastName.setBounds(342, 124, 114, 19);
77                  add(txtLastName);
78                  */
79
80                  JLabel lblNewLabel_1 = new JLabel("DOB :");
81                  lblNewLabel_1.setBounds(56, 155, 70, 15);
82                  add(lblNewLabel_1);
83
84                  txtDOB = new JTextField();
85                  txtDOB.setColumns(10);
86                  txtDOB.setBounds(118, 155, 114, 19);
87                  add(txtDOB);
88                  txtDOB.setText(EncryptAndDecrypt.decrypt(values.get("Date of birth")));
89
90
91                  JLabel lblNewLabel_1_1 = new JLabel("Address :");
92                  lblNewLabel_1_1.setBounds(260, 159, 70, 15);
93                  add(lblNewLabel_1_1);
94
95                  txtAddrs = new JTextField();
96                  txtAddrs.setColumns(10);
97                  txtAddrs.setBounds(342, 153, 114, 19);
98                  add(txtAddrs);
99                  txtAddrs.setText(EncryptAndDecrypt.decrypt(values.get("Date of birth")));
100                 /*
101                 JLabel lblNewLabel_2 = new JLabel("Type Of Tretmeant :");
102                 lblNewLabel_2.setBounds(12, 217, 145, 15);
103                 add(lblNewLabel_2);
104
105                 txtTreat = new JTextField();
106                 txtTreat.setColumns(10);
107                 txtTreat.addPnladdPnladdPnladdPnlsetBounds(170, 215, 286, 19);
108                 add(txtTreat);
109                 txtTreat.setText(EncryptAndDecrypt.decrypt(values.get("Type of Treatment")));
110                 */
111                 txtPres = new JTextField();
112                 txtPres.setColumns(10);
113                 txtPres.setBounds(170, 262, 286, 19);
114                 add(txtPres);
115                 txtPres.setText(EncryptAndDecrypt.decrypt(values.get("Prescription")));
116
117                 JLabel lblPre = new JLabel("Prescription :");
118                 lblPre.setBounds(12, 264, 132, 15);
119                 add(lblPre);
120
121                 /*
122                 JLabel lblReferral = new JLabel("Referral :");
123                 lblReferral.setBounds(12, 305, 80, 15);
124                 add(lblReferral);
125
126
127                 txtRef = new JTextField();
128                 txtRef.setColumns(10);
129                 txtRef.setBounds(170, 303, 286, 19);
130                 add(txtRef);
```

```
131                 txtRef.setText(EncryptAndDecrypt.decrypt(values.get("Doc Referral")));
132                 */
133             }
134         });
135         btnSeach.setBounds(224, 63, 117, 25);
136         add(btnSeach);
137
138
139
140     }
141 }
142
```

Login Frame

```
 1 package org.eclipse.om2m.smartehealth.gui;
 2
 3 import java.awt.BorderLayout;
31
32 public class loginFrame extends JFrame {
33
34     private JPanel contentPane;
35     private JTextField txtUsername;
36     private JPasswordField txtPassword;
37     private JLabel lblloginmsg = new JLabel("");
38     private HashMap<String, String> values;
39     /**
40      * Launch the application.
41      */
42     public static void init() {
43         EventQueue.invokeLater(new Runnable() {
44             public void run() {
45                 try {
46                     loginFrame frame = new loginFrame();
47                     frame.setVisible(true);
48                 } catch (Exception e) {
49                     e.printStackTrace();
50                 }
51             }
52         });
53     }
54
55     /**
56      * Create the frame.
57      */
58     public loginFrame() {
59         setUndecorated(true);
60         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
61         setBounds(100, 100, 600,400);
62         contentPane = new JPanel();
63         contentPane.setBackground(new Color(85, 107, 47));
64         contentPane.setBorder(new LineBorder(new Color(0,0, 128) , 2));
65         setContentPane(contentPane);
66         contentPane.setLayout(null);
67
68         JPanel panel = new JPanel();
69         panel.setBackground(Color.WHITE);
70         panel.setBounds(162, 117, 250, 40);
71         contentPane.add(panel);
72         panel.setLayout(null);
73
74         txtUsername = new JTextField();
75         txtUsername.addFocusListener(new FocusAdapter() {
76             @Override
77             public void focusGained(FocusEvent e) {
78                 if(txtUsername.getText().equals("Username")) {
79                     txtUsername.setText("");
80                 }
81                 else {
82                     txtUsername.selectAll();
83                 }
84             }
```

```
85              @Override
86              public void focusLost(FocusEvent e) {
87                  if(txtUsername.getText().equals("")){
88                      txtUsername.setText("Username");
89                  }
90              }
91          });
92          txtUsername.setFont(new Font("DejaVu Sans", Font.BOLD, 12));
93          txtUsername.setText("Username");
94          txtUsername.setBounds(12, 12, 167, 20);
95          txtUsername.setBorder(new LineBorder(new Color(0,0, 0) , 0));
96          panel.add(txtUsername);
97          txtUsername.setColumns(10);
98
99          JPanel panel_1 = new JPanel();
100         panel_1.setBackground(Color.WHITE);
101         panel_1.setBounds(162, 168, 250, 40);
102         contentPane.add(panel_1);
103         panel_1.setLayout(null);
104
105         txtPassword = new JPasswordField();
106         txtPassword.addFocusListener(new FocusAdapter() {
107             @Override
108             public void focusGained(FocusEvent e) {
109                 if(txtPassword.getText().equals("Password")) {
110                     txtPassword.setEchoChar('â€¢');
111                     txtPassword.setText("");
112                 }
113                 else {
114                     txtPassword.selectAll();
115                 }
116             }
117             @Override
118             public void focusLost(FocusEvent e) {
119                 if(txtPassword.getText().equals("")){
120                     txtPassword.setText("Password");
121                     txtPassword.setEchoChar((char)0);
122                 }
123             }
124
125         });
126         txtPassword.setFont(new Font("Dialog", Font.BOLD, 12));
127         txtPassword.setText("Password");
128         txtPassword.setEchoChar((char)0);
129         txtPassword.setBounds(12, 12, 165, 20);
130         txtPassword.setBorder(new LineBorder(new Color(0,0, 0) , 0));
131         panel_1.add(txtPassword);
132
133         JPanel pnlBtnLogin = new JPanel();
134         pnlBtnLogin.addMouseListener(new MouseAdapter() {
135             @Override
136             public void mouseClicked(MouseEvent e) {
137                 String userName = txtUsername.getText();
138                 String xmlString = SmartController.getUser
    (SmartConstants.AE_NAME,userName).getContent().toString();
139                 System.out.println(xmlString);
140                 DataParser data = new DataParser();
```

```
141                 values = data.parseData(xmlString);
142                 AuthenticationController auth = new AuthenticationController();
143
144                 if(auth.checkPassword(txtUsername.getText(), txtPassword.getText(),
    EncryptAndDecrypt.decrypt(values.get("User Data")))) {
145
146
147                     String value = auth.setUserNameandRole(txtUsername.getText(),
    EncryptAndDecrypt.decrypt(values.get("User Data")));
148                     DashBoard DOCdashBoard = new DashBoard();
149                     phDashBoard phDash = new phDashBoard();
150                     if (value.equals("Doctor")){
151                         DOCdashBoard.setVisible(true);
152                         loginFrame.this.dispose();
153                     }else if(value.equals("Front Dask")) {
154                         DOCdashBoard.setVisible(true);
155                         loginFrame.this.dispose();
156                     }else if(value.equals("Pharmacist")) {
157                         phDash.setVisible(true);
158                         loginFrame.this.dispose();
159                     }else if(value.equals("Nurce")) {
160                         DOCdashBoard.setVisible(true);
161                         loginFrame.this.dispose();
162                     }
163
164
165                 }
166                 else if(txtUsername.getText().equals("") || txtUsername.getText().equals
    ("Username") || txtPassword.getText().equals("") ||
167                         txtPassword.getText().equals("") || txtPassword.getText().equals
    ("Password")) {
168                     lblloginmsg.setText("Please Input all requirements");
169                 }
170                 else {
171                     lblloginmsg.setText("Wrong Password or Username! try again");
172                 }
173             }
174         });
175         pnlBtnLogin.setBackground(new Color(128, 128, 128));
176         pnlBtnLogin.setBounds(162, 251, 250, 40);
177         contentPane.add(pnlBtnLogin);
178         pnlBtnLogin.setLayout(null);
179
180         JLabel lblNewLabel = new JLabel("LOG IN");
181         lblNewLabel.setForeground(new Color(224, 255, 255));
182         lblNewLabel.setFont(new Font("Dialog", Font.BOLD, 14));
183         lblNewLabel.setBounds(102, 12, 136, 16);
184         pnlBtnLogin.add(lblNewLabel);
185
186         JLabel lblx = new JLabel("X");
187         lblx.addMouseListener(new MouseAdapter() {
188             @Override
189             public void mouseClicked(MouseEvent e) {
190
191                 if(JOptionPane.showConfirmDialog(null, "Are you sure you want to close this
    applicaiton", "Confirmation", JOptionPane.YES_NO_OPTION) == 0)
192                 {
```

```
193                     loginFrame.this.dispose();
194                 }
195             }
196             @Override
197             public void mouseEntered(MouseEvent e) {
198                 lblx.setForeground(Color.RED);
199             }
200             public void mouseExited(MouseEvent e) {
201                 lblx.setForeground(Color.WHITE);
202             }
203         });
204         lblx.setForeground(Color.WHITE);
205         lblx.setFont(new Font("Dialog", Font.BOLD, 14));
206         lblx.setHorizontalAlignment(SwingConstants.CENTER);
207         lblx.setBounds(574, 0, 26, 20);
208         contentPane.add(lblx);
209 
210         JLabel lblNewLabel_3 = new JLabel("Smart");
211         lblNewLabel_3.setFont(new Font("Abyssinica SIL", Font.BOLD, 24));
212         lblNewLabel_3.setForeground(Color.RED);
213         lblNewLabel_3.setBounds(173, 52, 90, 28);
214         contentPane.add(lblNewLabel_3);
215 
216         JLabel lblNewLabel_2 = new JLabel("E-Health System");
217         lblNewLabel_2.setFont(new Font("DejaVu Sans Mono", Font.BOLD, 16));
218         lblNewLabel_2.setForeground(Color.WHITE);
219         lblNewLabel_2.setBounds(254, 51, 262, 31);
220         contentPane.add(lblNewLabel_2);
221 
222         lblloginmsg.setForeground(Color.RED);
223         lblloginmsg.setFont(new Font("Dialog", Font.BOLD, 13));
224         lblloginmsg.setBounds(119, 224, 333, 15);
225         contentPane.add(lblloginmsg);
226 
227         JPanel pnlBtnSignUp = new JPanel();
228         pnlBtnSignUp.addMouseListener(new MouseAdapter() {
229             @Override
230             public void mouseClicked(MouseEvent e) {
231                 SignUpFrame signUpForm = new SignUpFrame();
232                 signUpForm.setVisible(true);
233                 loginFrame.this.dispose();
234             }
235         });
236         pnlBtnSignUp.setLayout(null);
237         pnlBtnSignUp.setBackground(Color.GRAY);
238         pnlBtnSignUp.setBounds(162, 303, 250, 40);
239         contentPane.add(pnlBtnSignUp);
240 
241         JLabel lblSignUp = new JLabel("SIGN UP");
242         lblSignUp.setForeground(new Color(224, 255, 255));
243         lblSignUp.setFont(new Font("Dialog", Font.BOLD, 14));
244         lblSignUp.setBounds(102, 12, 136, 16);
245         pnlBtnSignUp.add(lblSignUp);
246         setLocationRelativeTo(null);
247 
248 
249     }
250 }
251 
```

Dashboard Ph

```
1 package org.eclipse.om2m.smartehealth.gui;
2
3 import java.awt.BorderLayout;
20
21 public class phDashBoard extends JFrame {
22
23     private JPanel contentPane;
24     private AddInfoPanel addinfoPanal;
25     private editInfoPanel editinfoPan;
26     private GetInfoPanelPh GetInfoPanelPh;
27
28     /**
29      * Launch the application.
30      */
31     public static void main(String[] args) {
32         EventQueue.invokeLater(new Runnable() {
33             public void run() {
34                 try {
35                     phDashBoard frame = new phDashBoard();
36                     frame.setVisible(true);
37                 } catch (Exception e) {
38                     e.printStackTrace();
39                 }
40             }
41         });
42     }
43
44     /**
45      * Create the frame.
46      */
47     public phDashBoard() {
48         setBackground(new Color(0, 128, 128));
49         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
50         setBounds(100, 100, 502, 603);
51         // setUndecorated(true);
52         contentPane = new JPanel();
53         contentPane.setBackground(new Color(0, 139, 139));
54         // contentPane.setBorder(new LineBorder(new Color(0,0, 128) , 2));
55         setContentPane(contentPane);
56         contentPane.setLayout(null);
57         addinfoPanal = new AddInfoPanel();
58
59         editinfoPan = new editInfoPanel();
60
61         GetInfoPanelPh = new GetInfoPanelPh();
62
63         JPanel getPnl = new JPanel();
64         getPnl.setBackground(new Color(0, 100, 0));
65         getPnl.addMouseListener(new PanelButtonMouseAdapter(getPnl) {
66             @Override
67             public void mouseClicked(MouseEvent e) {
68                 menuClicked(GetInfoPanelPh);
69             }
70         });
71         getPnl.setBounds(119, 26, 133, 30);
72         // getPnl.setBorder(new LineBorder(new Color(0,0, 128) , 2));
73         contentPane.add(getPnl);
```

```
74
75          JLabel lblNewLabel_1_1 = new JLabel("GET INFO");
76          lblNewLabel_1_1.setForeground(Color.WHITE);
77          lblNewLabel_1_1.setFont(new Font("Century Schoolbook L", Font.BOLD, 15));
78          getPnl.add(lblNewLabel_1_1);
79          /*
80          JPanel addPnl = new JPanel();
81          addPnl.addMouseListener(new PanelButtonMouseAdapter(addPnl) {
82              @Override
83              public void mouseClicked(MouseEvent e) {
84                  menuClicked(addinfoPanal);
85              }
86          });
87          addPnl.setBackground(new Color(0, 100, 0));
88          // addPnl.setBorder(new LineBorder(new Color(0,0, 128) , 2));
89          addPnl.setForeground(new Color(0, 0, 0));
90          addPnl.setBounds(0, 26, 120, 30);
91          contentPane.add(addPnl);
92
93          JLabel lblNewLabel_1 = new JLabel("ADD");
94          lblNewLabel_1.setForeground(Color.WHITE);
95          lblNewLabel_1.setFont(new Font("Century Schoolbook L", Font.BOLD, 15));
96          addPnl.add(lblNewLabel_1);
97          /*
98          JPanel edietPnl = new JPanel();
99          edietPnl.addMouseListener(new PanelButtonMouseAdapter(edietPnl) {
100             @Override
101             public void mouseClicked(MouseEvent e) {
102                 menuClicked(editinfoPan);
103             }
104         });
105
106         edietPnl.setBackground(new Color(0, 100, 0));
107         edietPnl.setBounds(250, 26, 120, 30);
108         // panel_1.setBorder(new LineBorder(new Color(0,0, 128) , 2));
109         contentPane.add(edietPnl);
110
111         JLabel lblNewLabel_1_2 = new JLabel("EDIT");
112         lblNewLabel_1_2.setForeground(Color.WHITE);
113         lblNewLabel_1_2.setFont(new Font("Century Schoolbook L", Font.BOLD, 15));
114         edietPnl.add(lblNewLabel_1_2);
115         */
116         JLabel lblExit = new JLabel("X");
117         lblExit.addMouseListener(new MouseAdapter() {
118             @Override
119             public void mouseClicked(MouseEvent e) {
120                 if (JOptionPane.showConfirmDialog(null, "Are you sure you want to close this
    applicaiton",
121                         "Confirmation", JOptionPane.YES_NO_OPTION) == 0) {
122                     phDashBoard.this.dispose();
123                 }
124             }
125
126             @Override
127             public void mouseEntered(MouseEvent e) {
128                 lblExit.setForeground(Color.RED);
129             }
```

```
130
131             public void mouseExited(MouseEvent e) {
132                 lblExit.setForeground(Color.WHITE);
133             }
134         });
135         lblExit.setHorizontalAlignment(SwingConstants.CENTER);
136         lblExit.setForeground(Color.WHITE);
137         lblExit.setFont(new Font("Dialog", Font.BOLD, 14));
138         lblExit.setBounds(476, 0, 26, 20);
139         contentPane.add(lblExit);
140
141         JLabel lblNewLabel = new JLabel("Smart E-health Managment System");
142         lblNewLabel.setForeground(Color.WHITE);
143         lblNewLabel.setBounds(12, 5, 411, 15);
144         contentPane.add(lblNewLabel);
145
146         JPanel paneMainContent = new JPanel();
147         paneMainContent.setBounds(10, 68, 480, 476);
148         contentPane.add(paneMainContent);
149
150         JPanel signOutbtn = new JPanel();
151         signOutbtn.addMouseListener(new PanelButtonMouseAdapter(signOutbtn) {
152             @Override
153             public void mouseClicked(MouseEvent e) {
154                 if (JOptionPane.showConfirmDialog(null, "Are you sure you want to sign out ?")
    == 0) {
155                     loginFrame frameLogin = new loginFrame();
156                     frameLogin.setVisible(true);
157                     phDashBoard.this.dispose();
158                 }
159             }
160         });
161         signOutbtn.setBackground(new Color(0, 100, 0));
162         signOutbtn.setBounds(370, 556, 120, 20);
163         contentPane.add(signOutbtn);
164
165         JLabel lblNewLabel_1_2_1 = new JLabel("SIGN OUT");
166         lblNewLabel_1_2_1.setForeground(Color.WHITE);
167         lblNewLabel_1_2_1.setFont(new Font("Century Schoolbook L", Font.BOLD, 13));
168         signOutbtn.add(lblNewLabel_1_2_1);
169         paneMainContent.setLayout(null);
170
171         paneMainContent.add(addinfoPanal);
172         addinfoPanal.setVisible(true);
173         paneMainContent.add(editinfoPan);
174         paneMainContent.add(GetInfoPanelPh);
175         menuClicked(addinfoPanal);
176         menuClicked(addinfoPanal);
177     }
178
179     public void menuClicked(JPanel panel) {
180         addinfoPanal.setVisible(false);
181         //editinfoPan.setVisible(false);
182         GetInfoPanelPh.setVisible(false);
183
184         panel.setVisible(true);
185     }
```

```
186
187     private class PanelButtonMouseAdapter extends MouseAdapter {
188         JPanel panel;
189
190         public PanelButtonMouseAdapter(JPanel panel) {
191             this.panel = panel;
192         }
193
194         @Override
195         public void mouseEntered(MouseEvent e) {
196             panel.setBackground(new Color(112, 128, 144));
197         }
198
199         @Override
200         public void mouseExited(MouseEvent e) {
201             panel.setBackground(new Color(47, 79, 79));
202         }
203
204         @Override
205         public void mousePressed(MouseEvent e) {
206             panel.setBackground(new Color(60, 179, 113));
207         }
208
209         @Override
210         public void mouseReleased(MouseEvent e) {
211             panel.setBackground(new Color(112, 128, 144));
212         }
213     }
214 }
215
```

Signup Frame

```
1 package org.eclipse.om2m.smartehealth.gui;
2
3 import java.awt.BorderLayout;
25
26 public class SignUpFrame extends JFrame {
27
28     private JPanel contentPane;
29     private JTextField txtFristName;
30     private JTextField txtLastName;
31     private JTextField txtUserName;
32     private JTextField txtEmail;
33     private JTextField txtPassword;
34     private JTextField txtCPassword;
35
36     /**
37      * Launch the application.
38      */
39     public static void main(String[] args) {
40         EventQueue.invokeLater(new Runnable() {
41             public void run() {
42                 try {
43                     SignUpFrame frame = new SignUpFrame();
44                     frame.setVisible(true);
45                 } catch (Exception e) {
46                     e.printStackTrace();
47                 }
48             }
49         });
50     }
51
52     /**
53      * Create the frame.
54      */
55     public SignUpFrame() {
56         String [] roleList = {"Doctor" , "Front Dask" , "Nurce", "Pharmacist"};
57         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
58         setBounds(100, 100, 498, 438);
59         contentPane = new JPanel();
60         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
61         contentPane.setBackground(new Color(0, 139, 139));
62         setContentPane(contentPane);
63         contentPane.setLayout(null);
64
65         JLabel lblSmartEhealthManagment = new JLabel("Smart E-health Managment System:
   Registeration  Form");
66         lblSmartEhealthManagment.setForeground(Color.WHITE);
67         lblSmartEhealthManagment.setBounds(12, 12, 411, 15);
68         contentPane.add(lblSmartEhealthManagment);
69
70         JSeparator separator = new JSeparator();
71         separator.setBounds(22, 41, 451, 7);
72         contentPane.add(separator);
73
74         JLabel fristName = new JLabel("Frist Name :");
75         fristName.setBounds(22, 86, 88, 15);
76         contentPane.add(fristName);
77
```

```
78          txtFristName = new JTextField();
79          txtFristName.setColumns(10);
80          txtFristName.setBounds(128, 84, 114, 19);
81          contentPane.add(txtFristName);
82
83          txtLastName = new JTextField();
84          txtLastName.setColumns(10);
85          txtLastName.setBounds(359, 84, 114, 19);
86          contentPane.add(txtLastName);
87
88          JLabel lblLastName = new JLabel("LastName :");
89          lblLastName.setBounds(271, 84, 88, 15);
90          contentPane.add(lblLastName);
91
92          txtUserName = new JTextField();
93          txtUserName.setColumns(10);
94          txtUserName.setBounds(128, 115, 114, 19);
95          contentPane.add(txtUserName);
96
97          JLabel lblUserName = new JLabel("User Name :");
98          lblUserName.setBounds(22, 117, 101, 15);
99          contentPane.add(lblUserName);
100
101         txtEmail = new JTextField();
102         txtEmail.setColumns(10);
103         txtEmail.setBounds(128, 196, 286, 19);
104         contentPane.add(txtEmail);
105
106         JLabel lblEmail = new JLabel("E-Mail  :");
107         lblEmail.setBounds(40, 198, 70, 15);
108         contentPane.add(lblEmail);
109
110         JComboBox CBRole = new JComboBox(roleList);
111         CBRole.setBounds(128, 151, 209, 24);
112         contentPane.add(CBRole);
113
114         JLabel lblRole = new JLabel("Role :");
115         lblRole.setBounds(40, 156, 70, 15);
116         contentPane.add(lblRole);
117
118         txtPassword = new JTextField();
119         txtPassword.setBounds(179, 245, 209, 19);
120         contentPane.add(txtPassword);
121         txtPassword.setColumns(10);
122
123         JLabel lblPassword = new JLabel("Password :");
124         lblPassword.setBounds(79, 247, 99, 15);
125         contentPane.add(lblPassword);
126
127         txtCPassword = new JTextField();
128         txtCPassword.setColumns(10);
129         txtCPassword.setBounds(179, 276, 209, 19);
130         contentPane.add(txtCPassword);
131
132         JLabel lblConfirmPassword = new JLabel("Confirm Password :");
133         lblConfirmPassword.setBounds(22, 274, 137, 17);
134         contentPane.add(lblConfirmPassword);
```

```
135
136         JButton btnReg = new JButton("Register Now");
137         btnReg.addActionListener(new ActionListener() {
138             public void actionPerformed(ActionEvent e) {
139                 String pwd = txtPassword.getText();
140                 String firstName = txtFristName.getText();
141                 String lastName = txtLastName.getText();
142                 String role = String.valueOf(CBRole.getSelectedItem());
143                 String userName =  txtUserName.getText();
144                 String email = txtEmail.getText();
145                 Authentication encPwd = new Authentication();
146
147                 String userPwdDATA = encPwd.addInfo(userName, pwd, role, email ,
    firstName,lastName);
148
149                 SmartController.addUser(firstName,lastName ,userName , role ,userPwdDATA);
150                 if(JOptionPane.showConfirmDialog(null, "Registration succesfull", "Go Back to
    Login Page", JOptionPane.YES_NO_OPTION) == 0)
151                 {
152                     loginFrame frameLogin = new loginFrame();
153                     frameLogin.setVisible(true);
154                     SignUpFrame.this.dispose();
155                 }
156             }
157         });
158         btnReg.setBounds(161, 307, 216, 25);
159         contentPane.add(btnReg);
160     }
161 }
162
```

Activator

```
 1
 2 package org.eclipse.om2m.smartehealth.main;
 3
 4 import org.apache.commons.logging.Log;
13 /**
14  *  Manages the starting and stopping of the bundle.
15  */
16 public class Activator implements BundleActivator {
17     /** Logger */
18     private static Log logger = LogFactory.getLog(Activator.class);
19     /** SCL service tracker */
20     private ServiceTracker<Object, Object> cseServiceTracker;
21
22     @Override
23     public void start(BundleContext bundleContext) throws Exception {
24         logger.info("Register IpeService..");
25         bundleContext.registerService(InterworkingService.class.getName(), new SmartRouter(),
   null);
26         logger.info("IpeService is registered.");
27
28         cseServiceTracker = new ServiceTracker<Object, Object>(bundleContext,
   CseService.class.getName(), null) {
29             public void removedService(ServiceReference<Object> reference, Object service) {
30                 logger.info("CseService removed");
31             }
32
33             public Object addingService(ServiceReference<Object> reference) {
34                 logger.info("CseService discovered");
35                 CseService cseService = (CseService) this.context.getService(reference);
36                 SmartController.setCse(cseService);
37                 new Thread(){
38                     public void run(){
39                         try {
40                             SmartCycleManager.start();
41                         } catch (Exception e) {
42                             logger.error("IpeMonitor Smart E-Health error", e);
43                         }
44                     }
45                 }.start();
46                 return cseService;
47             }
48         };
49         cseServiceTracker.open();
50     }
51
52     @Override
53     public void stop(BundleContext bundleContext) throws Exception {
54         logger.info("Stop Ipe Smart E-Health");
55         try {
56             SmartCycleManager.stop();
57         } catch (Exception e) {
58             logger.error("Stop IPE Smart E-Health error", e);
59         }
60     }
61
62 }
63
```

Data Phaser

```
 1 package org.eclipse.om2m.smartehealth.main;
 2
 3 import java.io.StringReader;
14
15 public class DataParser {
16
17
18     public HashMap<String, String> parseData(String xmlString) {
19         HashMap<String, String> values = new HashMap<String, String>();
20
21         Document xml = convertStringToDocument(xmlString);
22         Node user = xml.getFirstChild();
23         NodeList childs = user.getChildNodes();
24         Node child;
25         for (int i = 0; i < childs.getLength(); i++) {
26             child = childs.item(i);
27             if (!child.getNodeName().equals("#text")) {
28                 //System.out.println(child.getNodeName() + " value: " + child.getTextContent
   ());
29                 getAttributesPairs(child, values);
30                 if (!child.getNodeName().equals("con")) {
31                     values.put(child.getNodeName(), child.getTextContent());
32                     //System.out.println("1");
33                 }
34             }
35         }
36         //System.out.println(values.get("Address"));
37         //System.out.println("Data is: " + values.entrySet());
38         return values;
39     }
40
41     private static void getAttributesPairs(Node child, HashMap<String, String> values) {
42         if (child.getTextContent().contains("xml")) {
43             String value = child.getTextContent();
44
45             Pattern NEWLINE = Pattern.compile("\\R");
46             String lines[] = NEWLINE.split(value);
47             for (String s : lines) {
48                 if (s.contains("<str")) {
49                     String[] vals = s.split(" name=");
50                     String val = "";
51                     String key = "";
52                     for (String a : vals) {
53                         if (a.contains("<str val=")) {
54                             val = a.substring(14, a.length() - 1);
55                             //System.out.println("ddddddddddd: " + a.substring(14, a.length() -
   1));
56
57                         } else if (a.contains("\"/>")) {
58                             key = a.substring(1, a.length() - 3);
59                             //System.out.println("gggggggggg: " + a.substring(1, a.length() -
   3));
60                         }
61
62                     }
63                     values.put(key, val);
64
```

```
65                }
66
67            }
68
69
70        }
71
72    }
73
74    private static Document convertStringToDocument(String xmlStr) {
75        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
76        DocumentBuilder builder;
77        try {
78            builder = factory.newDocumentBuilder();
79            Document doc = builder.parse(new InputSource(new StringReader(xmlStr)));
80            return doc;
81        } catch (Exception e) {
82            e.printStackTrace();
83        }
84        return null;
85    }
86
87 }
88
```

ObixUtil

```
 1
 2 package org.eclipse.om2m.smartehealth.main;
 3
 4 import org.eclipse.om2m.commons.constants.Constants;
13
14 public class ObixUtil {
15
16     /**
17      * Returns an obix XML representation describing the lamp.
18      * @param cseId - SclBase id
19      * @param appId - Application Id
20      * @param stateCont - the STATE container id
21      * @return Obix XML representation
22      */
23     public static String getDescriptorRep(String cseId, String appId, String stateCont) {
24         String prefix = cseId+"/"+ Constants.CSE_NAME + "/" + appId;
25         // oBIX
26         Obj obj = new Obj();
27         obj.add(new Str("Location",StatusRecord.LOCATION));
28
29         /**
30         obj.add(new Str("Patient name",StatusRecord.nameOfPatient ));
31         obj.add(new Str("Health Card Number",StatusRecord.HEALTHCARDNUM));
32         obj.add(new Str("Physician Name",StatusRecord.nameOfPhysician));
33         obj.add(new Str("Type of Treatment",StatusRecord.typeOfTreatment));
34         */
35
36         obj.add(new Str("appId",appId));
37
38
39         // OP GetState from SCL DataBase'
40
41         Op opState = new Op();
42         opState.setName("getState");
43         opState.setHref(new Uri(prefix  +"/"+SmartConstants.DATA+"/"+ ShortName.LATEST));
44         opState.setIs(new Contract("retrieve"));
45         opState.setIn(new Contract("obix:Nil"));
46         opState.setOut(new Contract("obix:Nil"));
47         obj.add(opState);
48
49         return ObixEncoder.toString(obj);
50     }
51
52     /**
53      * Returns an obix XML representation describing the current state.
54      * @param lampId - Application Id
55      * @param value - current lamp state
56      * @return Obix XML representation
57      */
58
59     public static String getStateRep(String lampId, boolean value) {
60         // oBIX
61         Obj obj = new Obj();
62         obj.add(new Str("Date of birth",StatusRecord.LOCATION));
63         obj.add(new Str("Patient name",StatusRecord.nameOfPatient ));
64         obj.add(new Str("Health Card Number",StatusRecord.HEALTHCARDNUM));
65         obj.add(new Str("Physician Name",StatusRecord.nameOfPhysician));
```

```
66          obj.add(new Str("Prescription",StatusRecord.prescription));
67          obj.add(new Str("Type of Treatment",StatusRecord.typeOfTreatment));
68          return ObixEncoder.toString(obj);
69
70      }
71
72
73      public static String getStateRep(String appID, String randIdNum, String namePatient,
    String loc , String treatType, String  preOrder, String phName , String addrs , String ref) {
74          // oBIX
75          String prefix = Constants.CSE_ID+"/"+ Constants.CSE_NAME + "/" + appID;
76          String strID = "" + randIdNum;
77          Obj obj = new Obj();
78          obj.add(new Str("Date of birth",loc));
79          obj.add(new Str("Patient name",namePatient));
80          obj.add(new Str("Health Card Number", strID));
81          obj.add(new Str("Physician Name",phName));
82          obj.add(new Str("Prescription",preOrder));
83          obj.add(new Str("Type of Treatment",treatType));
84          obj.add(new Str("Address" , addrs));
85          obj.add(new Str("Doc Referral" , ref));
86
87          Op opState = new Op();
88          opState.setName("getState");
89          opState.setHref(new Uri(prefix  +"/"+SmartConstants.DATA+"/"+ ShortName.LATEST));
90          opState.setIs(new Contract("retrieve"));
91          opState.setIn(new Contract("obix:Nil"));
92          opState.setOut(new Contract("obix:Nil"));
93          obj.add(opState);
94          return ObixEncoder.toString(obj);
95
96      }
97
98      public static String getAccessControlRep(String firstName, String lastName , String
    UserName, String role, String salt) {
99          Obj obj = new Obj();
100         obj.add(new Str("Username",UserName));
101         obj.add(new Str("Name", firstName + " " + lastName));
102         obj.add(new Str("Role",role));
103         obj.add(new Str("User Data", salt));
104         return ObixEncoder.toString(obj);
105     }
106
107
108 }
109
```

Operations

```
 1
 2 package org.eclipse.om2m.smartehealth.main;
 3
 4 import org.eclipse.om2m.commons.exceptions.BadRequestException;
 5 /**
 6  * Represent a operation
 7  *
 8  */
 9 public enum Operations {
10
11     GET_STATE("getData"),
12     SET_DATA("setDATA"),
13     UPDATE_DATA("updateData"),
14     DELETE_DATA("delateData");
15     private final String value;
16
17     private Operations(final String value){
18         this.value = value;
19     }
20
21     public String toString() {
22         return value;
23     }
24
25     public String getValue(){
26         return value;
27     }
28
29     /**
30      * Return the operation from the string
31      * @param operation
32      * @return
33      */
34     public static Operations getOperationFromString(String operation){
35         for(Operations op : values()){
36             if(op.getValue().equals(operation)){
37                 return op;
38             }
39         }
40         throw new BadRequestException("Unknow Operation");
41     }
42 }
43
```

Password

```
1 package org.eclipse.om2m.smartehealth.main;
2
3 import java.security.NoSuchAlgorithmException;
11
12 /**
13  * A utility class to hash passwords and check passwords vs hashed values. It uses a
  combination of hashing and unique
14  * salt. The algorithm used is PBKDF2WithHmacSHA1 which, although not the best for hashing
  password (vs. bcrypt) is
15  * still considered robust and <a href="https://security.stackexchange.com/a/6415/12614">
  recommended by NIST </a>.
16  * The hashed value has 256 bits.
17  */
18 public class Password {
19
20   private static final Random RANDOM = new SecureRandom();
21   private static final int ITERATIONS = 10000;
22   private static final int KEY_LENGTH = 256;
23
24   /**
25    * static utility class
26    */
27   Password() { }
28
29   /**
30    * Returns a random salt to be used to hash a password.
31    *
32    * @return a 16 bytes random salt
33    */
34   public static byte[] getNextSalt() {
35     byte[] salt = new byte[16];
36     RANDOM.nextBytes(salt);
37     return salt;
38   }
39
40   /**
41    * Returns a salted and hashed password using the provided hash.<br>
42    * Note - side effect: the password is destroyed (the char[] is filled with zeros)
43    *
44    * @param password the password to be hashed
45    * @param salt     a 16 bytes salt, ideally obtained with the getNextSalt method
46    *
47    * @return the hashed password with a pinch of salt
48    */
49   public static byte[] hash(char[] password, byte[] salt) {
50     PBEKeySpec spec = new PBEKeySpec(password, salt, ITERATIONS, KEY_LENGTH);
51     Arrays.fill(password, Character.MIN_VALUE);
52     try {
53       SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
54       return skf.generateSecret(spec).getEncoded();
55     } catch (NoSuchAlgorithmException | InvalidKeySpecException e) {
56       throw new AssertionError("Error while hashing a password: " + e.getMessage(), e);
57     } finally {
58       spec.clearPassword();
59     }
60   }
61
```

```
62  /**
63   * Returns true if the given password and salt match the hashed value, false otherwise.<br>
64   * Note - side effect: the password is destroyed (the char[] is filled with zeros)
65   *
66   * @param password     the password to check
67   * @param salt         the salt used to hash the password
68   * @param expectedHash the expected hashed value of the password
69   *
70   * @return true if the given password and salt match the hashed value, false otherwise
71   */
72  public static boolean isExpectedPassword(char[] password, byte[] salt, byte[] expectedHash)
    {
73    byte[] pwdHash = hash(password, salt);
74    Arrays.fill(password, Character.MIN_VALUE);
75    if (pwdHash.length != expectedHash.length) return false;
76    for (int i = 0; i < pwdHash.length; i++) {
77      if (pwdHash[i] != expectedHash[i]) return false;
78    }
79    return true;
80  }
81
82  /**
83   * Generates a random password of a given length, using letters and digits.
84   *
85   * @param length the length of the password
86   *
87   * @return a random password
88   */
89  public static String generateRandomPassword(int length) {
90    StringBuilder sb = new StringBuilder(length);
91    for (int i = 0; i < length; i++) {
92      int c = RANDOM.nextInt(62);
93      if (c <= 9) {
94        sb.append(String.valueOf(c));
95      } else if (c < 36) {
96        sb.append((char) ('a' + c - 10));
97      } else {
98        sb.append((char) ('A' + c - 36));
99      }
100   }
101   return sb.toString();
102 }
103 }
```

Read XML File

```
1 package org.eclipse.om2m.smartehealth.main;
2
3 import javax.xml.parsers.DocumentBuilderFactory;
10
11 public class ReadXMLFile {
12
13     public static void main(String[] args) {
14         // TODO Auto-generated method stub
15          try {
16
17
18                 File fXmlFile = new File("/home/tinsae/Desktop/Update/staff.xml");
19                 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
20                 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
21                 Document doc = dBuilder.parse(fXmlFile);
22
23                 //optional, but recommended
24                 //read this - http://stackoverflow.com/questions/13786607/normalization-in-dom-
  parsing-with-java-how-does-it-work
25                 doc.getDocumentElement().normalize();
26
27                 System.out.println("Root element :" + doc.getDocumentElement().getNodeName());
28
29                 NodeList nList = doc.getElementsByTagName("staff");
30
31                 System.out.println("----------------------------");
32                /*
33                 for (int temp = 0; temp < nList.getLength(); temp++) {
34
35                     Node nNode = nList.item(temp);
36
37                     System.out.println("\nCurrent Element :" + nNode.getNodeName());
38
39                     if (nNode.getNodeType() == Node.ELEMENT_NODE) {
40
41                         Element eElement = (Element) nNode;
42
43                         System.out.println("Staff id : " + eElement.getAttribute("id"));
44                         System.out.println("First Name : " + eElement.getElementsByTagName
  ("firstname").item(0).getTextContent());
45                         System.out.println("Last Name : " + eElement.getElementsByTagName
  ("lastname").item(0).getTextContent());
46                         System.out.println("Nick Name : " + eElement.getElementsByTagName
  ("nickname").item(0).getTextContent());
47                         System.out.println("Salary : " + eElement.getElementsByTagName
  ("salary").item(0).getTextContent());
48
49                     }
50                 }
51                 */
52                 } catch (Exception e) {
53                 e.printStackTrace();
54                 }
55          }
56
57 }
58
```

Request Sender

```
1
2 package org.eclipse.om2m.smartehealth.main;
3
4 import java.math.BigInteger;
16
17
18 public class RequestSender {
19
20     /**
21      * Private constructor to avoid creation of this object
22      */
23     private RequestSender(){}
24
25     public static ResponsePrimitive createResource(String targetId, Resource resource, int
  resourceType){
26         RequestPrimitive request = new RequestPrimitive();
27         request.setFrom(Constants.ADMIN_REQUESTING_ENTITY);
28         request.setTo(targetId);
29         request.setResourceType(BigInteger.valueOf(resourceType));
30         request.setRequestContentType(MimeMediaType.OBJ);
31         request.setContent(resource);
32         request.setOperation(Operation.CREATE);
33         return SmartController.CSE.doRequest(request);
34     }
35
36     public static ResponsePrimitive createAE(AE resource){
37         return createResource("/" + Constants.CSE_ID, resource, ResourceType.AE);
38     }
39
40     public static ResponsePrimitive createContainer(String targetId, Container resource){
41         return createResource(targetId, resource, ResourceType.CONTAINER);
42     }
43
44     public static ResponsePrimitive createContentInstance(String targetId, ContentInstance
  resource){
45         return createResource(targetId, resource, ResourceType.CONTENT_INSTANCE);
46     }
47
48
49     public static ResponsePrimitive getRequest(String targetId){
50         RequestPrimitive request = new RequestPrimitive();
51         request.setFrom(Constants.ADMIN_REQUESTING_ENTITY);
52         request.setTo(targetId);
53         request.setOperation(Operation.RETRIEVE);
54         request.setReturnContentType(MimeMediaType.XML);
55         return SmartController.CSE.doRequest(request);
56     }
57
58 }
59
```

Smart Constraints

```
1
2 package org.eclipse.om2m.smartehealth.main;
3
4 import org.eclipse.om2m.commons.constants.Constants;
5
6 public class SmartConstants {
7
8     private SmartConstants(){}
9
10     public static final String POA = "ehealth";
11     public static final String DATA = "DATA";
12     public static final String DESC = "DESCRIPTOR";
13     public static final String AE_NAME = "SMART_E-HEALTH";
14     public static final String ACP = "ACCESSCONTROL";
15
16     public static String CSE_ID = "/" + Constants.CSE_ID;
17     public static String CSE_PREFIX = CSE_ID + "/" + Constants.CSE_NAME;
18     public static final boolean GUI =
  Boolean.valueOf(System.getProperty("org.eclipse.om2m.smartehealth", "true"));
19
20 }
21
```

Smart Controller

```
1
2 package org.eclipse.om2m.smartehealth.main;
3
4 import org.eclipse.om2m.SmartSecurity.controller.EncryptAndDecrypt;
10
11
12 public class SmartController {
13
14     public static CseService CSE;
15     protected static String AE_ID;
16
17     public static void setPatient(String appID, String randIdNum,String namePatient, String loc
  , String treatType, String preOrder, String phName , String addrs , String ref){
18         // Set the value in the "real world" model
19             // Send the information to the CSE
20         String targetID = SmartConstants.CSE_PREFIX + "/" + appID + "/" + SmartConstants.DATA ;
21         ContentInstance cin = new ContentInstance();
22         //encryption
23         EncryptAndDecrypt decEnc = new EncryptAndDecrypt();
24         String hcardNum = decEnc.encrypt(randIdNum);
25         namePatient = decEnc.encrypt(namePatient);
26         loc = decEnc.encrypt(loc);
27         treatType = decEnc.encrypt(treatType);
28         preOrder = decEnc.encrypt(preOrder);
29         phName = decEnc.encrypt(phName);
30         addrs = decEnc.encrypt(addrs);
31         ref = decEnc.encrypt(ref);
32         cin.setContent(ObixUtil.getStateRep(appID, hcardNum,namePatient, loc , treatType,
  preOrder, phName , addrs , ref));
33         cin.setName(randIdNum);
34         cin.setContentInfo(MimeMediaType.OBIX + ":" + MimeMediaType.ENCOD_PLAIN);
35         RequestSender.createContentInstance(targetID, cin);
36     }
37
38     public static void addUserInfo(String appID,String firstName, String lastName , String
  UserName , String role , String userData ){
39         // Set the value in the "real world" mode
40             // Send the information to the CSE
41         String targetID = SmartConstants.CSE_PREFIX + "/" + appID + "/" + SmartConstants.ACP;
42         ContentInstance cin = new ContentInstance();
43         //encryption
44         EncryptAndDecrypt decEnc = new EncryptAndDecrypt();
45         //namePatient = decEnc.encrypt(namePatient);
46         //loc = decEnc.encrypt(loc);
47         //treatType = decEnc.encrypt(treatType);
48         //preOrder = decEnc.encrypt(preOrder);
49         //phName = decEnc.encrypt(phName);
50         //addrs = decEnc.encrypt(addrs);
51          userData = decEnc.encrypt(userData);
52         cin.setContent(ObixUtil.getAccessControlRep(firstName, lastName ,UserName, role ,
  userData));
53         cin.setName(UserName);
54         cin.setContentInfo(MimeMediaType.OBIX + ":" + MimeMediaType.ENCOD_PLAIN);
55         RequestSender.createContentInstance(targetID, cin);
56     }
57
58
```

```
59     public static ResponsePrimitive getData(String appID, String randNum) {
60         String str = randNum;
61         String targetId = SmartConstants.CSE_PREFIX + "/" + appID + "/" + SmartConstants.DATA
  +"/" + str;
62         return RequestSender.getRequest(targetId);
63     }
64     public static ResponsePrimitive getUser(String appID, String randNum) {
65         String str = randNum;
66         String targetId = SmartConstants.CSE_PREFIX + "/" + appID + "/" + SmartConstants.ACP
  +"/" + str;
67         return RequestSender.getRequest(targetId);
68     }
69
70     public static void addData(String randIdNum,String namePatient, String loc , String
  treatType, String preOrder, String phName, String addrs , String ref ){
71         setPatient(SmartConstants.AE_NAME,randIdNum,namePatient, loc , treatType, preOrder,
  phName , addrs , ref );
72     }
73     public static void addUser(String firstName, String lastName , String UserName, String role
  , String userPwdDATA) {
74         addUserInfo(SmartConstants.AE_NAME,firstName, lastName ,UserName, role , userPwdDATA);
75     }
76
77
78     public static void setCse(CseService cse){
79         CSE = cse;
80     }
81
82
83 }
84
```

Smart Cycle Manager

```
 1
 2 package org.eclipse.om2m.smartehealth.main;
 3
 4 import java.math.BigInteger;
18
19
20
21
22 public class SmartCycleManager {
23
24     private static Log LOGGER = LogFactory.getLog(SmartCycleManager.class);
25     private static Scanner sc;
26     private static String namePatient;
27     private static String loc;
28     private static String treatType;
29
30     /**
31      * Handle the start of the plugin with the resource representation and the GUI
32      */
33     public static void start(){
34
35         // Create initial resources for smart health
36
37         String appID = "SMART_E-HEALTH";
38         createSmartResources(appID, false, SmartConstants.POA);
39
40
41         //startInterface(appID);
42
43         if(SmartConstants.GUI){
44             loginFrame.init();
45         }
46
47
48     }
49
50     /**
51      * Stop the GUI if it is present
52      */
53     public static void stop(){
54
55     }
56
57     /**
58      * Creates all required resources.
59      * @param appId - Application ID
60      * @param initValue - initial lamp value
61      * @param poa - lamp Point of Access
62      */
63     private static void createSmartResources(String appId, boolean initValue, String poa) {
64         // Create the Application resource
65         Container container = new Container();
66         container.getLabels().add("smart_E-HEALTH");
67         container.setMaxNrOfInstances(BigInteger.valueOf(0));
68
69
70         AE ae = new AE();
```

```
71          ae.setRequestReachability(true);
72          ae.getPointOfAccess().add(poa);
73          ae.setAppID(appId);
74          ae.setName(appId);
75
76          ResponsePrimitive response = RequestSender.createAE(ae);
77          // Create Application sub-resources only if application not yet created
78          if(response.getResponseStatusCode().equals(ResponseStatusCode.CREATED)) {
79              container = new Container();
80              container.setMaxNrOfInstances(BigInteger.valueOf(10));
81              // Create DESCRIPTOR container sub-resource
82              container.setName(SmartConstants.DESC);
83              LOGGER.info(RequestSender.createContainer(response.getLocation(), container));
84              // Create DETA container sub-resource
85              container.setName(SmartConstants.DATA);
86              LOGGER.info(RequestSender.createContainer(response.getLocation(), container));
87              // Create APC container Sub-resource
88              container.setName(SmartConstants.ACP);
89              LOGGER.info(RequestSender.createContainer(response.getLocation(), container));
90
91
92              String content;
93              // Create DESCRIPTION contentInstance on the DESCRIPTOR container resource
94              content = ObixUtil.getDescriptorRep(SmartConstants.CSE_ID, appId,
   SmartConstants.DATA);
95              ContentInstance contentInstance = new ContentInstance();
96              contentInstance.setContent(content);
97              //contentInstance.setName("Happy");
98              contentInstance.setContentInfo(MimeMediaType.OBIX);
99              RequestSender.createContentInstance(SmartConstants.CSE_PREFIX + "/" + appId + "/"
   + SmartConstants.DESC, contentInstance);
100
101
102             // Create initial contentInstance on the STATE container resource
103
104             content = ObixUtil.getStateRep(appId, initValue);
105             contentInstance.setContent(content);
106             //RequestSender.createContentInstance(SmartConstants.CSE_PREFIX + "/" + appId +
   "/" + SmartConstants.DATA, contentInstance);
107         }
108
109
110
111     }
112
113 }
114
```

Smart Router

```
1
2 package org.eclipse.om2m.smartehealth.main;
3
4 import org.apache.commons.logging.Log;
12
13 public class SmartRouter implements InterworkingService{
14
15     private static Log LOGGER = LogFactory.getLog(SmartRouter.class);
16
17     @Override
18     public ResponsePrimitive doExecute(RequestPrimitive request) {
19         ResponsePrimitive response = new ResponsePrimitive(request);
20         if(request.getQueryStrings().containsKey("op")){
21             String operation = request.getQueryStrings().get("op").get(0);
22             //Operations op = Operations.getOperationFromString(operation);
23             String sysID= null;
24             if(request.getQueryStrings().containsKey("sysID")){
25                 sysID = request.getQueryStrings().get("sysID").get(0);
26             }
27             LOGGER.info("Received request in Smart E-healthIPE: op=" + operation + " ; sysID="
  + sysID);
28
29
30         response.setResponseStatusCode(ResponseStatusCode.OK);
31         return response;
32         }
33         return response;
34     }
35
36     @Override
37     public String getAPOCPath() {
38         return SmartConstants.POA;
39     }
40
41 }
42
```

Status Record

```
1
2 package org.eclipse.om2m.smartehealth.main;
3
4 public class StatusRecord {
5
6     /**Location */
7     public final static String LOCATION = "Monfort Hospital ";
8     /** Toggle */
9     public final static String nameOfPatient  =  "John Mark";
10    /** Default type */
11    public final static String typeOfTreatment = "Radiation therapy";
12    /** health card number */
13    public final static String HEALTHCARDNUM = "101034727";
14
15    public final static String prescription = "Tramadol (Ultram)";
16
17    public final static String nameOfPhysician = "Dr. bereket";
18
19
20    private String healthCardNumber;
21    private String location;
22    private String patientName;
23    private String treatmentType;
24    private String physicianName;
25    private String prescriptionOrder;
26
27
28
29
30
31    private boolean state = false;
32    /** app ID */
33    private String appID;
34
35    public StatusRecord() {
36
37    }
38    public StatusRecord(String appID, String HCN, String loc, String prOrder, String pName ,
  String phName)
39    {
40        this.appID = appID;
41        this.healthCardNumber = HCN;
42        this.setLocation(loc) ;
43        this.setPatientName(pName);
44        this.setPhysicianName(phName);
45        this.setPrescriptionOrder(prOrder);
46
47    }
48
49
50
51    /**
52     * @return the state
53     */
54    public void setPatientName(String pt) {
55        this.patientName = pt;
56    }
```

```
57
58      public String getPatientName() {
59          return patientName;
60      }
61
62      public String getHealthCardNum() {
63          return healthCardNumber;
64      }
65
66      /**
67       * @param state the state to set
68       */
69
70      public void setState(boolean state) {
71          this.state = state;
72      }
73
74      /**
75       * @return the appID
76       */
77      public String getappID() {
78          return appID;
79      }
80
81      /**
82       * @param appID the appID to set
83       */
84      public void setappID(String appID) {
85          this.appID = appID;
86      }
87
88
89
90      public String getLocation() {
91          return location;
92      }
93
94
95
96      public void setLocation(String location) {
97          this.location = location;
98      }
99
100
101
102     public String getTreatmentType() {
103         return treatmentType;
104     }
105
106
107
108     public void setTreatmentType(String treatmentType) {
109         this.treatmentType = treatmentType;
110     }
111
112
113
```

```
114     public String getPrescriptionOrder() {
115         return prescriptionOrder;
116     }
117
118
119
120     public void setPrescriptionOrder(String prescriptionOrder) {
121         this.prescriptionOrder = prescriptionOrder;
122     }
123
124
125
126     public String getPhysicianName() {
127         return physicianName;
128     }
129
130
131
132     public void setPhysicianName(String physicianName) {
133         this.physicianName = physicianName;
134     }
135
136
137 }
138
```

Trial

```
1 package org.eclipse.om2m.smartehealth.main;
2
3 import org.eclipse.om2m.SmartSecurity.controller.AuthenticationController;
5
6 public class Trial {
7
8     public static void main(String[] args) {
9         // TODO Auto-generated method stub
10            String Username = "tinsae" ;
11            String password = "Tinsae82@" ;
12            String role = "Doctor";
13            String contactinfo = "tindebeb94@gmail.com";
14            String address = "601 Laurier" ;
15            String healthCardNumber = "101010" ;
16            String medicaliinfo = "triatment";
17
18
19             Authentication encPwd = new Authentication();
20             String[] returnVa;
21
22
23             String medicalliinfo2 = encPwd.addInfo
  (Username,password,role,contactinfo,address,healthCardNumber);
24
25             AuthenticationController auth = new AuthenticationController();
26             System.out.println("value :" + auth.checkPassword(Username, password,
  medicalliinfo2));
27             //System.out.println("salt :" + authoA[1]);
28             //System.out.println(Password.getNextSalt());
29
30     }
31
32 }
```

Access Controller

```
1 package org.eclipse.om2m.SmartSecurity.controller;
2
3 import org.eclipse.om2m.SmartSecurity.model.AccessControl;
4
5
6 public class AccessController {
7     private AccessControl model;
8     //private AccessUI view;
9
10    String [] Title = {"technical support","Patient","Nurse","Doctor"};
11    String [] Objects = {"PatientPersonalInfo","PatientMedicalInfo","StaffInfo"};
12    String [] Access = {"0211","0112","2111"};
13    String [] Actions = {"seek","get","modify"};
14    LinkedList [] locations = new LinkedList[3];
15    public AccessController() {
16
17        //this.view = view;
18
19        for(int i = 0; i < locations.length;i++) {
20            locations[i] = new LinkedList(Objects[i],Title,Access[i]);
21        }
22    }
23
24    public boolean checkAccess (String requester, String obj, String action){
25        for(int i = 0; i < locations.length;i++) {
26            if(locations[i].listname.equalsIgnoreCase(obj)) {
27                if(locations[i].GrantAccess(requester,action)) {
28                System.out.println("ACCESS GRANTED");
29                return true;
30                }
31                else {
32                    System.out.println("ACCESS DENIED");
33                    break;
34                }
35            }
36        }
37        return false;
38    }
39 }
```

Authentication Controller

```
1 package org.eclipse.om2m.SmartSecurity.controller;
2
3
4 import java.security.MessageDigest;
8
9
10 public class AuthenticationController {
11
12     private Authentication auth = new Authentication();
13
14     public boolean checkUser(String username, String userDeatil){
15
16         return auth.checkUsername(username , userDeatil);
17     }
18     public void startup() {}
19     public boolean checkPassword(String name,String password , String userDATA) {
20         MessageDigest hashAlg;
21         try {
22             //getting the User details to take out the salt to encrypt and the password to
  verify.
23             Authentication Auth = new Authentication();
24             String []originalPass = Auth.getPasswordandSalt(name , userDATA);
25             byte [] salt = new byte[64];
26             byte [] Password = new byte[32];
27             String [] temp = (originalPass[1].substring(1, originalPass[1].length()-1)).split
  (",");
28             String [] temp2 = (originalPass[0].substring(1, originalPass[0].length()-1)).split
  (",");
29             for (int i = 0; i <salt.length;i++) {
30                 salt[i] =  Byte.parseByte(temp[i].replace(" ", ""));
31                 if(i % 2 == 0) {Password[i/2] =  Byte.parseByte(temp2[i/2].replace(" ", ""));}
32             }
33             hashAlg = MessageDigest.getInstance("SHA-256");
34             hashAlg.update(salt);
35             if(originalPass[0].equals(Arrays.toString(hashAlg.digest(password.getBytes()))) {
  return true; }
36
37         } catch (NoSuchAlgorithmException e) {
38             e.printStackTrace();
39         }
40
41         return false;
42     }
43
44     public String setUserNameandRole(String username , String userDetail) {
45         return auth.setUserNameandRole(username , userDetail);
46     }
47
48
49 }
```

Encryption and Decryption

```
1 package org.eclipse.om2m.SmartSecurity.controller;
2 import java.io.UnsupportedEncodingException;
9
10 public class EncryptAndDecrypt {
11
12     private static SecretKeySpec secretKey;
13     private static byte[] key;
14
15     public static void setKey()
16     {
17         String myKey = "12345678901234567890";
18         MessageDigest sha = null;
19         try {
20             key = myKey.getBytes("UTF-8");
21             sha = MessageDigest.getInstance("SHA-256");
22             key = sha.digest(key);
23             key = Arrays.copyOf(key, 16);
24             secretKey = new SecretKeySpec(key, "AES");
25         }
26         catch (NoSuchAlgorithmException e) {
27             e.printStackTrace();
28         }
29         catch (UnsupportedEncodingException e) {
30             e.printStackTrace();
31         }
32     }
33
34     public static String encrypt(String strToEncrypt)
35     {
36         try
37         {
38             setKey();
39             Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
40             cipher.init(Cipher.ENCRYPT_MODE, secretKey);
41             return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes
  ("UTF-8")));
42         }
43         catch (Exception e)
44         {
45             System.out.println("Error while encrypting: " + e.toString());
46         }
47         return null;
48     }
49
50     public static String decrypt(String strToDecrypt)
51     {
52         try
53         {
54             setKey();
55             Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
56             cipher.init(Cipher.DECRYPT_MODE, secretKey);
57             return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
58         }
59         catch (Exception e)
60         {
61             System.out.println("Error while decrypting: " + e.toString());
62         }
63         return null;
64     }
65 }
```

LinkedList

```java
package org.eclipse.om2m.SmartSecurity.controller;

public class LinkedList {
    Node head; // head
    String listname;
    String [] Actions = {"seek","get","modify"};
    // the node class I will need for the solution
    static class Node {

        String name;
        int number;
        Node next;

        Node(String n,int d)
        {
            name = n;
            number = d;
            next = null;
        }
    }

    // Initializes the LinkedList into the sets I need exactly
    public LinkedList(String list,String [] name,String codes) {
        this.listname = list;
        for(int i = 0; i < name.length; i++) {
            int a = Character.getNumericValue(codes.charAt(i));
            if(a != 4) {
                this.insert(name[i], a);
            }
        }
    }

    // inserts a new node into the list
    public void insert(String name, int number) {
        Node new_node = new Node(name,number);
        new_node.next = null;

        if (head == null) {
            head = new_node;
        }
        else {
            Node last = head;
            while (last.next != null) {
                last = last.next;
            }
            last.next = new_node;
        }

    }

    //grants access to the user if the request is with the limit of the access of said title
    public boolean GrantAccess (String name, String access){
        System.out.println("Checking Access to "+ listname +" for a " + name + " wanting to " +
access );
        for(int i = 0; i < Actions.length;i++){
            if(Actions[i].toLowerCase().contentEquals(access.toLowerCase())) {
                int a = this.GetAccess(name);
```

```
                if(a >= i ) {
                    return true;
                }
            }
        }
            return false;
    }

    //gets the access level of the given title
    private int GetAccess(String name){
        Node currNode = this.head;
        while (currNode != null) {
            if(currNode.name.equals(name)){
                return currNode.number;}
            currNode = currNode.next;
        }
        return -1;
    }


    public String getAccessName (String name) {
        int temp = this.GetAccess(name);
        if(temp != -1) {
        return Actions[temp];}
        else return null;

    }
}
```

Authentication

```
1 package org.eclipse.om2m.SmartSecurity.model;
2
3 import java.security.MessageDigest;
4 import java.security.SecureRandom;
5 import java.util.Arrays;
6
7 public class Authentication {
8
9     private String username;
10    private String role;
11
12    public Authentication() {
13    }
14
15    public String getuserName() {
16        return username;
17    }
18
19    public String getRole() {
20        return role;
21    }
22
23    public String[] getUserDetails(String Username, String Files) {
24        String line = "";
25        String[] details = new String[5];
26        //Retrieves the users details based of the username
27        //BufferedReader reader = new BufferedReader(new FileReader("details.txt"));
28        line = Files;
29
30        //System.out.println(Files);
31        while(line != null) {
32            details = line.split(":");
33            return details;
34        }
35        //reader.close();
36        //System.out.println(details);
37        return null;
38    }
39
40    public String addInfo(String Username,String password,String role, String
  contactinfo,String address,String healthCardNumber) {
41
42            this.username = Username;
43            this.role = role;
44            SecureRandom rand = new SecureRandom();
45            byte[] salt = new byte[64];
46            byte [] Password;
47            String Out = "";
48            try {
49                // Making the salt and putting it in the hash alogrithm SHA-256 Salt is 64
  bytes
50                MessageDigest hashAlg = MessageDigest.getInstance("SHA-256");
51                rand.nextBytes(salt);
52                hashAlg.update(salt);
53                Password = hashAlg.digest(password.getBytes());
54                //making a string of the user details to upload to the server password and
  authintcaiton files, format is item:item:item
```

```
55                Out = Username + ":" + Arrays.toString(Password) + ":" + Arrays.toString(salt)
  + ":" + role+ ":" + contactinfo+ ":" + address+ ":" + healthCardNumber;
56                //Writing to the passwd.txt file
57            }
58            catch (Exception e1) {
59            }
60            return Out;
61        }
62
63    public boolean checkUsername(String username ,String datafile) {
64
65        String line = "";
66        String details;
67        while(line != null) {
68            details = line.split(":")[0].replace(" ", "");
69            if(details.toLowerCase().equals(username.toLowerCase())) {
70                return false;
71            }
72
73        }
74
75        return true;
76    }
77
78    public String [] getPasswordandSalt(String name , String userDetail) {
79        String [] info = this.getUserDetails(name , userDetail);
80        String [] values = new String [2];
81        values[0] = info[1];
82        values[1] = info[2];
83        this.username = info[0];
84        this.role = info[3];
85        return values;
86        }
87
88
89    public String setUserNameandRole(String username , String userDetail) {
90        this.username = username;
91        String [] details = this.getUserDetails(username , userDetail);
92        this.role = details[3];
93        return this.role;
94    }
95
96 }
```

Access Control

```
1 package org.eclipse.om2m.SmartSecurity.model;
2
3 public class AccessControl {
4
5     private String username;
6     private String role;
7     public AccessControl(String username,String role) {
8
9         this.role = role;
10        this.username = username;
11    }
12    public String getUsername() {
13        return username;
14    }
15    public String getRole() {
16        return role;
17    }
18    public void setUsername(String username) {
19        this.username = username;
20    }
21    public void setRole(String role) {
22        this.role = role;
23    }
24 }
```

Appendix B – Project Proposal

PROJECT PROPOSAL

SYSC 4907 Fourth Year Project

Supervisor

Mohamed Ibn Kahla

Coordinators

Ali Farhat

Abdelrahman Eldosouky

APRIL 14, 2021

YOHANNES KUSSIA,TINSAE DEBELE, MOHAMMED ISMAIL, KAMRAN SAGHEIR

Professor Mohamed Ibn Kahla

Abstract

One of the main concerns of Internet of Things (IoT) is machine to machine (M2M) communication. The objective of M2M communication is to establish interactive links among billions of heterogeneous devices capable of interacting with each other. For machine-to-machine related communication, the European Telecommunications Standards Institute (ETSI) released specifications for the machine-to-machine related communication. The open source OM2M project provides ETSI-compliant M2M service platform. The OM2M is widely used for communication among the IoT-enabled e-Health Systems [1]. The IoT-enabled e-Health Systems are used for the self and remote monitoring for several health-related conditions and the use is growing rapidly, day by day. Similarly, red hat hackers and cyber criminals are trying to make these devices unsafe for users, and it is raising a lot of privacy and security concerns for both, users and manufacturers [2]. The OM2M platform barely addresses the security and privacy issues that are necessary for safe transfer of information, and data at a large scale. In order to make the IoT-enabled e-Health Systems more safe and secure, an extension to the OM2M platform is proposed under the OM2M security management layer.

1. Introduction

One of the most prominent areas of Internet of Things (IoT) is electronic health systems. A lot of IoT enabled electronic healthcare devices exist and are used on daily basis by masses. The core role of IoT in electronic healthcare devices is their ability to monitor, record, and transmit data related to patients to doctors, nurses or family members[2]. This data is transmitted from

the devices to the hospital servers over wireless networks from wherever the patient uses the device from. This makes it easier to respond to the patients in case of emergencies for the health care professionals. This can help in saving the lives of the patients in case the patient is suffering from life threating situation [2].

2. Background Information:

IoT has a vast use in healthcare areas such as health monitoring, fitness programs, chronic diseases, and elder care [3]. There were 417 health security breaches in the year 2016 that were publicly disclosed. The breaches exposed healthcare records of more than 3.2 million people according to According to Privacy Right Clearinghouse [4]. In the United States alone in 2017, records of 5.579 million patients were affected by various data breaches [5]. Generally, in the case of security breach, electronic and other digital information related to the health of the patient is compromised. Hence, privacy and security are major concerns in electronic healthcare [6][7][8][9]. According to Cisco, 50 billion devices will be connected to the Internet by the year 2020 and this will outnumber the projected human population of 7.7 billion by five to one. This makes the security threats even higher [10] [11]. Fortunately, that projection was not accurate. However, there are more connected devices today as compared to a decade ago. Information Handling Services (IHS) Markit Ltd predicted that there will be 17.6 billion connected devices and managing users 'privacy is a major challenge for such number of connected devices [12]. Some of the solutions that have been put for the IoT enable e-Health Systems threat management and prevention are limited due to reasons such as less power availability to the devices, low memory, and low processing speed [13][14]. The threats related

to the IoT security includes eavesdropping, masquerading, routing attacks, denial of service, data modification, man-in-the-middle, user compromise, hard-ware compromise, software compromise, data replay, data fabrication, data privacy, and authentication attacks [15][16][17][18][19][20]. As a result of these attacks, the patient's private health related information is exposed to the public. This information can be used for insurance related frauds and can also cause the death of the patient if the patients' medical data is manipulated [15][17][21]. The IoT enabled e-Health Systems such as smart watches, arable electroencephalography (EEG) headsets and wireless electrocardiogram (ECG) necklaces are usually sensor based and communicate over a wireless network. [15] [17] [19] [22]. According to some researches, the IoT enabled e-Health systems are mostly used by the elderly, and chronically ill patients having diseases such as blood pressure and heart issues, among others [16] [17] [19] [22] [23]. “Systematic and structured security technique, used to identify the security objectives, threats and vulnerabilities of an
application, to help make design and engineering decisions, and determine where to prioritize efforts in designing, developing and deploying secure Applications” is called Threat Modeling [24]. For the determination of security and privacy threats at early stages, understanding the requirements to secure the system, and development of better security systems the threat modeling has proven to be very successful [25]. In this proposal, the team is proposing plug-ins that will help making the IoT enable e-Health Systems more secure and hence help achieve the goal of Security threat modeling for IoT enabled e-Health systems.

3. List of Proposed Team Members Names:

The names of the team members who will be working on the project, “Security threat modeling for IoT enabled e-Health systems” for the Sysc 4907 course under the supervision of Professor: “Mohamed Ibn Kahla” are as follows:

> Hamood Ismail

> Kamran Sagheir

> Tinsae Debele

> Yohannes Kussia

4. Objective

The objective of the proposal is to fulfil the requirements of Sysc 4907 course. The team is proposing a security threat model for IoT enabled e-Health systems by identifying threats and finding out ways to encounter the issues such as authentication, authorization control, and so on. To achieve the objective the team is planning to develop security tools as plugins to OM2M project. One of the plugins will enable the management and registration of devices and users. The plugin will achieve its objective by registering users of different roles, registering devices with different interfaces, and by creating a profile that suits each user. The tools will also have a plugin that will allow for the access and trust management. The objective of the plugin will be achieved by providing different users of the e-Health system, different access levels and based on the access level, providing them with different levels of trust. Another plugin will allow for the setting of security policies. The plugin will achieve its objective by facilitating the encryption and authentication. Therefore, based on the type of device and resource, switch between regular, heavy or lightweight encryption and so on will be made. These tools will create a library

that will be generic. The library will be capable of providing security features to IoT enabled e-Health systems that use OM2M project as their tool for communication.

5. Significance

The introduction of the tools containing the security plugins for the OM2M project will help solve the issues related to the security of IoT enabled e-Health systems. This will increase the trust of consumers on e-Health systems such that they will be able to utilize the e-Health services more confidently. These tools will make the records of the patients more secure and safe. This will decrease the chances of getting personal information leaked by the red hat hackers or cyber criminals. This will make the IoT enabled e-Health systems infrastructure more safe and secure. The registration of devices and the authentication of the devices will make sure that no unwanted and threatening device gets connected to the network. This will prevent the attempts by the cyber criminals to get access to the network and manage/edit the information and data by manipulating it and using it for their own purposes. Similarly, the encryption of the sent messages will make sure that even if someone get access to see the data going across the network, they won't be able to utilize it as the data will be encrypted. Decrypting this data will require a lot of resources and time and if the criminals manage to get access to the data, it will be too late for them to utilize it and the data will be redundant by that time. The tools will also provide a plugin that will facilitate the dynamic data access in a secure manner. This will prevent the unauthorized access to the secure and sensitive information. This shows the significate of the plugins to the OM2M project that will make the IoT enabled e-

Health systems more safe and secure. This will make the users of the e-Health systems are less vulnerable to the issues and threats associated with the IoT enabled systems. Hence, the users will be able to utilize the IoT enabled e-Health systems in a more safe and secure manner than before.

6. Design

The design of the security system will be based on the different nodes of the health system. Different nodes have different access levels and different level of sensitivity of information. Based on the nodes, we can have different security layers, management, access control strategies. The encryption and decryption algorithm will also differ between the different nodes. If the information at the node is sensitive and the impact of the information being known by an adversary is high, we prefer security over performance. If the sensitivity of the data passed at the node is less sensitive and the impact of the information being known by an adversary is low, performance can be chosen over security.

Based on this we can chose an encryption and decryption algorithm and length. Authentication will be multifactor authentication using password based and token based (a card to tap on the sensor). The system will have a service layer that includes device management, device and user enrollment and data management. We will use security design principles such as least privilege (only give access for a user to perform the task needed and revoke access afterwards), economy of mechanism (keep it simple), complete mediation (Every access must be checked against access control mechanism), separation of privilege (system should not grant permission

based on a single condition).With this principles and methods we can enforce the CIA triad which is confidentiality, Integrity and availability.

6.1 Security Layers

In the system we identify 7 security layers each having their characteristics. The first layer is the major asset being protected, this can be medical records, patient conversation and hospital transactions. This layer is known to be called mission critical assets, because the assets are high risk assets if adversaries have access. Thus, high security is needed to be implemented in this layer. This layer can be depicted as a data structure or database that has records in SQL or java.

The second layer is data security which deals with protecting the storage and transfer of data. This can be implemented as an access control where different users have different access of data. The data can be encrypted and decrypted to provide confidentiality.

The third layer is Application security which protect access to an application, an application's access to your mission critical assets, and the internal security of the application. Applications can be any software internal to the system or external to the system that can have access to a sensitive data. We can know where the app is sending a request from and only the applications that have access are registered in a database in where access is granted to only applications that are enrolled into the system.

The fourth layer is Endpoint security which protect the connection between devices and networks. Conversations between doctors and nurses or doctors and patients are one part of the connections that need to be secure. Another type of connection cam be data transfer from

a nurse to a doctor. A major asset can be passing through these channels, thus high security needs to be implemented. We will use encryption and decryption algorithms to provide confidentiality as well as use authentication algorithms to provide integrity and availability.

The fifth layer is network security which protects an organization's network and prevent access of unauthorized access of the network. The network is important because most communication these days is done through the internet. In the health system each worker has a tablet that they communicate with and transfer files from one to the other. This network connections are done through TCP where segments of data are sent. The data being sent over the network must be encrypted well to provide confidentiality; sequence numbers can show data is being received properly.

The other layer is permitter security which includes both the physical and digital method that protect the E-health system. The last layer is the human layer. Humans are known to be the weakest link in a security system. This layer includes access management controls to protect mission critical assets from human threats such as hackers, malicious insiders (nurse entering a negligent doctor's system) and negligent users (a user leaving account open for a while unattended, dropping access pass).

All data and channels do not require equal level of security. Identifying and categorizing the sensitivity of data and surrounding the security around the mission critical is the best way for the system to provide an efficient security based on performance and cost.

6.2 Security Nodes

This identifies the different entities and communication in the health system. Most entities and whom security may concern is Patients. Patients are the customers that are unknown in number and are a lot depending on the healthcare organization. They have the mission critical asset to be protected. The other entities identified in the system are Receptionists which are the first stop to most hospitals and clinics. They enter data from a user such as health card number, name, age, etc. These are confidential data that must be protected. The other entity is a nurse that takes this data and gathers more data from the patient. This data will then be sent to the fourth entity the Doctor.

We can identify many communication links between the entities. The pair of entities and communication pair is known as nodes. One node is the communication between a receptionist and a nurse. The other node identified is communication between a nurse with a doctor. A doctor can send data direct to a user which could be medical result. So another communication is between a doctor and a user.

We can see the level of sensitivity of data within the different nodes is different. We classify this sensitivity as low, moderate and high risk. Node 1 which indicates the communication between receptionist and nurse includes identity data such as (name, age, health-card number), this data if seen or altered by an adversary can have risk, classifying it as a moderate risk.

Nurse can have data gathered from a patient during interaction and a doctor can see the patient result. This node can either be moderate or low risk. With the assumption that a doctor usually asks the patient about symptoms and also checks the patient further himself/herself,

even if an adversary altered the record the patient can deny it an and the doctor recheck the patient's data such as heartbeat, blood pressure…etc. Thus, this node is identified as moderate/low risk.

The communication between a doctor with a user is a high-risk node. This is because patients tend to acknowledge everything their doctors say and if a doctor prescribes a medicine and gives a medical result to a patient the patient will get the medicine and believe the result. Imagine an adversary alters the result or prescription medicine or can read a patient's prescription. This would cause a very high risk in the patient's health and life, not only that but also on the integrity of the organization (Lawsuits, accusations, closure, etc.).

6.3 Security Management

6.3.1 Access control strategies

6.3.2 Security Policies

6.3.2.1 Encryption and Decryption mechanisms

6.3.2.2 Authentication Services

6.4 Service Layer

The system will have a service layer that includes device management, device and user enrollment and data management. We will use security design principles such as least privilege (only give access for a user to perform the task

needed and revoke access afterwards), economy of mechanism (keep it simple), complete mediation (Every access must be checked against access control mechanism), separation of privilege (system should not grant permission based on a single condition).With this principles and methods we can enforce the CIA triad which is confidentiality, Integrity and availability.

6.4.1 Device Management

To be able to have a secure system we will need to be able to manage the devices in the system and do it securely. Device management is the ability to identify threats and stop them as efficiently as possible.

First point about managing the registration of a new device, the system must make sure that the device doesn't have any harmful content in it for example device is carrying malware or some other threat against the system which could happen from a device.

Second is making sure the data on the device is secure. If the device got breached or had to go out of the hospital range a protocol would activate and wipe the data out for confidentiality reasons.

Third is making sure the communication between the devices, device and user, or device and database is safe and would be received by others in the system.

6.4.2 Data Management

This part will be about how much data is sensitivity, how to manage it, and how to leverage its assets for the hospitals gain.

By knowing the sensitivity of the data that is being saved in the system, it will enable the system to achieve the best performance to security ratio. For example, nonconfidential information would rather have the best performance than having security so the system would optimise itself for that situations.

If they information the system has is managed properly the system would be able to increase its speed, efficiency, and service. If the system knows how many patients are awaiting help, how many beds needed, how many are available, etc. it will be able to make the best route it can with the information given so if more information is given and managed in the best manner it would help a lot.

leveraging the hospitals assets is knowing what may be needed or what could help, for example, having information about set type of patients

7. Tools

To achieve a secure e-health system, the product must go through Designing, implementation and testing in several conditions and hardware. In this project we conduct the design, implementation and test our project using the following hardware and software's.

Eclipse oneM2M iot Environment: the e-health system industry relay on in different machine to machine solutions that involves extensive use of specifically customize hardware and software, which designed only for the industry's service. The objective of the oneM2M is to

minimize M2M service layer standard market fragmentation by consolidating currently isolated M2M service layer standards activates and a globe specification [26]. The commons service layer is intended to be embedded withing a various hardware and software components to ensure to M2M devices can communicated on a global scale.

Raspberry PI 4: Raspberry Pi is a series of small single board computer and high-performance computer, that runs Linux OP system. the most important part of the project is implementing the secure e-health system and pass it though test. To achieve the goal of creating a secure system, it important to implement and run the project. To achieve this, the project will be implemented using a Raspberry. The platform is implemented on two or more raspberry Pi 4 that will be connected using the oM2M environment. Each Raspberry Pi will be used apply

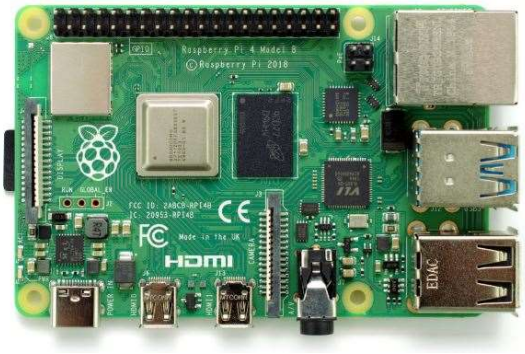

Figure 29: Raspberry pi

Java: Java is a class-based, object-oriented programming language. It will be used extensively to implements the e-health system. In addition, oneM2M is programmed using a Java. As result, in this project java is used to program the plug-ins and the security layer.

Database: In the project, PostgreSQL will be used to store any relevant information, such as hashmaped password and encrypted data. The main advantage of using PostgreSQL is compatibility with Java. It can be deployed scalable PostgreSQL with Cost-efficient and resizable hardware capacity.

Arduino: is an open-source hardware and software, which is a single board microcontrollers and microcontroller kits. In our project, it is used to read a data from card reader sensor and send it to Raspberry PI, which is responsible for processing the received data.

8. Expected Results

The e-health security system is modeled focuses on protecting computer systems and electronic data against unauthorized access, hardware theft, data manipulation, and against common threats and exposures such as backdoors, denial-of-service (DoS) attack and phishing [27]. Our e-health security system model is expected to provide the three security pillars namely, integrity, confidently and availability. Each component represents a fundamental objective of the E-health security system. The measures that will be applied is expected to attain protection of valuable data and system; securing system resource includes protection of a computer system hardware and software, whereas the data security will provide more concerned with protecting data that are stored or transmitted between computer systems, as well as cloud systems. In addition, the security system will be relived healthcare provider the rigor involved to manage infrastructure and provide them ample opportunity to familiarized

with IT service. Furthermore, the system will offer a numerous benefit ranging from scalability, cost effectiveness, agility enhancement of collaborative sharing of resources securely.

Achieving security and privacy of the patient in e-health is very a vital in achieving the objective of the wellbeing of the patient. As indicated at College of physicians and surgeon of Saskatchewan, “it is not possible to provide appropriate medical care to patients if the patients withhold relevant information out of a concern that the confidentiality will not be maintained” [28]. Any breach in confidently reduces the faith and trust of members of the public in the medical profession and in the confidently of health information. As result, it is expected that the system will safeguards the information that is gather and limits access of information for unauthorized accesses, which increase the wellbeing of patient and the public trust into the health system in return.

The academic and social experience in this project, it is expected that the project-based learning will provides the opportunist to collaborate, and teaches the skills of problem solving, critical thinking and time management. The project will give the opportunities to apply the application of skills and knowledges attained during their earlier years of our degree’s programs. It is expected that each student able to develop a new ability for applications to real-world problems and learn the through research-based projects.

9. Management and Timetable

As indicated on table below in order to handle the project efficiently and effectively, the workload is divided into a group and individually. In addition, the project is divided into milestone and Task. The milestone is more related to the software and designed, whereas the task is related to the submittable paper and research. To manage time, each Milestone and Task are given time based on the workload. As indicated on the table below, each project has starting date and deadline. Each millstone and Tasks are divided to sub-section, each sub-section is given time with the workload. This allows to have more clarity where the project must be and helps in the project to develop a better understanding of how things are unfolding. In addition, the members are set up to meet three time a week and one meeting with project supervisor and coordinator in a week.

Tasks		**Responsibility**	**Starting**	**Deadline**
Milestone Zero: Install and Run Eclipse oM2M:		**Individual**	**16-Sep**	**29-Sept**
	Install oM2m	Individual		
	Build and Run oM2M	Individual		
	Run Interface and Plug-in [research]	Individual		
Milestone One: Research and Design requirements:		**Individual/Group**	**20-Sept**	**25-Oct**
	Research E-heath Requirement	Individual	20-Sept-2020	29-Sept-2020
	Research Security Requirement of E-health	Individual	29-Sept-2020	07-Oct-2020
	Design E-health system security	Group	07-Oct-2020	25-Oct

Task 1: proposal Due:		**Group/ individual**	**01-Oct**	**02-Nov**
	Frist Draft copy	Individual	01-Oct-2020	10-Oct-2020
	Second Draft Copy	individual	20-Oct-20	25 – Oct-2020
	Final Proposal	Group	25-Oct	02-Nov
Task 2: Oral Presentation:		**Group/individual**	**15-Oct**	**11-Dec**
	Research and Preparation of key point	Individual	15-Oct	01-Dec
	Preparation of oral practices	Group	01-Dec	11-Dec
Milestone 2: Implementation of E-health system and Security		**Group**	**21-Dec**	**26-Feb-2021**
	Implementation of E-health system plug-in	Group	21-Dec-2020	25-Jan-2021
	Integrate the security future to oM2M	Group	25-Jan-2021	26-Feb-2021
Task 3: Progress Reports		**Group/ individual**	**01-Jan-2021**	**20-Jan-2021**
	Draft Progress Reports	Individual	01-Jan-2021	10-Jan-2021
	Final Progress Reports	Group	10-Jan-2021	20-jan-2021
Task 4: Draft of Final report		**Group**	**24-Jan-2021**	**26-Feb-2021**
	First Draft Final report		24-jan-2021	15-Feb-2021
	Review Draft Final report		15-Feb-2021	26-Feb-2021

Task 5: Animation Design:		**Group**	**01-Feb-2021**	**09-Apr-2021**
	Design animation [research]	Group	01-Feb-2021	15-Feb-2021
	Implement the Design	Group	15-Feb-2021	26-Feb-2021
Milestone 3: Test and implementation of E-health system:		**Group**	**26-Feb-2021**	**05-Mar -2021**
	Debugging and Testing	Group	26-Feb-2021	05-Mar -2021
Task 6: Oral Presentation		**Group/Individual**	**21-Feb-2021**	**12-Mar-2021**
	Preparation of presentation materials	Individual	21-Feb-2021	28-Feb-2021
	Oral Practices	Group	28-Feb-2021	12-Mar-2021
Task 7: Final Project Report		**Individual/Group**	**15-Mar-2021**	**09-Apr-2021**

Table 3: Timetable and break down of Responsibility

The Gantt chart is used to manage the time and the progress of the project. Gantt charts represent a critical information such as who is assigned to what, duration of task and overlapping activates project. As indicated on Figure 1, each subsection of milestone one is completed 100%, whereas proposal sub-section is completed only 82%, which indicated the progress of the projects. This allows individual or group responsibility to be managed properly and timely matter. It helps to manage and project teams to assess how long a project should take, determine the resource needed, understanding the dependences between task and milestone. Furthermore, it helps to visualization of prioritization tool as it provides the total overview of the project.

Figure 30: Gantt Chart Phase One

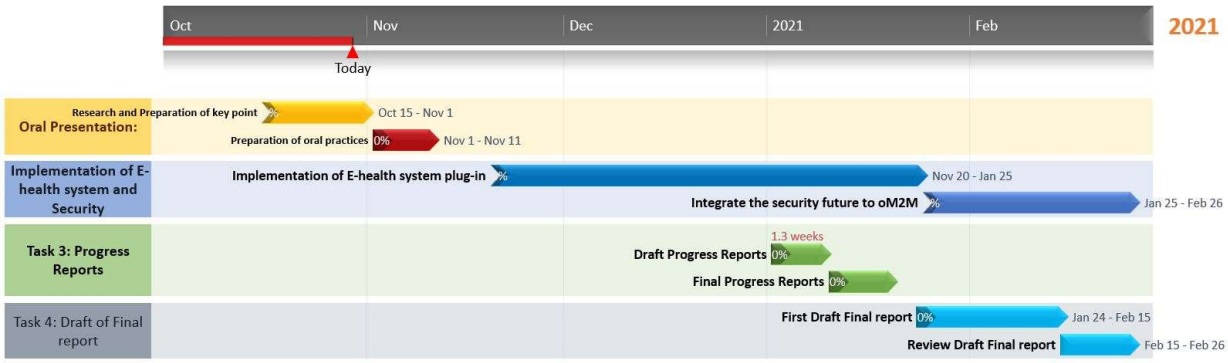

Figure 31: Gantt Chart Phase Two

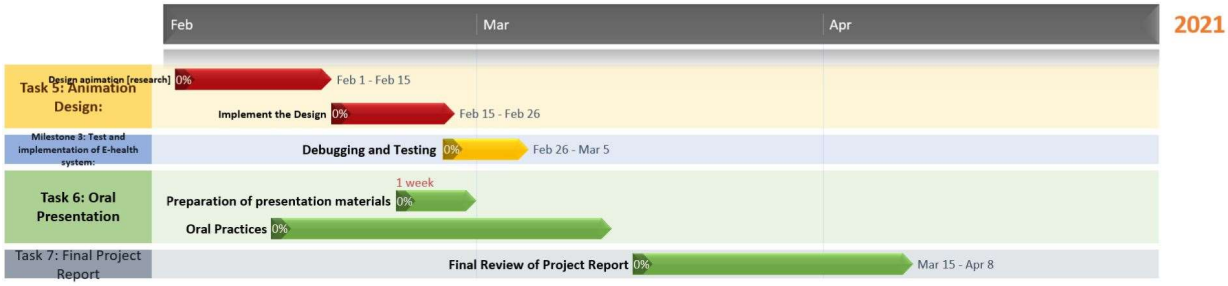

Figure 32: Gantt chart Phase Three

10. References

[1] Sabrina Sicari, Alessandra Rizzardi, Alberto Coen-Porisini, Luigi Alfredo Grieco, Thierry Monteil. Secure OM2M Service Platform. International Workshop on Self-Aware Internet of Things (SELFIOT 2015), Jul 2015, Grenoble, France. 6p. ffhal-01228303

[2] Adebayo Omotosho, Benjamin Ayemlo Haruna & Olayemi Mikail Olaniyi (2019) Threat Modeling of Internet of Things Health Devices, Journal of Applied Security Research, 14:1, 106-121, DOI: 10.1080/19361610.2019.1545278

[3] Islam, S. R., Kwak, D., Kabir, M. H., Hossain, M., & Kwak, K. S. (2015). The internet of things for health care: a comprehensive survey. IEEE Access,3, 678–708.

[4] Omotosho, A., Emuoyibofarhe, J., & Oke, A. O. (2017). Securing private keys in electronic health records using session-based hierarchical key encryption. Journal of Applied Security Research,12(4), 463–477. doi:10.1080/19361610.2017.1354272

Privacy Right Clearinghouse (2016). Data breaches. Retrieved from https://www.privacy-rights.org/data-breaches.

[5] Landi, H. (2018). 2017 Breach report: 477 breaches, 5.6M patient records affected. Retrieved

from https://www.healthcare-informatics.com/news-item/cybersecurity/2017-breach-report-477-breaches-56m-patient-records-affected 120 A. OMOTOSHO ET AL.

[6] Omotosho, A., & Emuoyibofarhe, J. (2014). A criticism of the current security, privacy and accountability issues in electronic health records'.International Journal of Applied Information Systems,7(8), 11–18. doi:10.5120/ijais14-451225

[7] Omotosho, A., & Emuoyibofarhe, J. (2014). A criticism of the current security, privacy and

accountability issues in electronic health records'.International Journal of Applied Information Systems,7(8), 11–18. doi:10.5120/ijais14-451225

[8] Omotosho, A., Emuoyibofarhe, J., & Meinel, C. (2017a). Ensuring patients'privacy in a cryptographic-based-ehr using bio-cryptography. International Journal of Electronic Healthcare,9(4), 227–254. doi:10.1504/IJEH.2017.085800.

[9] Omotosho, A., Emuoyibofarhe, J., & Meinel, C. (2017b). Securing E-prescription from medical identity theft using steganography and anti-phishing techniques. Journal of Applied Security Research,12(3), 447–461. doi:10.1080/19361610.2017.1315788

[10] Campbell, D. (2016). Internet of things security and privacy –part 1. Retrieved from http://www.macadamian.com/2016/11/15/internet-of-things-security-and-privacy-part-1/

[11] Evans, D. (2011). The internet of things how the next evolution of the internet is changing everything. Retrieved from https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[12] Nordrum, A. (2016). Popular internet of things forecast of 50 billion devices by 2020 is outdated. Retrieved from https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

[13] Abie, H., & Balasingham, I. (2012, February). Risk-based adaptive security for smart IoT in eHealth. In Proceedings of the 7th International Conference on Body Area Networks (pp.269–275). Brussels, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[14] Doukas, C., Maglogiannis, I., Koufi, V., Malamateniou, F., & Vassilacopoulos, G. (2012, November). Enabling data protection through PKI encryption in IoT m-Health devices. In 2012

IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE) (pp. 25–29). Larbaca, Cyprus: IEEE

[15] Agrawal, V. (2015). Security and privacy issues in wireless sensor networks for healthcare. In Internet of Things. User-Centric IoT (pp. 223–228). Cham, Switzerland: Springer

[16] Gonc¸alves, F., Macedo, J., Nicolau, M. J., & Santos, A. (2013, September). Security architecture for mobile e-health applications in medication control. In 2013 21st International Conference on Software, Telecommunications and Computer Networks (SoftCOM) (pp.1–8). Primosten, Croatia: IEEE.

[17] Gope, P., & Hwang, T. (2016). BSN-Care: A secure IoT-based modern healthcare system using body sensor network. IEEE Sensors Journal,16(5), 1368–1376. doi:10.1109/ JSEN.2015.2502401

[18] Islam, S. R., Kwak, D., Kabir, M. H., Hossain, M., & Kwak, K. S. (2015). The internet of things for health care: a comprehensive survey. IEEE Access,3, 678–708.

[19] Joshitta, M. S., & Arockiam, L. (2017). EPC based authentication of devices in the smart health system. International Journal of Electrical, Electronics & Computer Science Engineering,4(4), 6–11. Landi, H. (2018). 2017 Breach report: 477 breaches, 5.6M patient records affected. Retrieved from https://www.healthcare-informatics.com/news-item/cybersecurity/2017-breach-report-477-breaches-56m-patient-records-affected120 A. OMOTOSHO ET AL.

[20] Moosavi, S. R., Gia, T. N., Nigussie, E., Rahmani, A. M., Virtanen, S., Tenhunen, H., & Isoaho, J. (2016). End-to-end security scheme for mobility enabled healthcare Internet of Things. Future Generation Computer Systems,64,108–124. doi:10.1016/j.future.2016.02.020

[21] Sivaraman, V., Gharakheili, H. H., Vishwanath, A., Boreli, R., & Mehani, O. (2015, October). Network-level security and privacy control for smart-home IoT devices. In 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob) (pp. 163–167). IEEE, Abu Dhabi, United Arab Emirates

[22] Savola, R. M., Savolainen, P., Evesti, A., Abie, H., & Sihvonen, M. (2015, August). Risk-driven security metrics development for an e-health IoT application. In Information Security for South Africa (ISSA), 2015 (pp. 1–6). Johannesburg, South Africa: IEEE.

[23] Zhang, X. M., & Zhang, N. (2011, May). An open, secure and flexible platform based on internet of things and cloud computing for ambient aiding living and telemedicine. In 2011 International Conference on Computer and Management (CAMAN) (pp. 1–4). IEEE, Wuhan, China.

[24] Jagannathan, V. (2018). Threat modeling: Architecting & designing with security in mind (pp. 1–33). Retrieved from https://www.owasp.org/images/a/a6/AdvancedThreatModeling.pdf

[25] Shostack, A. (2014). Threat modeling: Designing for security. Indianapolis, IN: John Wiley &Sons.

[26] *oM2M*. (2017). Eclipse. https://www.eclipse.org/om2m/

[27] Yazan Al-Issa, Mohammad Ashraf Ottom , and Ahmed Tamrawi. (2019). *EHealth Cloud Security Challenges: A Survey*, 2. https://doi.org/10.1155/2019/7516035

[28] C.P.S.S. (2014, November). *Confidentiality of patient Information*. Https://Www.Cps.Sk.Ca.https://www.cps.sk.ca/imis/CPSS/Legislation__ByLaws__Policies_and_

Guidelines/Legislation_Content/Policies_and_Guidelines_Content/Confidentiality_of_Patient_Information.aspx