Design of a Backend System to Integrate Health Information Systems – Case study: Ministry of Health and Social Services (MoHSS)-Namibia

by
ANNA- LIISA SHOOPALA

This Dissertation is submitted to the Department of Electrical Engineering, in partial fulfilment of the academic requirements
for the Degree of

Master of Engineering in Telecommunications

Supervisor: Dr Joyce Mwangama

Faculty of Engineering and The Built Environment
UNIVERSITY OF CAPE TOWN

February 2020

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Engineering in Telecommunications Engineering in the University of Cape Town. It has not been submitted before for any degree or examination at any other university.

I know the meaning of plagiarism and declare that all the work in the document, safe for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signature of Author: Signed by candidate

Cape Town

14 February 2020

Abstract

Information systems are the key to institution organization and decision making. In the health care field, there is a lot of data flow, from the patient demographic information (through the electronic medical records), the patient's medication dispersal methods called pharmaceutical data, laboratory data to hospital organization information such bed allocation. Healthcare information system is a system that manages, store, transmit and display healthcare data.

Most of the healthcare data in Namibia are unstructured, there is a heterogeneous environment in which different health information systems are distributed in different departments [1][2]. A lot of data is generated but never used in decision-making due to the fragmentation. The integration of these systems would create a flood of big data into a centralized database. With information technology and new generation networks becoming a called for innovations in every day's operations, the adaptations of accessing big data through information applications and systems in an integrated way will facilitate the performances of practical work in health care.

The aim of this dissertation is to find a way in which these vertical Health Information System can be integrated into a unified system. A prototype of a back-end system is used to illustrate how the present healthcare systems that are in place with the Ministry of Health and Social Service facilities in Namibia, can be integrated to promote a more unified system usage. The system uses other prototypes of subsystems that represent the current systems to illustrate how they operate and in the end, how the integration can improve service delivery in the ministry.

The proposed system is expected to benefit the ministry in its daily operations as it enables instant authorized access to data without passing through middlemen. It will improve and preserve data integrity by eliminating multiple handling of data through a single data admission point. With one entry point to the systems, manual work will be reduced hence also reducing cost. Generally, it will ensure efficiency and then increase the quality of service provided.

Nomenclature

API	Application Integration Interface
ARV	Anti-Retroviral Drugs
CLI	Command Line Interface
CSS	Cascading Style Sheet
DHIS	District Health Information System
EDT	Electronic Dispensing Tool
EFC	Entity Framework Core
EMR	Electronic Medical Record
GUI	Graphical User Interface
FESC	Facility Electronic Stock Card
HATEOAS	Hypermedia as the Engine of Application State
HIS-	Health Information System
HIV	Human Immunodeficiency Virus
HTTP	Hyper Text Transfer Protocol
ICT	Information Communication Technology
IIS	Internet Information Services
IP	Internet Protocol
MVC	Model View Controller
MoHSS	Ministry of Health and Social Services
TCP	Transmission Control Protocol
REST	Representational State Transfer
SIAPS	Improved Access to Pharmaceuticals and Services
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
IRL	Uniform Resource Locator
URN	Uniform Resource Name
WHO	The Word Health Organization

Table of contents

List of Figures:

CHAPTER 1

Introduction

Within health departments, the acquisition of health information is done through day to day data capturing, analysis and health reports. Most of the cases are originated from various sources and different vertical and independent health information systems that are used. The vertical programs are independent because they are most likely to have originated from numerous donors who often designate their explicit instructions and requirements on how their program should be handled [1]. Due to the number of funds these organizations pour into these programs and the necessities to the developing communities, countries like Namibia have very few options hence ending up accepting the donation[2].

The Namibian public health falls under the supervision and management of the Ministry of Health and Social Services (MoHSS).It is classified with around 30 district hospitals including 3 intermediate hospitals, 286 clinics and 37 health care centres facilities, this data is pinpointed to focus primarily on the dominating public healthcare [3]. Figure 1 summarises all the types and ownership of the health facilities in Namibia in general.

Facilities By Type And Ownership

Types	Faith Based Organisation	MOD	NGO	Police	Private	Public_MoHSS	Unsure
Hospital	7	1	0	0	8	27	0
Health Centre	6	0	7	0	10	37	1
Clinic	14	0	9	2	80	286	0
Regional Health Office	0	0	0	0	0	14	0
District Health Office	0	0	0	0	0	38	0
Intermediate Hospital	0	0	0	0	0	3	0
Others	1	0	0	9	2	25	0

Figure 1: Health facilities in Namibia. [3]

The study conducted in [2], indicate that the MoHSS through its above-listed heath facilities in Figure 1 has manual medical records as well as electronic information systems located in the mentioned health facilities. The MoHSS's Directorate of Health Information Systems and Research through its three intermediate hospitals is the primary coordinator of the three major health information systems that the hospitals make use. That is the DHIS2, a system that collects and process all health data, the Pharmaceutical Management Information System that was initiated through the Improved Access to Pharmaceuticals and Services (SIAPS) program and the eHealth initiation that consist of the Electronic Medical Record system and other independent small systems within the hospital departments[4]. The DHIS2 is an open source system accessible for deployment by any interested organisation, the Pharmaceutical Management Information System is donor based and eHealth is a national initiation. The information systems in place are not only a result of multiple donors to the government but also the ministry's own initiations.

1.1 Problem Statement

The systems received from donors has created a problem around health information systems (HIS) in most developing countries (Namibia not excluded) and it is known to have caused the HIS to be incomplete and disoriented, mostly fragmented. [5]. Due to the fragmentations, the health facilities operating with information from different sources and databases ends up using different filling and retrieval systems for the same type of operations. The information obtained can be inconsistent and contradictory[4].

Departments in the MoHSS have autonomous information systems that are managed independently and run-on different software. As a result, the ministry ends up with uncoordinated silo type systems that cannot inter-operate. This created a conflict with the main objective of the mission of MoHSS which is: ‘‘To provide integrated, affordable, accessible and equitable, quality health and social welfare services that are responsive to the needs of the population. ’As stated in its 2017/2018 strategic plan.[6]

The following subjects have been observed, caused by the disintegrated systems in the Namibian public healthcare:

- Duplication of functionalities. More than one data entry for the same type of data as they are required to be entered for every independent system.
- Information is allocated in various and heterogenous systems distributed departmentally.
- There is no collective use of collected data.
- No continual observation and channelization of all attained data collectively.
- No mechanism for proper storage for collective healthcare data.

It is becoming a serious challenge for healthcare to make sense of the data it contains. The integration of these systems would create a flood of big data into a centralized database that can be useful for collective usage. With information communication technologies evolving fast and becoming a call for infrastructures in everyday operations, the adaptations of accessing big data and making use of it efficiently through its applications and systems will facilitate the performances of practical work in health care. The question is, could an integrated system with a centralized database and a collective usage of the generated data tackle the above-mentioned observations and many more?

1.2 Objectives and Sub-objectives

In a bid to provide technical solutions to the healthcare industry in Namibia, with a purpose to empower health workers and healthcare decision-makers, the objectives of this research are to assess the level of integration for the fragmented health information systems into a unified platform.

To achieve this, the following sub-objectives should be met.

- Assess the different information system that is used in the MoHSS hospitals, narrowing down to the 3 intermediate hospitals.
- Identifying information needs as well as the users’ needs.
- Study the system integration process and its requirements.
- Identifying and proposing the appropriate approach for the systems in need that can meet the need of the users and the public in general.

The sub-objectives serve as a pathway to achieve the following thesis objectives:

- Design sub-systems from different departments that inter-operate in their day to day processes as prototypes health systems.
- Model and implement a mechanism to combine the sub-systems into an integrated system.
- Evaluate and test the integrating system to validate the connectivity between the two sub-systems in a proposition of allowing proper information exchange and collective data storage.

1.3 Thesis Scope.

The scope of the dissertation involves a thorough review of the systems current in use at the health facilities of the MoHSS. The three main systems are studied on their operations; what they contribute, the problem they solve and their functionalities. The methods of integration are also assessed to find out the best way to come up with a prototype that can achieve the integration of the information system structured to the Namibia health systems. The dissertation further defines the involvement of API's (Application Programming Interface) in the integration process.

After the analysis of the existing facts, the system implementation is elaborated, this includes the design structure definition and implementation, documenting the design implementation and results. After that, follows the evaluation and test of the design to validate the proof of concept. The scope further continues to the analysis of the obtained results and recommendations, future work and conclusion.

Limitations

There are strict restrictions on allowing outsiders to have access to the information due to data integrity and confidentiality. It is therefore impossible to observe the case study systems thoroughly thus only a sneak peek of the systems was given to the researcher.

The demographic location of the health facilities in Namibia limited the observations to be made to only reachable facilities.

1.4 Document Outline

- Chapter 1, An introductory chapter that defines the research topic, explicitly describes the reason behind and factors that motivated the author to engage in the research within the health information systems (HIS) field. It also outlines the purpose and the significance of the research through the problem statement and highlighting the major objectives of the study.

- Chapter 2 consists of a detailed summary of the HIS present in the MoHSS. Detailed information is given on the description, the benefits and the contribution of the health information systems in place. The chapter also introduces the factors involved in the process of integration. It further tackles the technical part of the solution, explaining the technologies, methods and protocols involved Web applications, designing Frameworks and API technologies,

- Chapter 3 gives an outline for research progression by introducing the procedural overview and the research system requirements. It also outlines the design and implementation methodology. It later continue with the design of the integrated system and its overall system

architecture. The designs of the prototypes needed in the integration demonstration are given as well as the API design.

- Chapter 5 consists of a process on how the integration is achieved; First a description of the environmental setup and the project structure and later explanations of the technologies behind the implementation of the system. This also includes defining the architecture, finding the resources and the characteristics to be included in the concept of the system implemented.

- Chapter 6 contains the evaluation and testing of the designs and the presentation of the results.

- Chapter 7 conclude the thesis with details on the discussion of results, conclusion and recommendations.

2 Literature Review

This chapter gives an assessment of the technologies, methods, and protocols on which the implementation is built. First, a review of the system to be prototyped is presented to give an understanding of what the prototype requirements are. Detailed information is given on the description, the benefits, the contribution of integration to HIS and factors involved in the process of integration. It further tackles the technical part of the solution, explaining the factors involved in the development front and back end design of the web applications. This involves technologies such as HTML, CSS, JavaScript, HTTPs protocol; it as well presents the framework ASP.NET used for development and Database models.

2.1 Related Work

System integration is not a new but still a hot topic within the Information Technology world. It has been playing a role in different fields that involves information systems. In the health departments, it has been touched within different systems and diverse approaches to meet customized needs.

There seem to be a common problem of defragmented health information systems in developing countries, the integration of health information systems have been attracting the attention of many researchers across nations. In [1] the author tackled the challenges being experienced to exercise the integration process of the health information systems. They have centred their research with case study of Zanzibar, Tanzania. The District Health Information System DHIS is used in their health care facilities. To integrate the already existing HIS with the new DHIS, it was readjusted to accommodate the old datasets. DHIS is adapted in a lot of African Countries because it is open source, and it can be customised. This makes it easy to integrate with other HIS.[1]

While in [7] the authors proposed a networked healthcare system using cloudlets infrastructures, in which they reviewed how to collect and analyse health big data. They also focused on investigating ways in which mobile devices can be useful in integrating platforms in the health care industry. They suggested this to be achieved by collecting health data from Electronics health records applications such as Health Kit, Apple health, Samsung Health App etc.

There is a couple of methods and customised platforms that enable the integration of systems. Authors [8][9] have researched on a health information systems integration approaches and aspects to have in considerations when working on system integration. Having all this approaches still require one to select or come up with the most appropriate method that suit the need of the systems.

In [1][2] studies have been made on the status of health information systems in the Namibian hospitals, pinning it down to Central and Katutura Hospital in Windhoek and Oshakati Intermediate Hospital in Oshakati. They investigated the strategic ways on how to integrate the Health information systems in Namibia as well as the challenges that may arise. In doing so, applications to be realistic need to be customised based on the economic factors and geographical location, to meet the specific needs of the targeted populations.

2.2 A Review on Systems to be integrated

Gathering information and data collection process happens in the everyday operations. The execution of projects in the research facilities and health education, delivering high quality service, finding cures

and treatments in the healthcare industry, planning and decision making heavily relies on available and reliable information. [10]

The Word Health Organization (WHO), defines the Health Information Systems as a system that manages, accumulates and process data that are involved in the healthcare industry service delivery. [10] When information systems are discussed, the attention often goes to digital systems. In [2], authors argued that information systems can be electronic-based (paperless) or paper-based and they existed long before the electronic era. For a system to be formed up, there is a need to be an integration of different data collection methods, with all types of data from different departments to perform specific tasks.[2]

The MoHSS is fighting within its capacity to improve the quality of healthcare service delivery through its standard HIS and ICT involvement in the Namibian healthcare industry. Integrating ICT services have been an ongoing process which involves a couple of completed and ongoing projects throughout the globe[6]. The following reviews the HIS available and in place within the MoHSS limited to the three major and intermediate hospitals, Windhoek Central, Katutura and Oshakati hospital.

2.2.1 E-health

E-health is a very general and broad term. According to [11], the definition of e-health is pinned down to the connection between medicine, information and technology. Just like other sectors such as; e-commerce or e-business; e-health is a combination of medical informatics and digital health. This can be articulated as health services delivery (public health, health information, and business) improved with the use of the internet and related technologies. Figure 2 give a presentation of a general e-health system as a collective use of health information.

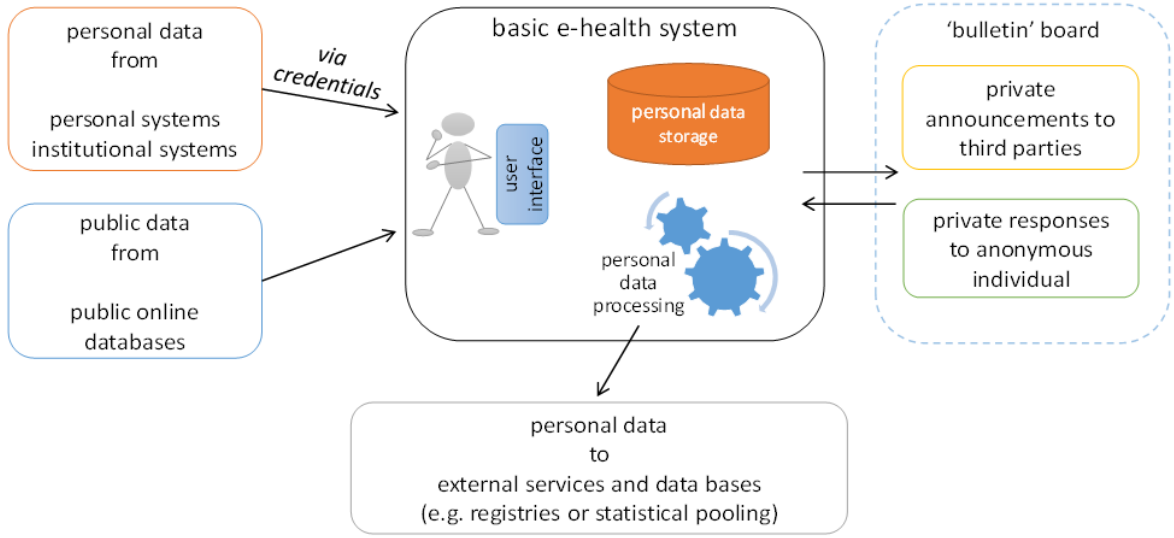


Figure 2: An illustration of an E-health system

The aim of e-health should be to increase efficiency and improve the quality of service in medical services delivery. E-health also plays an important role in transforming healthcare and making it smarter.[11]

2.2.1.1 E-health in Namibia

The E-health in Namibia is implemented through different proposed and implemented platforms. The most common one readily available and in usage is a customized Electronic Medical Record (EMR). With the custodian of the MoHSS, the EMR system known as only the eHealth system is planned to operate soon in all public hospitals in the country, currently it is only implemented in the 3 intermediate hospitals. The two hospitals (Katutura Hospital and Central Hospital) located in Windhoek with their database and the Oshakati Intermediate Hospital in the northern region of the county has its own database as well [4].

The system was first proposed to be just an initiation whereby the ministry issued the replacement of the booklets Health passports by the electronic medical card. The card consists of the patient identification details and operates as an access point to the electronic system [12].

It is a web-based application and covers most of day to day patient handling hospital operations, operated through the networked system, that connects the two hospitals in Windhoek, each with its own individual database.[13]**.**

Key Features[12]

- Medical record (medical history, diagnosis, and prescriptions) of the patient stored in digital format.
- Issue billing statements.
- Admission and discharge.
- Process data and generate reports to feed the DHIS.

2.2.2 Pharmaceutical Management Information System

The development of a Pharmaceutical Management Information System was initiated through the ministry's "treatment for all" campaign. The campaign was initiated to control the spread of HIV through the dispersal of antiretroviral (ARV) drugs for infected people. The ministry however saw the need to include all other general medicine dispersal and other integrated pharmaceutical needs and information management [14].

The ministry developed the system through the help of the USAID-funded Systems for Improved Access to Pharmaceuticals and Services (SIAPS) and it is currently in use in over 50 health points in the country. The implementation of this system has drastically improved the pharmaceutical services in the public hospitals as it has simplified the procurement and data management duties which enabled the workers to pay more attention to the patients and deliver good customer services[13].

In general, a Pharmaceutical Management Information System can be defined as a platform that manages the gears of the pharmaceutical processes. That includes the data collection, the procurement management, pharmaceutical resources, and medicine inventory, inpatient and outpatient medicine distribution management as well as data processing and presentation of information for the decision-making process[15].

1. Electronic Dispensing Tool (EDT)

This works as a drug distribution and inventory management tool. This software operates at the patient interaction points whereby the pharmacists are required to enter data into the system through a desktop application[14] The software has capabilities of recording patients' information (age, gender,

contacts, etc) and keep a record of either new, regular patient and follow-ups, distribution history, appointments for regular patients, stock taking, invoices and report generation. Figure 3 bellow indicates an example of report from the EDT tool.

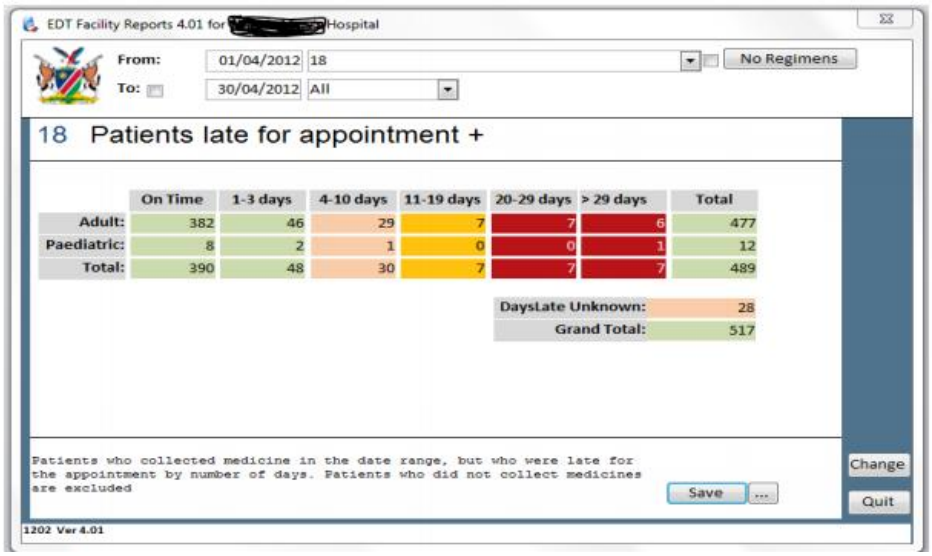


Figure 3: Patient record of the Antiretroviral drug data for a certain hospital in Namibia given, using the EDT system. [16]

Apart from a desktop application tool, the EDT extends to a mobile dispensing record tool. It is used for an outreach programme and off-site services. This enabled using a remote laptop connection through a national cellular 3G network modems, provided by a local network providing telecommunication company Mobile Telecommunications (MTC).[16] It also uses a mobile scanner among others to scan the distributed drugs for inventory purposes. Figure 4 shows a picture of such a scanner.

Figure 4: The mobile scanner. [16]

2. EDT National Database

The database enables the storage from the EDTs, the movement of data between facilities and the organization of information collected at the central point and the national level[14]. It was initialized in 2007, to put together data from all facilities to enable proper planning, management, and decision making. This data transferring process is enabled by the MTC's 3G networks after other attempts of using commutable USB devices (was impractical due to the devices getting lost and corrupted before reaching the destination) and another attempt to use General Packet Radio Service (GPRS) modem that turned out to be more expensive due to lack of telephone lines in all the facilities.[15]

3. Facility Electronic Stock Card (FESC).

The Facility Electronic Stock Card which is supported by the SIAPS program of the USAID is an e-stock card used by the pharmacists to replace manual stock-taking [13]. It is simply a card that is used to manage stock taking via an inventory management software (Softworks) installed in the pharmacy desktops. Its duties are to generate reports of the procurements and issuing of the stock and forward to the national dashboard

4. Pharmaceutical Management Information Dashboard

The dashboard is a web application system used by the MoHSS to manage information from the EDT National Database. Its purposes are to serve as tool to[14]:

- Collect all regional information, analyse and distribute the aggregated processed information for pharmaceutical services performance evaluation.
- Enhance the MOHSS visibility of the national level operations.
- Allow statistic evaluation.
- Enable proper planning and decision making.

The usage of the system is at the ministerial level and other committees use the information, such as the National Planning Commission and /or the WHO.

2.2.3 DHIS2

District Health Information System (DHIS2) is a Health Management Information System (HMIS) open source software instrument that is used for data collection, to authenticate the collected data, to do a thorough examination (validate and authenticate the data to ensure the quality, e.g. filter for unwanted data or repeating information), and to generate results through data presentation and interpretation.[17].

The DHIS2 evolved from DHIS which is a result of a study that was implemented by students and staff of the University of Oslo in 1994, through their HIS program. After several alterations created, new versions was created over time from a system that was initially created as a centralized database with features to give reports from surrounding health facilities [18]

It is a generic tool, compared to many trademarked health information systems. With DHIS2, the implementers rather define their customized systems that meet their needs without having to be an expert in programming [18]. DHIS2 is an integrated open source web application package. Figure 5, illustrate how the DHIS2 gather data from data capturers, gets data from other health information systems and data capturing devices to gather information which they process. The system processes the data for presentation and decision making.

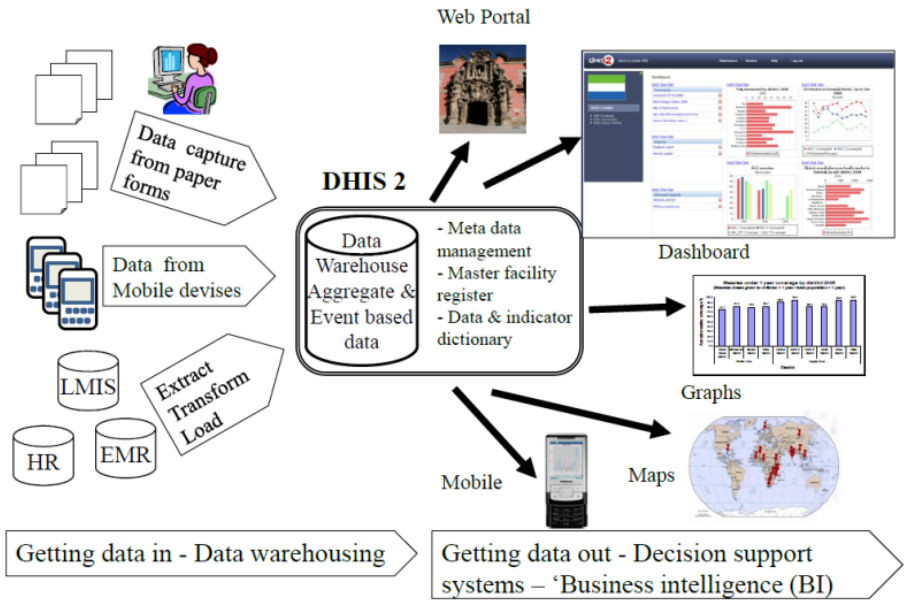


Figure 5: An illustration of the DHIS2 overview. [19]

2.2.3.1 The significant features of DHIS2 can be summarised as follows [17][19]:

- It can be customized by the end-users to match their needs (organizational, regional and operational needs) via the user interface. This setting can be done without any programming experiences.
- DHIS2 can accept different data formats.
- Reports can be generated through a one-click operation with an option of selecting what format to generate. The system allows other outside report making tools and design applications to access raw classified data for proper data presentation that it cannot support.
- The system contains an analytic module that enhances the quality of the recorded data by developing it and generate the data presentation formats such as tables, reports, etc.
- The customization of the application allows users to design their dashboards as per their preferences.
- It provides a flexible additional set up, adding more facilities, managing additional data groups and users via the end-user interfaces.
- The system flexibility contributes as a powerful feature to the DHIS2 system that enables the overall connection between the DHIS2 software and the external software entities such as through API.

2.3 Systems Integration Approaches

Relating to Health Information Systems, integration refers to the act of bringing data and systems together from disparate sub-systems and allowing communication between the sub-systems. The integration involves technical components, applications and infrastructure of the existing systems into a unified platform. It requires all systems to share data electronically.[20]

Why integrate systems?

An integrated system will in overall improve the quality of service delivery in the MoHSS through the following:

- Permits application to share data instantly without third parties.
- Advances data integrity by eliminating multiple manual data handling.
- Single data admission point reduces labour cost.
- Allows patients to keep a more reliable and comprehensive record.
- Simplifies data tracking which is useful in future planning.
- Stimulate overall efficiency of all the health information processes.

2.3.1 HIS Integration Requirements

For proper integration, an organisation evaluation needs to be conducted. It should include establishing a study of the present data structures and systems. This will conclude the reason for integration and the problem it is solving. When conducting system integration, the following factors must be considered.

Interoperability

It is defined as the ability of the business and information systems to be able to inter-exchange data and cooperate with each other[21]. In[4] author argued that without interoperability there is no integration,, integration start off with interoperability between systems, that means allowing technologies to talk to each other and enabling the flow of information.

Flexibility and scalability
Something that is flexible must able to adapt to all conditions presented to it. A proper integration is developing an integrated system whereby data can be accessed and sent at whatever speed. The system is flexible when it is able to accommodate different capacity, speed, and volume of data and be able to adapt to the change in such.

Flexibility goes hand in hand with the elasticity of the system too. A flexible system should also be able to be extended and/or constrained by scaling it up or down depending on the demand[22].

Connectivity
Connectivity is the key to all communication. Without connection, systems will not be able to communicate, in order to integrate decentralised systems, there is need to be a reliable connection between their locations.[22]

When independent HIS are integrated, a big pool of data is expected. Good connectivity will enable the proper handling of data through networked facilities. The few selected hospitals are the only ones with access to 4G networks. Many of the health care facilities do not have the network and computing capacity to handle big data characteristics.[4].

Data security and privacy

Allowing the usage of health data while preserving security, privacy and data integrity is a very difficult task. Before system integration, every department has its own system and there is less data handling. It is however considered that, data security concerns are more at the level of the personnel handling the data then the system itself. The threat that data security faces when integration is taking place therefore inappropriate access to data should be considered [22].

2.3.2 Application Integration

In most independent or standalone, HIS have their own databases, the data collected and stored in the databases per system, this database does not communicate with one another. This causes cases of fragmented data collection.

The interoperability between applications is achieved over the exchange of information between the systems via their software applications and database systems. This is known as Application Integration. [4] Figure 6 illustrate the connectivity of different HIS that can be accessed on one device in one instance. This is achieved through the Application integration level platform and the use of APIs.

Figure 6 A representation of HIS integrated through an application platform.[23]

To achieve this application-oriented solution, an overlaid centrally managed application or user interface is placed above all the existing interfaces and databases. There should also be a support in the movement of data streams and exchanging of information, thus the integration of processes, combination of control logics and merging of servers and databases should be achieved first. It is essential to define and understand the application and processes in the organization to provide application-oriented solutions via Application Programming Interfaces (APIs).[8]

Challenges of application integration

- Compatibility between systems – systems being vendor based,
- Communication protocols

- Data encoding formats
- Conflicts between transport and application layers
- Messaging protocol

2.4 Web Application Development

Web-based applications are programs developed to provide a service (or services) from a server to a client over the web. The client may use a remote server or host on the same environment. The remote server is accessed via a network [24]. Hypertext Transfer Protocol (HTTP) is of importance in a web application development as it is a protocol used to transfer data over the internet from the server to the client. Web applications are accessed through web browsers or web surfers such as Google Chrome, Opera, Firefox, Internet Explorer, and many others. A browser uses web pages to exchange data via hypertext links.[24]

A web application is categorized as a client-server application where by a browser is a client and a web server as the server, though the server can also operate as a client to other back end servers or databases. Running a web application is straightforward; it does not require some skill. It can perhaps be compared to running other types of applications. In that way, using a web application is the most basic thing as no configurations or setting up are required[25].

Developing it can be a headache if it's not planned properly. The web application architecture can be divided between the back end and front-end development. The back end is the coordination between data storage and management make, as well as the business logic of the application (refer to the illustration in Figure 7). The user interface and presentation of the logic is part of the front-end design. There is no restriction to which browser to use as long as the hypertext link, also called the Web link is known to the client. The web link is the gateway to the data on the server. The main idea behind the web application is the utilisation of collaborative services between the application users via the server[25].

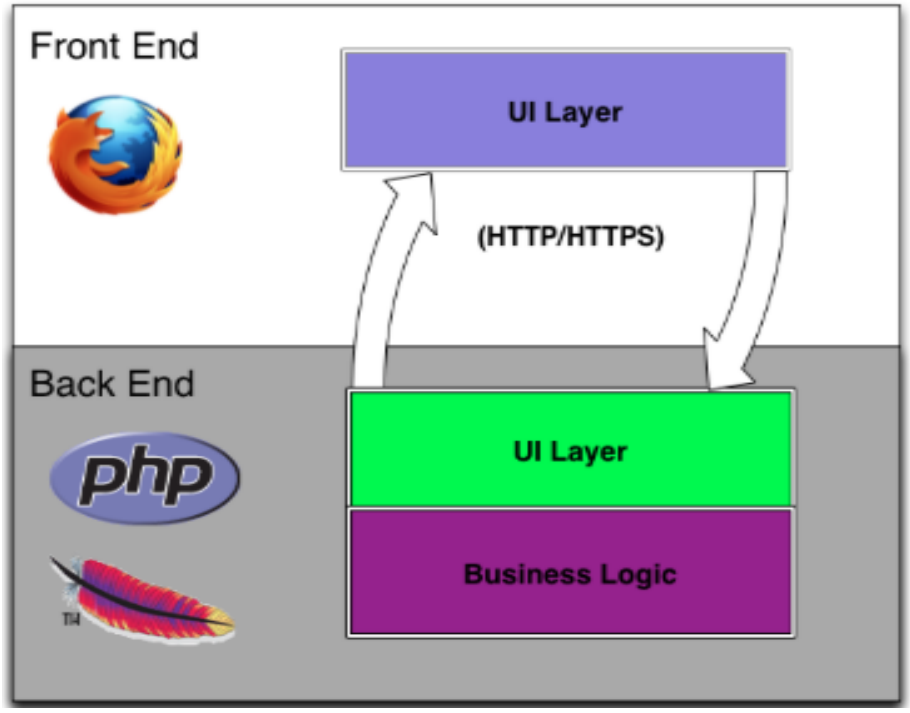


Figure 7 Interface layers in a typical Web applications architecture.[25].

2.4.1 Front-end Design

Front-end design is the creation of the user interface, it's through this interface that users interact with the business logic. The important consideration in designing this is that, it should be presentable, user-friendly and fall within the business requirements and prerequisites. There are a lot of tools being used in the front-end design. Traditionally and still popular for the front-end design, includes the usage of technologies such as HTML, CSS, and JavaScript to create a presentable web page[25].

2.4.1.1 HTML

Hypertext Markup Language is a computer language with syntaxes and codes structure that can be interpreted by web browsers. HTML is used in the front-end design to generate website interfaces. It is a string of codes normalized as tags, written by the web page authors in the HTML text editors saved as HTML files. The tag declaration starts with the tag name, the content of the tag/body and ends with the tag name. An example of a tag declared to define the title of the web documents can be as: *<tag title> Web title</tag title>*[26].

Apart from the title tags, other tags are used to declare the beginning of the document(the main HTML tag, every document start with this), the document head, the body content of the page, the hyperlinks, the end of the page, etc[27]

The HTML language structure consists of the tag type, enables the web page designers to publish web contents on the websites online and fetch data and contents of pages through the hypertext links again online[26].

The HTML elements are used to define the properties of the web structures and the authors' specifications on the comportment of the site. The elements structure is declared with the initial tag, the body, and the concluding tag[25].

2.4.1.2 CSS

Cascading Style Sheets (CSS) is the styling method used in the front-end design, to arrange the HTML elements. It takes care of the style, visuals, layouts, colour, etc all in one how the document's presentation should be defined.

2.4.1.3 HTTP Protocol

HTTP (Hypertext Transfer Protocol) is the fundamental protocol behind web application implementations. It allows communications over the network by defining the format and transmission method of communication messages. Since the main idea behind a web application is to exchange information, the HTTP, involvement in the web application designs is of great importance.[25]

HTTP is a protocol used in the application layer of the TCP/IP internet paradigm because it operates at the user interface level which is responsible for the presentation of information as well as interaction with the user. This all happens at the application layer and HTTP plays a big role in the interaction of the client through the web browser with the web server, hence it is an application layer protocol. Regardless of HTTP being an application layer protocol, is very much linked to the transportation layer [26].

Main features:

- HTTP operates in a request/response method, the client send a request following a format and the server respond to the request in a response format. [24]
- It operates without a continuous state or connection, meaning it does not maintain a state but rather the client makes a request and then waits for the response from the server[26].
- It's a resource independent protocol, meaning the resource in transportation is not adhered to by a certain format; it can be anything like a file, a script or a document.

2.4.1.4 HTTP Request and Response methods

All the HTTP requests are uplink messages that are sent to the server, they follow a format with 3 components, first the request method line, the header, and the body. This request and response methods the main highlights in the server-client communications in the web applications prototypes and the APIs used in the demonstration in Chapter 5. The lines are structured as below [25]:

METHOD /address HTTP/number} Method

information about the client} header

……

body: optional depending on the method type]} body

The request method instruction is a single word that instructs the server on what to perform. The header consists of further details of the request, it includes the information about the web browser or the type of networks that are used. The body is not always used, it is mostly only used when the client wants to send data to the server[26].

When a server receives a request method, it interprets it accordingly and returns a response message to the client. The response message is a status line that indicates the version number of the HTTP protocol in use, one or more header name(s) that indicate further information that the server wishes to communicate to the client and the body. The body is optional and contains details that are defined in the header if necessary. The response message is normally structured as bellow [25]:

HTTP-version, status-code eg: HTTP 2.0 101 OK } status line

Header name + value eg: HTTP

…….

[optional response body depending on the request format]

The used HTTP request methods are explained as follow [25][26] [28]**:**

1. GET- the GET method is the frequently used and simplest request method. It is used to fetch data from the server via a specified method. The GET method serves to acquire information from a specified server assuming that a Uniform Resource Identifier (URI) is provided. The URI's purpose is to identify the name of a file with the Uniform Resource Name (URN) and at the same time allow access location of such file with the specified Uniform Resource Locator (URL). The GET has no other effect on a file than to locate and return requested data. The script below represents an example of a get request that is used to fetch a file from a server. The script example almost resembles other request and response methods with common changes only appearing in the request method type.

GET / file/invitation. Text/html HTTP/2.0

User-Agent Chrome/51.0.2704.84
Host: www.events.com
Connection: Keep-Alive

The typical response to a GET request is principally a retrieval of data requested. The body if included should contain files requested such as images or text files etc.

An example response that can attend the request in the script above is as below:

HTTP/2.0 200 OK
Date: Tue, 01 Oct 2019 11:35:23 GMT
Server: Apache/2.3.14 (Win32)
Last-Modified: TUE, 02 Sep 2019 16:32:22 GMT
Accept-Ranges: bytes
Content-Length: 98
Content-Type: text/plain
Connection: Closed
……

Invitation.html

2. HEAD: Works the same as the GET method; however it doesn't return a body but only a header line. The head is almost used when the client needs to request information about the file or information they need to request,

The Head response consists of the information contained in the response's status line and header.

3. POST- the POST request is used to direct data to the server, the post method is used mainly for submitting files, data, client's details. This requires the client to submit an HTML form with the POST method to the server.

The POST method does not retrieve anything in response the server solemnly, it confirms the status of the request process (e.g. approved, denied).

4. PUT- the PUT method is used to upload data to the server corresponding to certain URI. It operates with the body just like the POST method but used when a replacement to existing resources need to be made. The response to this method first implies the server to store the resources and then respond with a notification that a new resource is created.

5. DELETE- the name says it all. It is used to erase the existing resources on the server that correspond with a given resource identifier. The response includes a confirmation if the procedure has been successful with an OK, accepted if received but pending action or No content if no file was found.

6. CONNECT- It produces a network connection to connect to a server corresponding to the specified URL. This is done when a connection is needed through proxies. The server responds with a confirmation response (e.g Connection established.)

7 OPTIONS- this method works as a request to the server for specification of communication options that the specified URI supports. The server returns all the request methods the client can use.

8 TRACE: Also known as diagnostic method is used to test the authenticity of the operation since it returns the exact content of the request to the client. It is mostly used in debugging because of its loop back characteristics. The server returns no body but an OK confirmation.

2.4.2 Back-end Design

The back end development is the logic and the implementations of the elements that take care of the data management and make the web application viable[29]. The back end involves elements such as the server - and database. At most cases, tools such database management systems are used; the APIs, their operating systems and the most important, a web framework that integrates all the ASP.NET Core.

The back-end development is up to the developer's specifications as long as the client's requirements are met. They are developed with the main programming languages that are supported by the framework used in developing it. The most preferred and common languages are[30] :

- Java,
- C# /C++
- Node.js together with JavaScript.

They are developed mostly in frameworks for easier development. Frameworks are opinions on how best to solve a problem. They are developed to create standards for software developers to write code in a standard or expected way. Frameworks often include code libraries that are known to be dependable and that developers are expected to use. The most popular frameworks are:

- ASP.NET, support all the .NET languages (C#, Visual Basic, etc).
- Django preferred with python codes,
- Express JS, preferred for Node.js

2.4.2.1 ASP.NET Core MVC

Developing a back-end system can be a nightmare for a software developer/engineer. Software engineering is plagued with problems of application, maintainability and scalability amongst many others. Frameworks provide solutions to common software development pitfalls. Frameworks also provide commonly used code libraries that provide solutions to common problems and often enforce opinions on how developers should organise their code in order to avoid common pitfalls. One common opinion that is largely considered to be the best practice in software development is the idea of having a 'separation of concerns. This is enforced in MVC frameworks where developers are forced to have a separation between the data models, the controllers and views.[30].

Among all other platforms, ASP.NET is the most favoured. This is backed up by the stack overflow survey conducted in 2019, with more than 88 000 correspondences.[29]. One of the main causes to its rise in popularity is the amount of reliable (tried and tested over many years) libraries it has that supports the development of all kinds of applications (Web, mobile, window, APIs etc) [31].

ASP.NET is a Microsoft product (developed and maintained) which has been around for many years even before the NET Framework was released in 2002, while ASP.NET Core first came into light in 2016 [31]. It is released as an Open Source platform and it is considered as ASP.NET for all operating systems since ASP.NET only runs on Windows. The idea was to come up with the Core version that is compatible with all kinds of operating systems.[32]. It can be run on Window, Mac, and Linux Operating systems for research and development in developing the following[32]:

- General Web applications,

- Mobile Application
- Window OS applications (Server, phones and computer)
- Cloud based online Web applications (Microsoft Azure)
- APIs. (Web and Mobile)

.

Main features and benefits of the ASP.NET core platform include the following [32],[33]:

- Fast and scalable
- Data security is enabled with ASP.NET through its app security Cross-Site Scripting.
- It's not based on a singular operating system but rather on its ability to run on various systems making it a cross platform, including Microsoft Azure (unrestricted Hosting).
- It is open source, used actively on Stack-Overflow and GitHub
- Unified platform for both applications and APIs in one program.
- An Integrated command line tool
- It is compatible with other NET Frameworks,
- Unified platform for both design and testing
- Razor pages combination.
- It supports several languages.

ASP.NET Core Web Applications Models

This Microsoft technology is executable through the Microsoft Visual Studio Community. The platforms allow the development of applications through optional models or views. Web application follows the Model View Controller platform or the Razor Pages model. In the ASP. NET architecture designs, the MVC web applications, pages and Web forms as part of sites and the Web API and SignalIR as services. Figure 8 shows an illustration of layers of the ASP.NET architecture.

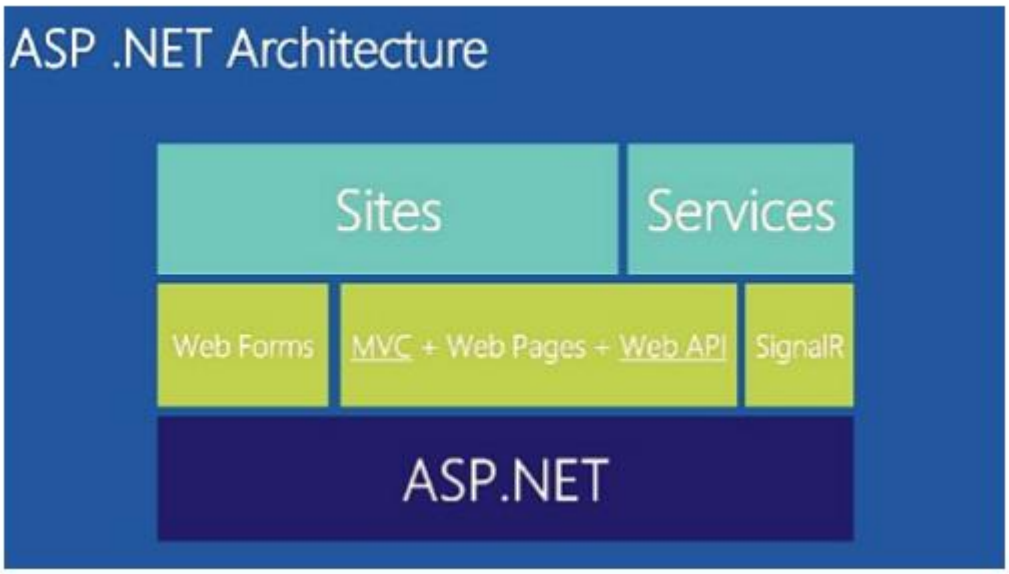


Figure 8: The ASP.NET Architecture[30]

Model-View-Controller (MVC).

The application codes are done to meet different tasks that then make up the application such as the end user interface and the business structural part of the program which is the logic that defines the application. To achieve this separation, the developers use the MVC as a structural pattern that allows the programmers to divide their codes according to their responsibilities' (separation of concern)[32].

The reason why the design is preferred to be done following the MVC architecture, in comparison to the normal freelancing coding and development run behind the numerous advantage and quality, this architecture add to the web applications developments. A few can be summarised as:

- Allows parallel programming
- Removal of dependencies on other classes making it easy to design and develop, fix errors and test the application distinctly.
- Separation between the front and the back end
- Multiple views
- Changes made to the controller or the view have no impact on the model
- Easy scalability,

In the MVC approach, the codes are separated into the Model responsibilities, the one responsible for the View and the Controllers which interact with each other as illustrated in Figure 9. All the components can be programmed with the same language (C#, C++) and together they are compiled into one file that can be executed[34].

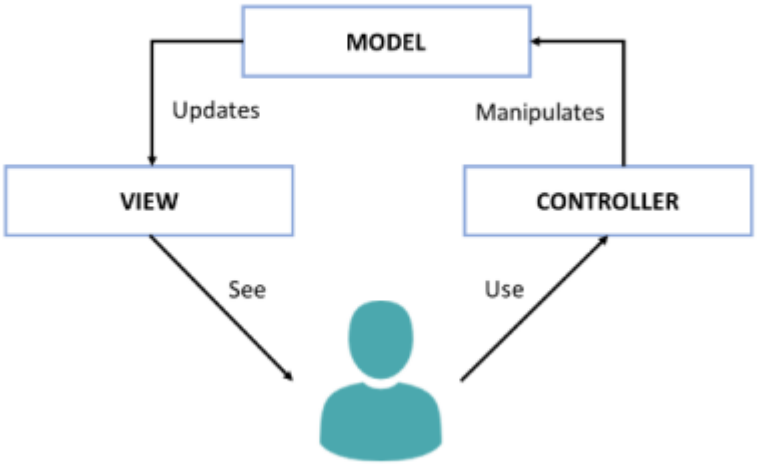


Figure 9. Model, View Controller interactions [35]

Model Responsibilities

The Model also known as the Domain Model is the centre of the whole architecture as it illustrates and contains most of the application business structure and logic. It is responsible for defining the operation and the implementation of the business logic of the application [30]. Being the central domain, it as well links the data to the user, depending on the logic that the user uses through the controller.[34].

Defining the Model section means introducing classes in the application that define the business logic, and the content that enables communication with the database. This is done in the Model subfolder

that comes predefined by the system or mainly through the Entity Framework Core. All the sections are automatically predefined when starting a new project[32].

View Responsibilities

The view section is responsible for the representation of the application, it's where the User interface logics are defined and determine what the user view, to enable interaction with the application. This is where the front end of the application is designed[35]. The view engine is responsible for generating HTML for clients.[34].

Defining the View components as well does require the programmer to define classes to enable the display of the content on the User Interface (UI). [30]. To enable the displaying of content and the communication between the Model, the programming of the View is a combination of HTML and the .NET Programming language in use, hence the View classes are saved as e.g.. cshtml (for c# programming languages) to accommodate the two. To switch and differentiate between codes a @ symbol is used. The view content is highly influenced by the inputs from the Controllers.[36]

Controller Responsibilities

The Controllers determine the behaviour of the application. This depends on the inputs through the interaction with the user. The behaviour is defined by handling the user's interaction via the browser's HTTP requests. Controllers translate input (or request) to action (or output/response) for the application This include: [35].

- security, routing, and switching.
- Get data from the model
- Provide data to a specific view depending on the HTTP request that is used or;
- Provides a HTTP request to get data and give a response.

Defining a controller includes creating *action methods.* The action methods jobs involve getting and matching the browser's request to the business concerns through logics defined in the controller classes[37].

2.4.2.2 Database

When developing a Web application, the developer must consider among other things, the storage of their contents. Since web application often deals with enormous amount of data, the option of using the server to store data as well may not be pleasant as to using a database [25].

A database is a collection of data. A database server is either a physical server or an application that is used to access the data. Information on a database are assorted in a certain way to allow users not only to store the data but also to engage with it regularly through entering new information, edit, retract and data analysis[38].

A database can be a single data source customized for a certain application (single user application) or a shared collection of data (multiple users applications) however, its structure typically comprises of [38]:

- Tables – consist of assorted interrelated data in different or similar data format, e.g. data related to patients in a Health Information system

- Fields (columns) – one type of data in the same data format belonging to the same group that is normally inserted and stored e.g. The demographic information of the patients like Name, age, contact, etc. are fields of a table belonging to patients.
- Records (rows) – information belonging to different but associated fields. E.g. the Names of recorded patients.

To communicate with databases, a Structured Query Language (SQL) is used in database management systems to query (access and retrieve, analysis and manipulation of data) instances of the database. [39].

Database management systems are applications built to interact with database servers. The applications make it easier for the user by providing GUI that contains tools to easily query data stored in the database. Querying the database is simplified by database management systems by providing intelligence driven query editing windows and GUI controls that often query databases without writing SQL code. A commonly used database management system is SQL Server Management Studio[38].

Database approaches for the MVC framework

Creating an application database when using Entity Frameworks, there are various approaches that the developers can utilise. These approaches depend on the developer's need. For example, if there is an existing database already or it's a new design, etc. Further details on the approach options can be observed in Figure 10. They are; Database First, Code First or Model First Approach[40].

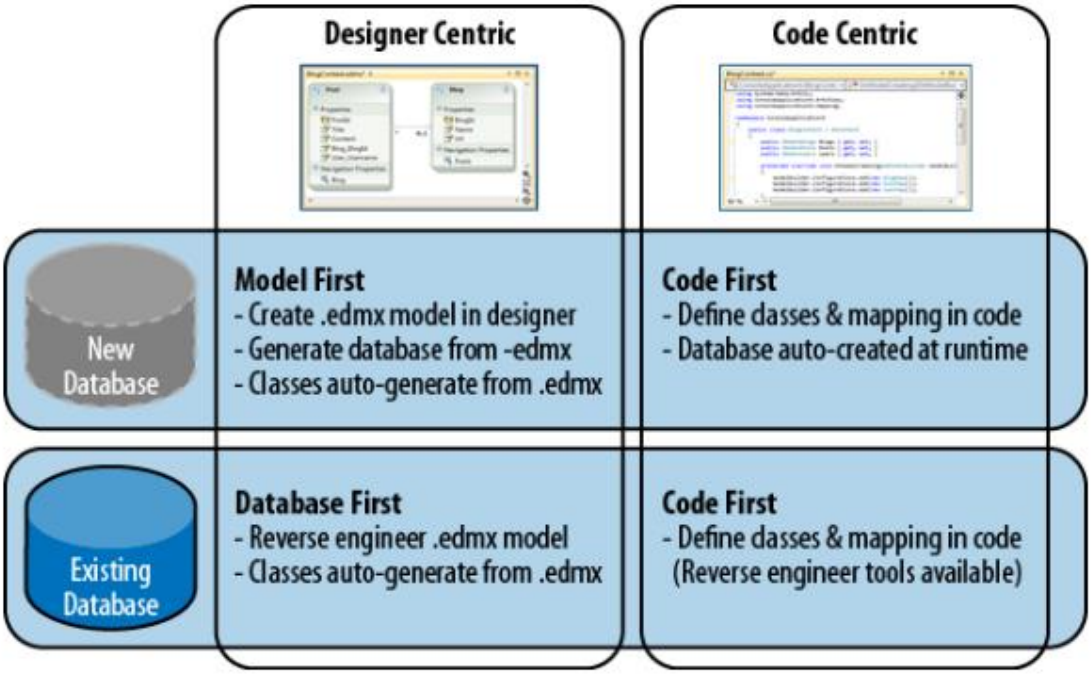


Figure 10: Database approach options[40]

With the database first approach, the developer creates or uses an existing database first and the code will take care of the class generation automatically. While with the code first, the database is created automatically. The best and most effective approach depends on the requirements and the nature of the application. The code however first approach creates a database that is completely depending on the developer's data model and Entity classes created in the model.[40]

Code first approach database migration

Data models are dynamic and change constantly. Amid this dynamic change, they have a constant need to be in sync with the application. The best way to preserve the existing data while at the same time updating constantly to synch the model, data developers can use the database migrations[41]

To achieve the dynamic database management approach, where the database and the data model are relational and updated simultaneously, the code first approach database migration is used. In this method, the developers program a code that deals with updating the database every time the data model has changed. The code will apply the same changes to the database as well. This eliminates the constant need to keep recreating database tables[41].

When preparing for or before deploying the application in the code first approach, the database is then created automatically through Entity Framework migrations. The first migrations are done through the .NET Core Command Line Interface CLI. To create the first migration, the developer simply runs the *add migration* command on the CLI and the database migration files are added to the project in the Migration folder. This is an automated process and it depends entirely on the pre-established codes and classes that are defined. To ensure the data model and the database stay in synch, the migrations have to be updated with the *update database* command.

2.4.3 Application Programming Interfaces

Application Programming Interfaces are software interface that enables applications to communicate with other applications, servers, operating systems or data centres without needing to integrate the underlying process.[42] API acts as communication protocols simplifies the interaction between applications and third-party developers by allowing them to exchange information. There are three types of APIs: Open API, Partner API, and Internal APIs illustrated in Figure 11.[43].

- Open API – Open APIs offers data or services publicly without any restrictions or rules, other developers can modify or use the API in a self-service manner.[44]
- Partner API- They are created as an Open API type but only the business partners have access to the API[43].
- Internal API- They are used for closed development on related applications only.

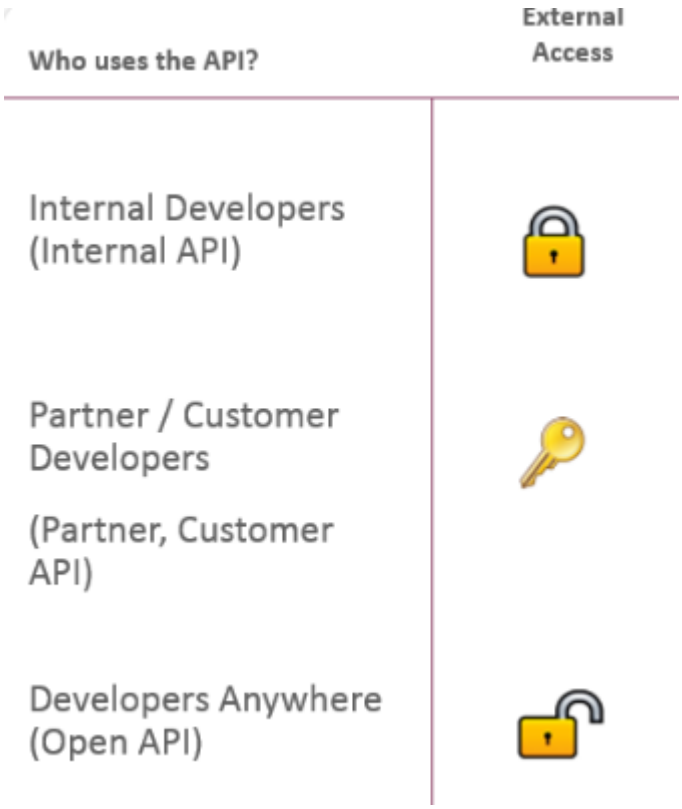


Figure 11 Different API types.

2.4.3.1 Why APIS

In the IT world, there is a great drive to have independent services as opposed to monolithic applications that provide multiple services. APIs provide access to different services using standard access methods. Having smaller services is much more desired because multiple teams can work independently and from that one can get more benefits. API provides developers with building blocks that can be assembled to create a new application. They allow interaction between applications that were made at different times, programmed in different languages and by different programmers[45].

APIs enable applications to be decoupled. The front end (JavaScript, CSS, HTML) can be separated from the back end (databases and MVC app). One can have multiple front-end applications (example a web application + a mobile application) using a single back end application (accessing it through an APIs)

APIs are used to integrate systems through the following motives[45]:

- Bi or Multi directional exchange of information in independent systems or applications
  APIs support the communication in parallel applications where different departments operate different applications independently but to achieve a common goal
- APIs support application conditions and rules to be in place, while exchanging information without compromising other data or application structure. When transactions are taking place, other services can be off the limit. Restrictions ae preserved and data is protected.
- APIs enhance the quality of services and customer satisfaction by enabling more new supports to be added to existing services without changing the service layout.
- APIs enhance the implementation of Machine to Machine (M2M) communication services in society. The restriction and preservation of rules enable the API to give confidence in allowing M2M communication without compromising the security of data.

Simple Object Access Protocol (SOAP) APIs

SOAP APIs are standardised protocol styled API that is established between two applications on how they should communicate [46]. SOAP API only supports XML data formats and it transfers structured data. Figure 12 illustrates how SOAP data need to be transferred via SOAP standards to reach the server.

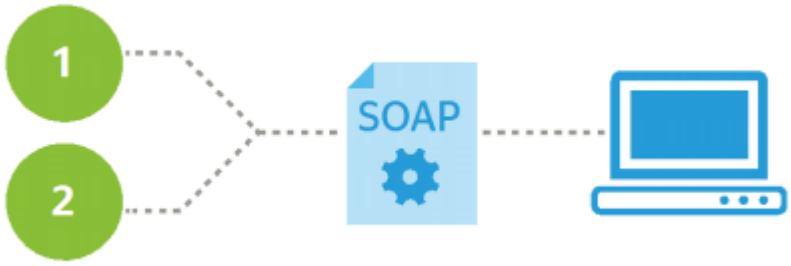


Figure 12 Simple Object Access Protocol APIs[46]

2.4.3.2 Representational State Transfer REST APIs

REST API architectural style uses the simplest format to communicate between applications or hosts and servers. It does not require data to go through a certain standardization process. Figure 13 demonstrates that unlike the SOAP, REST API relies on communication between the server and client[47].

The data transfer supports XML or JSON data formats and uses but not strictly HTTP protocol[46]. Just like most HTTP formatted communication, REST APIS uses the HTTP request methods. On the contrary, while REST is the API architectural style, RESTful is a state when the POST, GET, PUT and DELETE requests are made[47]. These request methods are explained in Chapter HTTP Request and Response methods2.4.1.4.

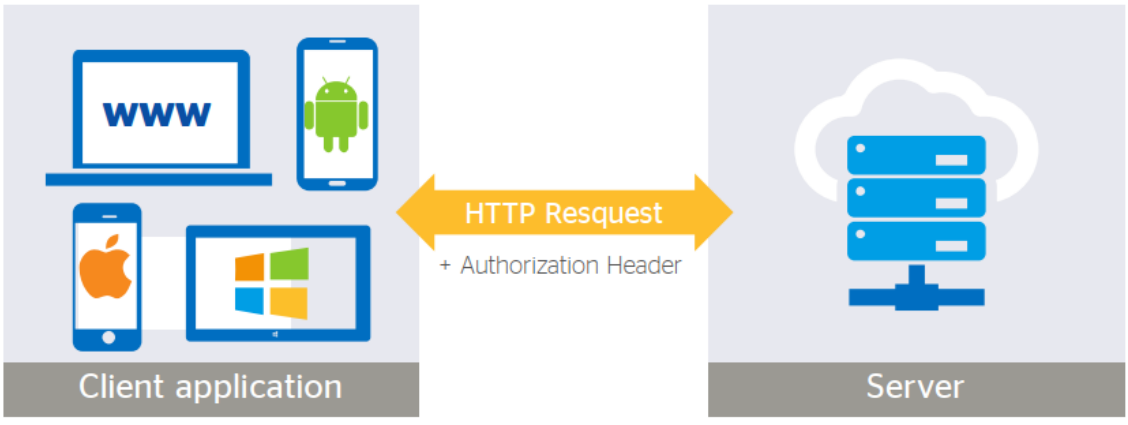


Figure 13: REST API communication[46]

Why REST APIs?

Due to its simplicity in implementation REST APIs have taken more popularity in ICT than SOAP API because of the following objectives[48][47][46]:

- RESP API is a stateless client/server protocol since it uses HTTP protocol, each request runs independently and the server does not need to keep session information or relate them to previous sessions. This makes the API uncomplicated and easy to scale
- Operating the elements at the Uniform Resource Identifier makes it impossible to access other information since it's an only resource identifier for every resource
- Uniform interface allows the components to be transferred and they are associated with their processes. Only precise acts are to be operated for example; the identification.
- The use of hyperlinks or hypermedia also known as Hypermedia as the Engine of Application State HATEOAS, which allows the information to be transferred through hyperlinks as hypermedia.

2.5 Literature Findings

This chapter focused on reviewing the Health Information System in place: how they work, their technology and what solution they provide. Understanding them will make it easy to find out the right method to use to integrate them. The chapter further tackles the theory of system integration and the methods of integration, the requirements of HIS integration and levels to be operated. This concluded the application integration to be the suitable methods for integrating the HIS in place both horizontally and vertically. Having found the methods, the chapter ends in describing the technologies and theory behind achieving the solution of creating Web applications to demonstrate the integration and APIs to allow interoperability.

Concluding on the findings of Chapter 2, the E-Health and the Pharmaceutical systems can be integrated to have a single data entry that would together feed the dashboard of the DHIS2. This would reduce data redundancy, duplication of functionality while enhancing the collective use of data.

3 System Design

The content of this chapter is the presentation of the proof of concept needed to comprehend the implementation of the design. The objective of the thesis is to primarily prototype two subsystems of the health information systems being the E-health and Electronic Dispensing Tool and secondly, integrate the two. The methodology outlines and details the research work as well as the design structure.

The desperate need for automation and faster service delivery is the main drive for the proof of concept implementation. This chapter introduces the design of the experimental prototype proposed as believed to be a solution to the problems presented in Chapter 1. The chapter is presented through the proposed overall designs, the implementation overview, the design set up and desirable results.

3.1 Procedure Overview

The components of the design will include the simplified replicas of two systems chosen for demonstration purposes, the E-Health system, and the Pharmaceutical system. How the two original systems used in the use case operations are introduced and described in the literature review. The prototype designed are ought to represent the systems in use for the selected case study. The prototypes are designed with the purpose of being used in the demonstration of the integration process only. They serve to demonstrate and illustrate how the selected hospitals in the case study can benefit from the integrated system.

The waterfall model, in Figure 14was used to implement the whole process of the design implementation of the integrated system applications and the chapter concept

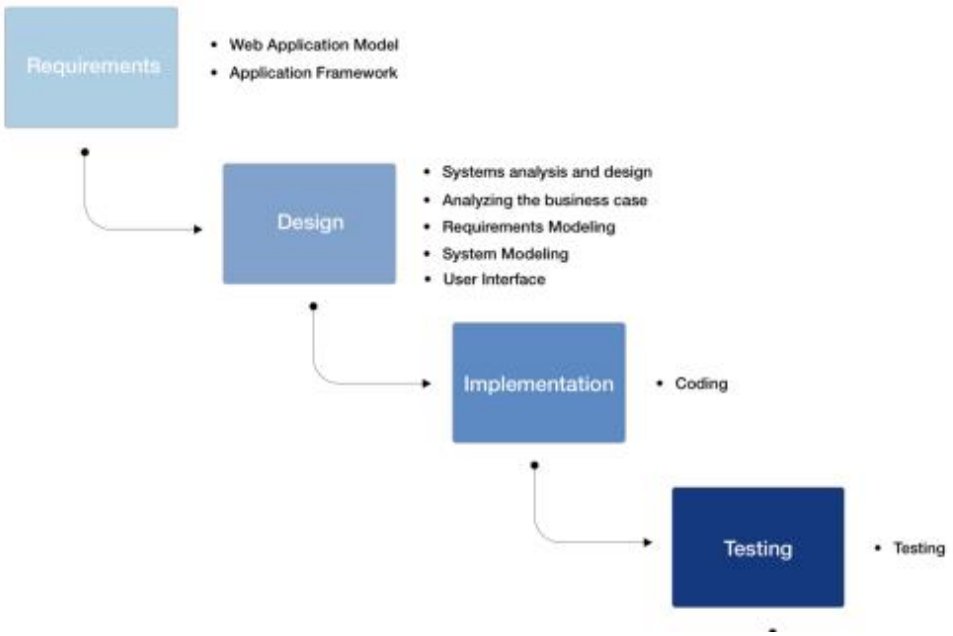


Figure 14 System implementation overview.

The overview of the overall integrated system to be described is specified to detail the concept and better comprehension with the fragmented design explanation. Through the literature review conducted and presented in Section 2.2, the general functionalities of the two systems are explained and presented. They

introduced the path to choose the methods and technologies used to prototype the desired functions to be used in the demonstration.

3.2 System Requirements

Through the analysis of the Literature review on the systems to be integrated explained in Section 2.2, the system requirements are discovered. To achieve the functionalities of the integrated system as desired from the systems, the following requirements are outlined; The functional requirements give an overview of what the systems are expected to ensure while the non-functional requirements are the technical matters that are required to ensure that the functional requirements are met. They are conditions in the aspects that deliver system performance, security, and software development specifications.

3.2.1 Functional Requirements.

The two sub-systems are specified to meet the following requirements from the end-users. The following section describes the business model's functional requirements for the two sub-systems. The overall objective is to integrate the sub-systems. The overall functional requirements of the integrated system are defined in the following bullet points. To meet them, first the sub-system functional requirement defined in Section 3.2.1.1 have to be met.

- ✓ Two-way communication between the E-health system and the Electronic Dispersal tool system
- ✓ Electronic Dispersal tool to access the E-health database only for selected and required data.
- ✓ Feedback from the Electronic Dispersal tool to the E-health system
- ✓ Enabling data integrity with single entry of data

3.2.1.1 E-Health Functional Requirements

The E-Health sub-system is required to work as a standalone system as it serves currently in the use case presented in Section 2.2. It operates as an Electronic Medical Record system that should meet the following requirements:

- ✓ Patient record management, whereby a record of all the patients together with their medical history, demographic information, etc. is kept electronically other than the booklets that are still currently in use in most of the hospitals in Namibia especially in the rural areas.
- ✓ An admin interface to allow the administrator to add users.
- ✓ An interface to add patients, this includes functions for attaching their demographic information.
- ✓ Functionality for adding descriptions and diagnosis per patients
- ✓ An interface to add and select the hospital, this is limited to the number of hospitals that the application is implemented.
- ✓ The participant interface to view all registered patients and users

3.2.1.2 The Electronic Dispersal Tool Functional Requirements

Electronic dispersal tool in place in the case study hospitals is operating under the Pharmaceutical Management System to serve the entire operation of the pharmacy which include medicine inventory and stock taking. The prototype in the demonstration is limited only to the sub-system of the Electronic Dispersal tool. The reason behind this discrimination of implementation is because it is the only section of the system that correlate with the E-health system. The prototype system on its own however does not entirely replicate the original case study system nonetheless a section of it. The subsystem is required to have the following functions

- ✓ The searching platform to allocate a new prescription. The searching reacts to an individual's identification number
- ✓ The acknowledgment button for medicine collection approval
- ✓ The participant interface to view all registered patients, the number of prescribed medicine and the number of the collected one out of the prescribed ones.

3.2.2 The Technical Requirements

These are non-functional requirements that guarantee the quality of Service in the attributes of the application,

- ✓ Security - Confidentiality is an issue in the health industry. The system should ensure the secureness of the patients' health information through authentications and data encryptions.
- ✓ Capacity - Every day, new users are aggregated to the database, one of the system requirements is that its database set up should be able to rescale and accommodate the incoming number of users and patients' data.

3.3 Design Methodology

With the application requirements outlined in Section 3.2, the application design stage is equipped to take off. The guidelines for the design of the overall application scheme as elaborated in Chapter 4 are;

- ✓ The modelling of the application requirements.
- ✓ Its sub-systems organization.
- ✓ The complete integrated systems architecture design.
- ✓ The architecture of the development process, in particular.

3.4 Implementation Methodology

The methodology used in the realization of the system is explained fully in Chapter 5. In this section, the environment for implementation and testing is chosen and set up through software installations and configuration. All the technologies and protocols used are explained in Chapter 2.3. This includes all the sub-systems and API implementations. The implementation methodology involves coding and connection systems.

3.5 Testing

Testing involves assuring the functional readiness of the application. Taking into account that the aim is to integrate the systems, API access availability and validation test are conducted through testing software. This will complete the design process the checklist is signed off. The API test will determine the following:

- ✓ The HTTPS requests are working.
- ✓ The request methods components (GET, PUT, DELETE,) are interacting as expected.

- ✓ Verifying if the API is updating the data structure as they should. (Prescription feedback post)
- ✓ Correct status code is given
- ✓ Performance profiling of CPU and Memory usage
- ✓ API validation

3.6 Design Overview

To give an understanding of how the design arrangement process is achieved, the diagram in Figure 15 is used. The design as shown in Figure 15 is divided into three parts that are designed independently. The two sub-systems and the API together with an interoperated database form an integrated system. Other parts of the system included in the design are the database and server.

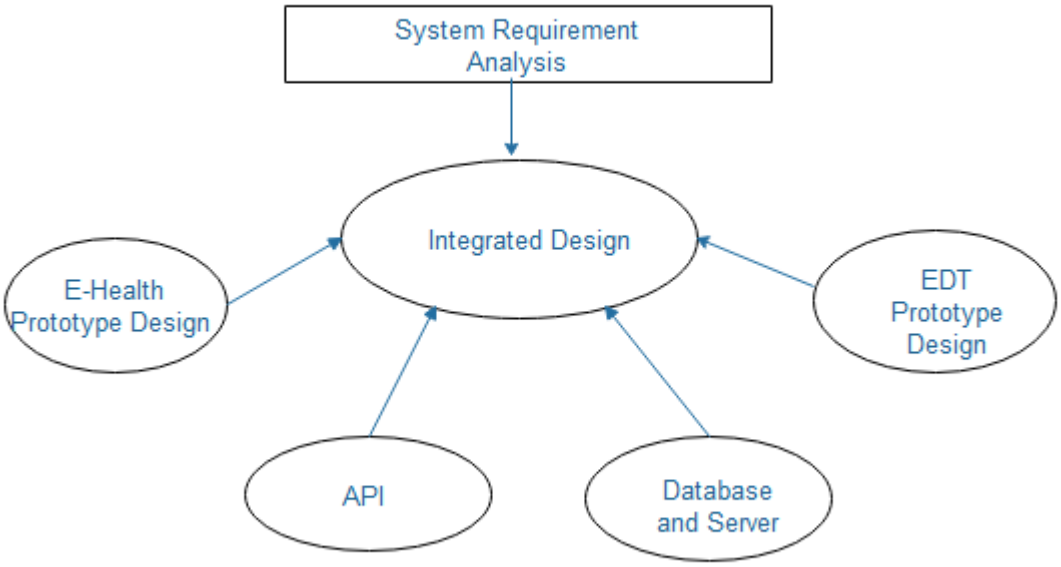


Figure 15: Integrated system design overview.

3.6.1 Overall Architecture Overview

The integrated system comprises of two prototypes; the E-health and the EDT systems with one single data entry point through the E-health. The E-health sub-system should contain the patients' information and doctors' prescriptions in its database. The EDT, however, does not have data on patients but rather on the medicines in stock only. The EDT is in the custodian of the Integrated Pharmaceutical Management system that manages the pharmacy operation such as medicine inventory and stock taking. To prescribe medicine, the EDT requires patient information and prescription which can be accessed from the E-health system through system integration. An overall architecture overview on how this is conducted is given in Figure 16.

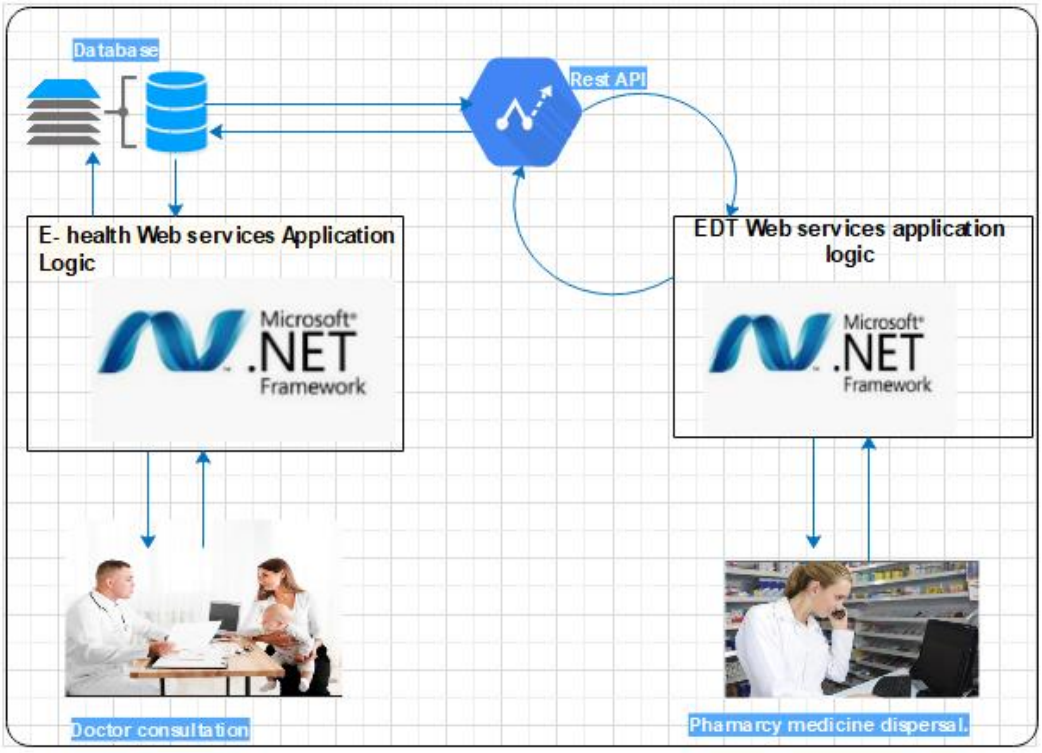


Figure 16: Overall system architecture overview.

3.7 Prototypes Design

The two prototypes are both web applications and designed to be implemented, the design method illustrated in the following models. Their implementations and requirements achievements are explained in Chapter 5. The basic design of the E-health is presented in Figure 17.

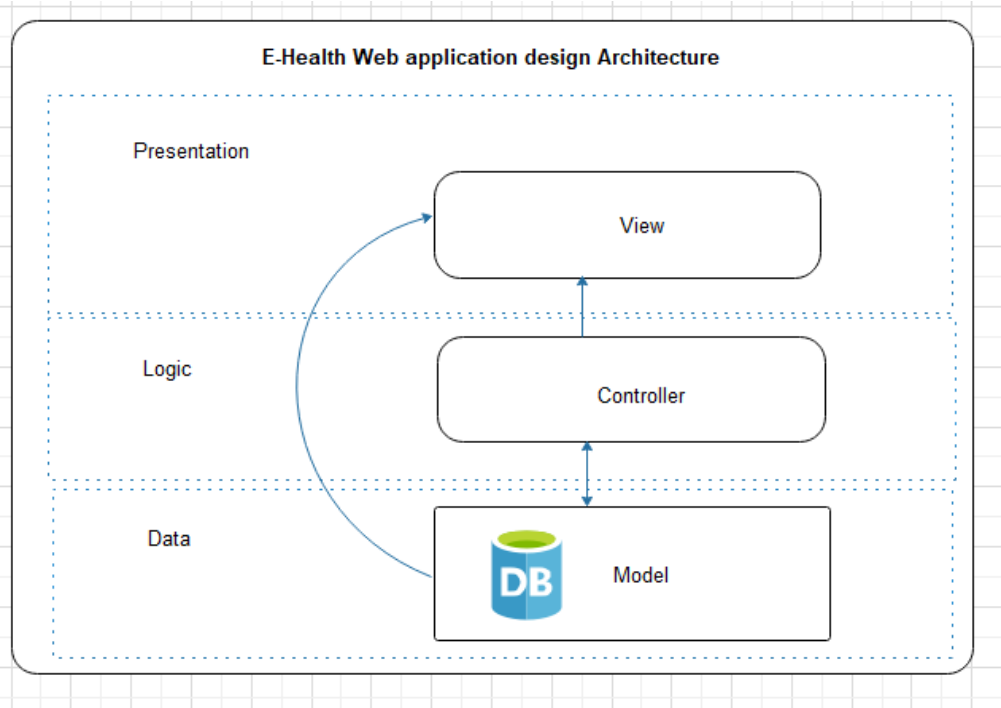


Figure 17: E-health design overview

The overview is a general representation of a web application design method following an MVC architecture. The architecture provides commonly used code libraries that provide solutions to common problems and often enforce an organization of code in order to avoid common pitfalls. In the MVC, this is enforced by having a separation of concerns between the data models, the controllers, and views.

The EDT system follows the same architecture however, the connection between the controllers and the data model is controlled and enabled by the API. The design overview is shown in Figure 18.

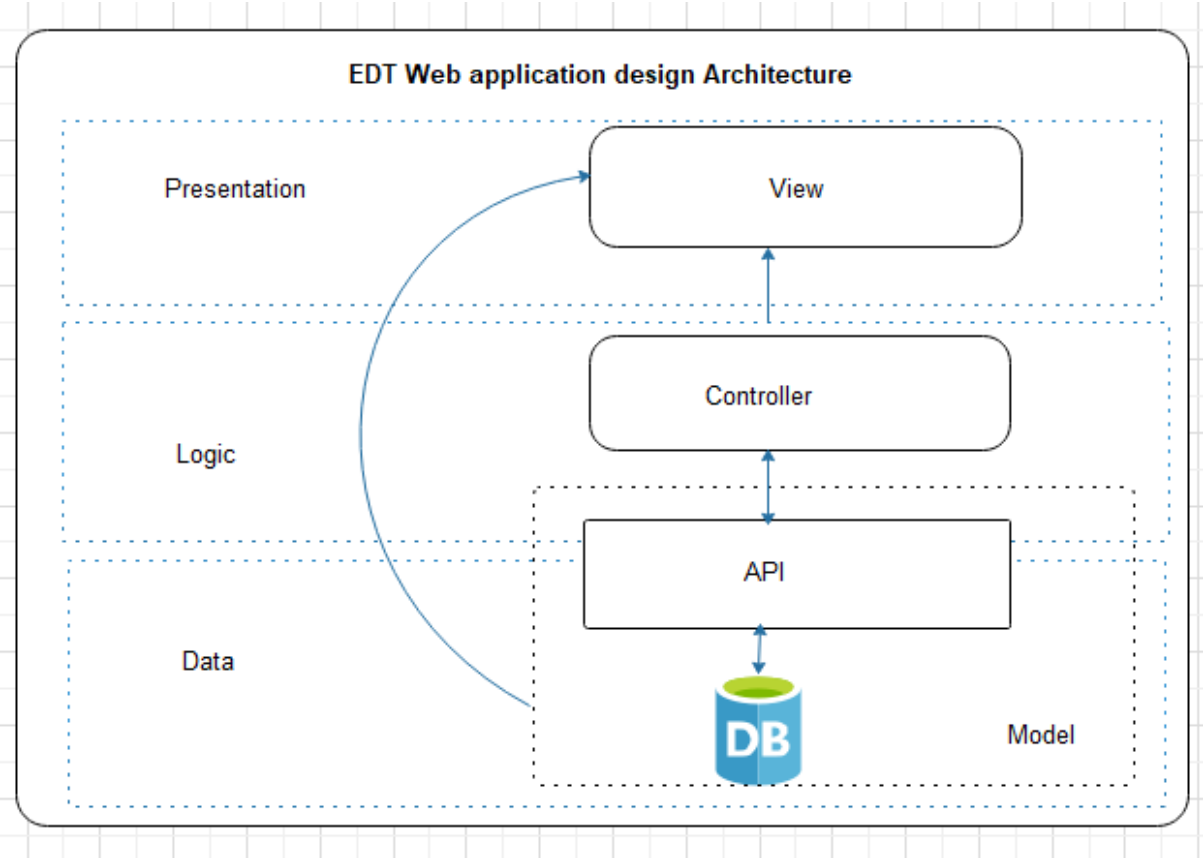


Figure 18: EDT design overview.

Presentation – View classes

The presentation layer covers most of the web application requirements via the user interface. It operates in the model module and is associated with the acquisition and manipulation of data. The presentation is implemented in the view classes. They allow the user interface interaction with the rest of the application.

The model- Data handling

The model takes care of the controller's queries to the database. Constructs every database instances that allows manipulation of data in the database such as; get, set, list, create, update, delete, select, The E-health contains model classes for the basic instances of the system such as the Hospital operation logic, the patients, Pharmacy lists, Patient viewing, prescriptions, etc. It is as well responsible for the migrations of the database tables. It enables communication with the database in the E-health application through the model codes.

Logic -Controller

The classes that operate as an intermediator between the model and the view, all the action methods are created in the controller classes. The Controllers determine the behaviour of the systems.

3.8 Web-API Design

To enable the integration between the two web applications, there is need to be communication between the two. This is achieved through a web API. The aim of creating a web API for the E-Health web application is to ensure that the EDT is integrated into the E-Health's Database in a way that it can retrieve and modify authorized data hence eliminating data redundancy and enhancing faster service delivery. They communicate using the HTTPs protocol (among others) as a format to transfer data with; GET, PUT, POST and DELETE requests

The API is designed following the MVC architecture in the same ASP.NET Core platform. It does not require the view section as it is a Back-end system without a user interface. In this development, a restful API is used due to its advantages spotted out in Section 2.3.3..

The EDP tool application's necessities included the medicine prescription straight from the E-Health requests. The only controller classes that need to be integrated are the patients with their numbers as their identification, the prescription itself that the doctor prescribed and correspond to the patients' ID and the values of the total number of active prescriptions.

4 Integrated System Implementation

The first phase of the implementation focuses on developing functioning sub-systems prototypes and their basic front-end functionalities and services. After developing the prototypes, the back-end API design is then developed to integrate the two.

4.1 Implementation Tools.

Before the implementation is done, it's necessary to select the suitable approach for the methods and instruments to use in the development process.

4.1.1 Development Hardware

The prototypes for the E-Health system, the Electronic Dispensing tool web applications and the API were implemented run and tested on a Dell Laptop operating with the following specifications:

- Windows 10 operating system.
- Intel(R) Core i3 6006U CPU @ 2GHz processor
- 8 Gb RAM

4.1.2 Development Software

To provide the necessary environment for the development, the following technologies, software programs and platform were installed in the laptop:

4.1.2.1 Microsoft Visual Studio community edition.

Microsoft Visual Studio is an integrated computer application that creates an environment to develop, test and run the Web application prototypes and the Web API to integrate the two prototypes. This was selected mainly because of the following reasons:

- It is free of charge; the community edition is open source and comes at zero cost ideal for students and low-profile developers hence good for academic research.
- It supports a lot of programming language including the C# with the .NET Framework using .Net core 2.2 which is used for the development of this project. Other built-in popular languages supported include JavaScript, C, C#, .NET, etc. and others such as python, node.js, and Java with the use of plugins, etc.
- It is integrated. This gives advantages as it contains a code editor with good features for the code development and edit, the debugger for testing and the builder for build and run apps.

This application is the core of the project development as it provides the runtime environment for the C# ASP.net core codes. It can be downloaded from the Microsoft official site and be installed in any window computers. This project used the Visual Studio 2019, version 16.3.4

4.1.2.2 Microsoft SQL Server Management Studio (SSMS)

Like any other information managing systems, the prototypes must run on a server with a database to store the data being projected into the systems. The designed systems use the SQL Server Management Studio (SSMS) version 18.2. The SSMs is an integrated environment used to create and

manage the Microsoft SQL Server SQL Database. It is a form of database server developed by Microsoft and can be downloaded to be used on Microsoft Window operating systems, but it can also run on Linux or as an image on a Docker container.

There are dozens of databases and database management products, like MySQL, Oracle MaxDB etc, Microsoft SSMS was chosen particularly to be used for this design because of the following main features and advantages:

- It is obtained free of charge, again perfect for academic research,
- It allows integration services selection upon installation. Since they are from the same developer (Microsoft), they make integration with the application developing tool easier,
- Object Explorer: developer or administrators can navigate, choose and act upon elements that are in the server.
- It allows the developer to configure, manage and keep an eye on the elements of the server and the database.
- It can run databases on the local computer or on the cloud. (Developer's choice)

The free SSMs is downloaded and is used in this design to deploy the database and server on a local computer. The database designs its deployment using the code first approach migration and the procedure for deployment is explained in the project structure and implementation section.

It is run autonomously with SQL Server database engines SQL Configuration Managers but their releases are independent.

4.1.2.3 SQL Server configuration Manager

The configuration manager allows the configuration of connections such as the services and client network configurations. It is not a standalone program and it runs together with the server management studio. It can also either run as a local server or on a cloud.

4.1.2.4 Programming languages.

The programming language and protocols used for both front end and back end design are explained with more details in Chapter 2.4.1.

C#: Back -end implementation in ASP.Net framework.

HTML, CSS, and JavaScript: Front end implementation.

MS SQL: Database.

4.1.2.5 Testing environment

SWAGGER Inspector: testing tool that is used test the readiness and connectivity of the API. It's an open API Initiative that test API by calling the endpoint and gives outcomes in forms of request responses. Swagger provide API development environment as well as testing environment.

4.2 System Implementations.

The eHealth and EDT applications are implemented with the MVC architecture. The Net Entity Framework was used to create Data Models. The views are written in HTML Java Script for the web application outlook and CSS for styling), though there are some c# codes for logic processing before sending HTML off to the client browser. Controllers are pure C# implementations of the application.

4.2.1 Models

In the eHealth application there are Doctors, Hospital, Medicine, Patients, Pharmacy, Prescription models which provide abstractions that help us to easily work with data in the corresponding tables in the database. In addition to the data models, the application also have view models, namely, Error View Model and Patient View Model which are primarily used to structure data for the HTML views. Figure 19 shows snapshot of the E-health models in the solution explorer of the .Net Entity Framework.

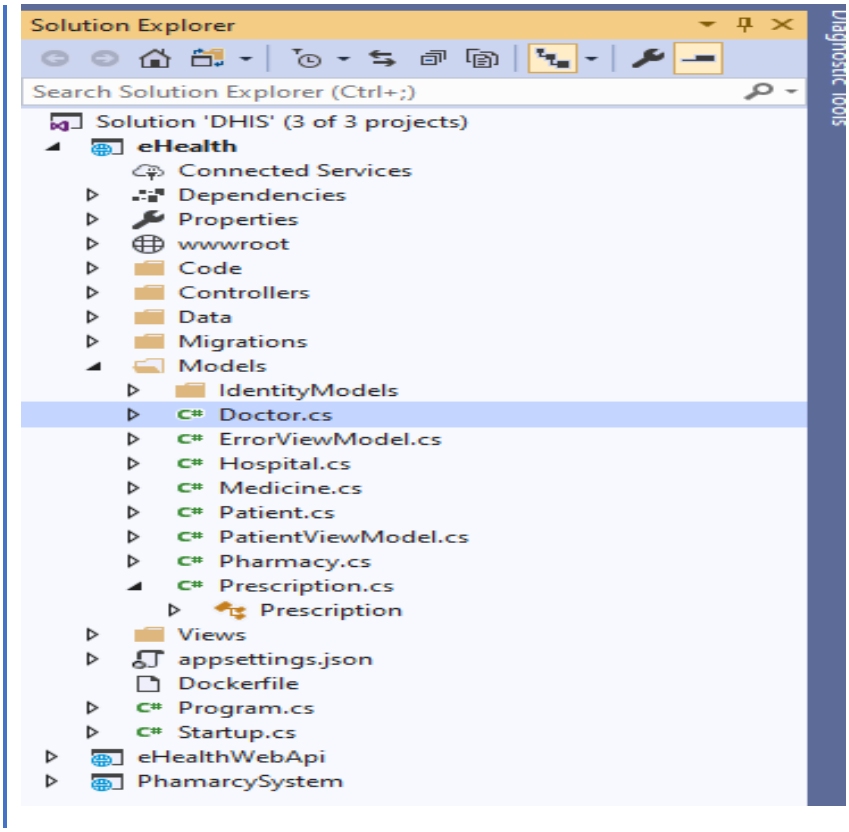


Figure 19: E-health Models.

The following is an example of a code for the Hospital Model.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;
```

```
namespace DHIS.Models {
   [Table ("Hospital", Schema = "dbo")]
   public class Hospital {
      [Key]
      [Database Generated (Database Generated Option Identity)]
      Public int Hospital ID { get; set; }

      [Required]
      [Column (Type Name = "varchar(100)")]
      [Display (Name = "Hospital Name")]
      Public string Hospital Name {get; set;}
```

The Hospital Model has two properties, namely, Hospital ID and Hospital Name which correspond to columns in the Hospital table in our database. The Hospital ID is the primary key in this table. The database generates a unique integer for the primary key. This functionality is programmed using the [Key] and [Database Generated (Database Generated Option Identity)] decorators. This is implemented to enable the addition of new attributes to the hospital class without having the need to redefine it. The new addition will inherit from the defined class.

The Hospital Name is a string stored with a maximum of 100 variable characters. This field is always required for each Hospital object. This model definition is implemented in a similar manner for all the models (Hospital, Patients, Prescriptions and Doctors) that require addition of attributes at run time.

4.2.2 Views

Views are primarily for displaying data and there is not much logic in them like there are in the controllers. Below is a code snippet of the create view. This is the view an authorized user gets when they wish to add a new hospital to the database.

```
@model DHIS.Models.Hospital

@{
   ViewData["Title"] = "Create";
}
<h2>Create</h2>

<h4>Hospital</h4>
<hr />
<div class="row">
   <div class="col-md-4">
```

```
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="HospitalName" class="control-label"></label>
                <input asp-for="HospitalName" class="form-control" />
                <span asp-validation-for="HospitalName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </form>
    </div>
</div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.Render Partial Async ("_Validation Scripts Partial");}
}
```

The @ symbol can be observed as it is used to change between the c# codes and HTML. The "View Data "is a syntax (for c#) view engine that is responsible for generating HTML to clients. The rest are HTML codes that allow the display of the hospital name label once the hospital name is created. This is implemented similarly for all other view classes.

The view codes correspond to all the models and each model view includes classes that enable the display attributes for Create, Delete, Edit, Details and Index.

4.3 Controllers

Controllers contain application logic that determines the behaviour of the application. For each sub-directory in the URLs, there is a corresponding controller. The E-health application has six controllers, namely:

- AccountController – control the authorization and manage accounts
- Doctors Controller – Enables the creation, edit view, etc of Doctors
- Home Controller – Controls the home page logic. This includes the view of home page attributes.
- Hospitals Controller - Enables the creation, edit view, etc of Doctors
- Patients Controller – defines the creation of new patients' profiles, add patient information and all patient-related logic matters.
- Prescriptions Controller – deals with the logic of adding prescription details to patients. (this contains endpoints for communication with API)

The EDT controllers include the Account controller, the Hospital controllers, the Home controllers, the Medicine and Pharmacy controllers.
Below is an example of a controller code for prescriptions. This allows the application to get data corresponding to the prescription and create them. The same is done to delete, edit, etc.

```
    // GET: Prescriptions/Create
    public IActionResult Create()
    {
        ViewData["PrescriptionPatientID"] = new SelectList(_context.Patients, "PatientID", "CellphoneNumber");
        return View();
    }

    // POST: Prescriptions/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("PrescriptionID,PrescriptionPatientID,PrescriptionNotes,Diagonosis,DiagonosisBy,PrescriptionCollected,Created_by,Modified_by,Created_on,Modified_on")] Prescription prescription)
    {
        if (ModelState.IsValid)
        {
            _context.Add(prescription);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        ViewData["PrescriptionPatientID"] = new SelectList(_context.Patients, "PatientID", "CellphoneNumber", prescription.PrescriptionPatientID);
        return View(prescription);
    }
```

The logic in these controllers is activated when their corresponding sub-directory is browsed by the user. For example, the logic in the Account Controller is run when a user navigates to /account in their browser.

In each controller, there are methods for each endpoint in the URL. For example, when a user navigates to /hospitals/edit/5, the edit method below is run.

```
public async Task<IActionResult> Edit(int? id) {
```

```
        if (id == null) {
            return NotFound();
        }
        var hospital = await _context.Hospitals.FindAsync(id);
        if (hospital == null) {
            return NotFound();
        }
        return View(hospital);
}
```

In this Edit method, there is a parameter that is passed on. This is the ID of the hospital to edit. When no ID is passed into the Edit method or the Hospital Object with passed in ID cannot be found, Not Found method is called, and a 404 page is returned instead. If a hospital is found with the passed in ID, the corresponding Edit View is shown to the user with the hospital object to be displayed.

4.3.1.1 Account Controllers

To enable security in the web application, the account controllers for both applications are set up. The purpose of this function is to manage the identity membership on the applications with emails (user names also) and passwords. Below is an account controllers code that enables user authentication and also triggers account lockout.

```
[HttpGet]
    [AllowAnonymous]
    public async Task<IActionResult> Login(string returnUrl = null)
    {
        await HttpContext.SignOutAsync(IdentityConstants.ExternalScheme);

        ViewData["ReturnUrl"] = returnUrl;
        return View();
    }

    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null)
    {
        ViewData["ReturnUrl"] = returnUrl;
```

```
        if (ModelState.IsValid)
        {
            var result = await _signInManager.PasswordSignInAsync(model.Email, model.Password,
model.RememberMe, lockoutOnFailure: false);
            if (result.Succeeded)
            {
                _logger.LogInformation("User logged in.");
                return RedirectToLocal(returnUrl);
            }
            if (result.RequiresTwoFactor)
            {
                return RedirectToAction(nameof(LoginWith2fa), new { returnUrl, model.RememberMe
});
            }
            if (result.IsLockedOut)
            {
                _logger.LogWarning("User account locked out.");
                return RedirectToAction(nameof(Lockout));
            }
            else
            {
                ModelState.AddModelError(string.Empty, "Invalid login attempt.");
                return View(model);
            }
        }
```

The code above trigger account lockout for anonymous loggers and authorizes authenticated users. The Account Controller is decorated by the [Authorize] feature. This restricts most of the controller's actions; the application will be limited only to authorized users. All the attributes above that are decorated by [AllowAnonymous] are availed to all types of users are unauthorized and unauthenticated. These are for example the POST and GET requests that represent attributes for Logins and registering. The code also allows password matching to enable user lockout in case of matching failure.

4.3.2 Database Implementation.

Before deploying Web applications, the database engines have to be connected. This project uses the Microsoft SQL Server Management Studio. Before creating the database tables, the SQL server must

be configurated using the configuration manager. This is for deciding which instances should be running and which should not. Figure 20 shows a snippet of the SQL server enabled and running.

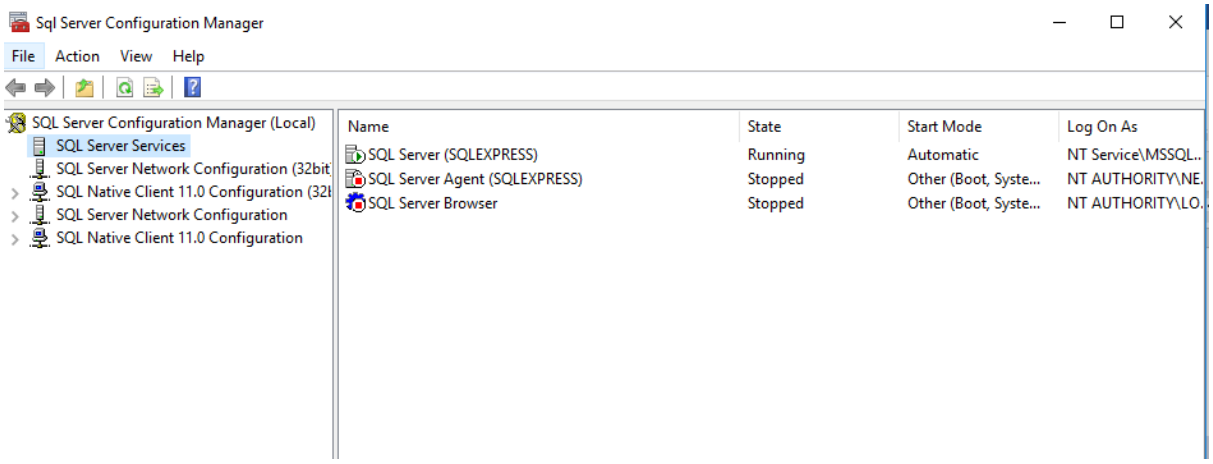


Figure 20: SQL Configuration Manager with the SQL server running.

To connect to the database engine to the server the configurations have to correspond with the host, which in this case is local hence to configure the local server then the server name will correspond with the device name: DESKTOP-288GRV3. Figure 21 shows the connection with server authentication. Once it is connected the database can now be created.

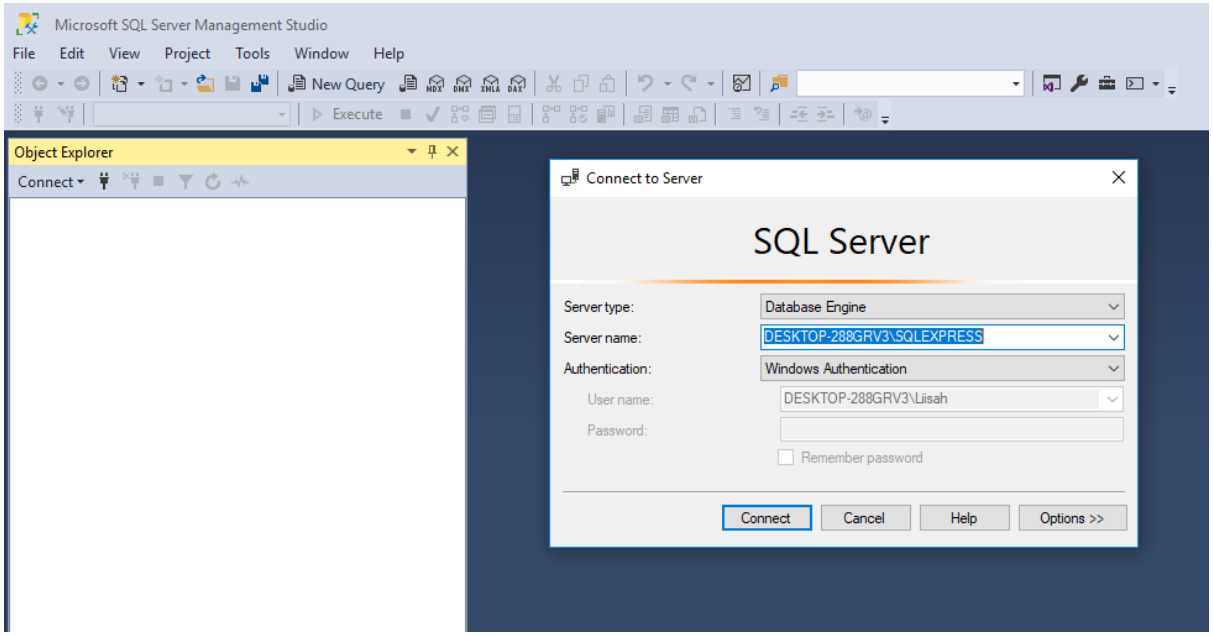


Figure 21: Server connection

Table Migrations

The database is created using the code first approach migration. This method is explained in chapter 3 and it is chosen because it keeps the database in sync with the web application code in the data model. Every time a change is applied to the model the database gets updated. This way the changes are kept intact.

To create the database, first in the Microsoft Visual Studio where the system is developed, from tools the package Manager console is opened as shown in Figure 22. The package manager console is used to enable the migration of the database tables and update.

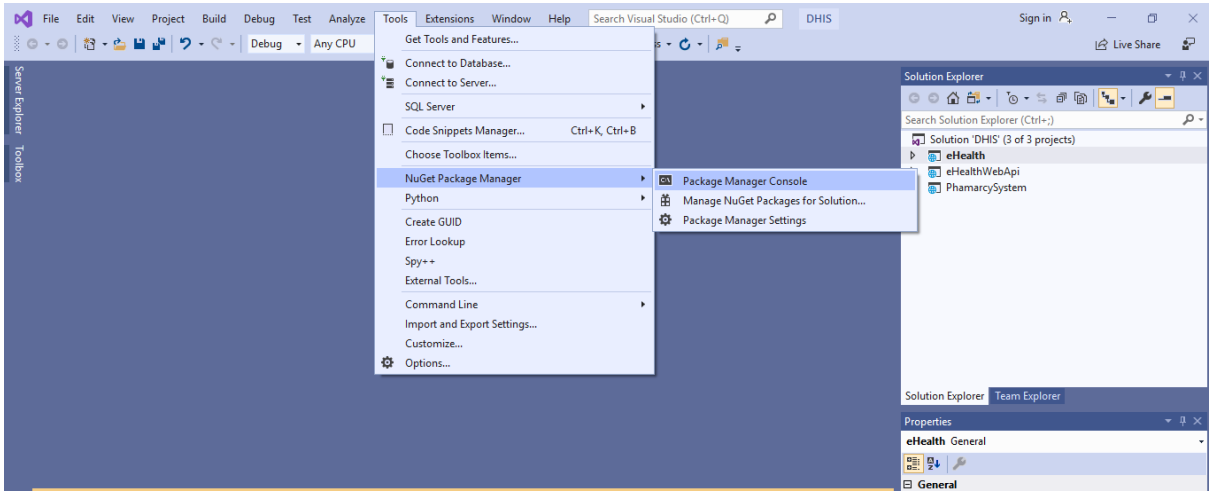


Figure 22: Procedure to opening the console for table migration

Once the console is open the add-migration command is run as shown in Figure 23. Once it is executed the code is generated to create the database tables.

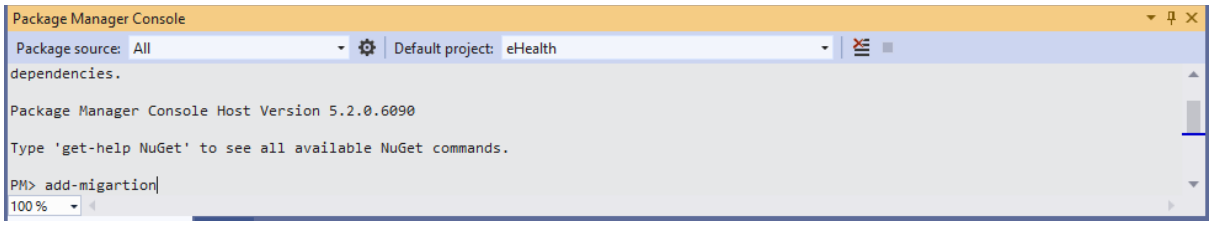


Figure 23 Table creation command

Figure 24 shows the console after a successful migration, for the tables to appear on the SSMS database engine the *"update table"* command is run on the package manage console.

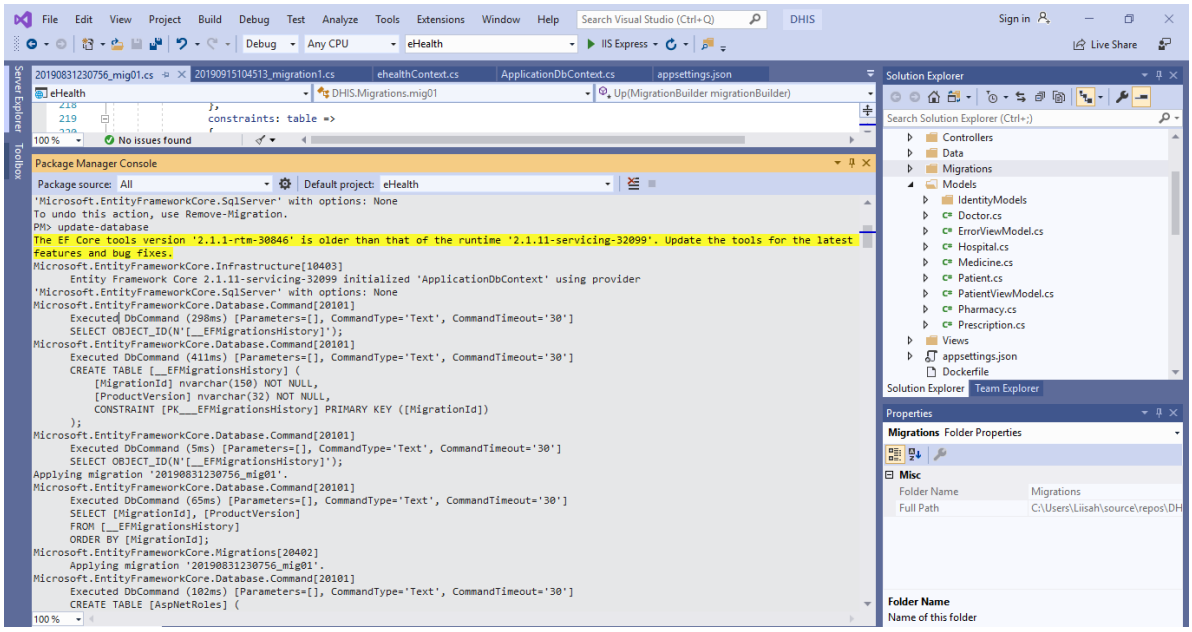


Figure 24: After a successful migration

The tables will only be created after the *"update table"* command is run. Figure 22 shows how the studio looks like before the tables are created and figure shows after the tables are created and updated in the Microsoft Visual studio. The update table command will not only generate the tables but also fill them. This will happen automatically after the application is deployed

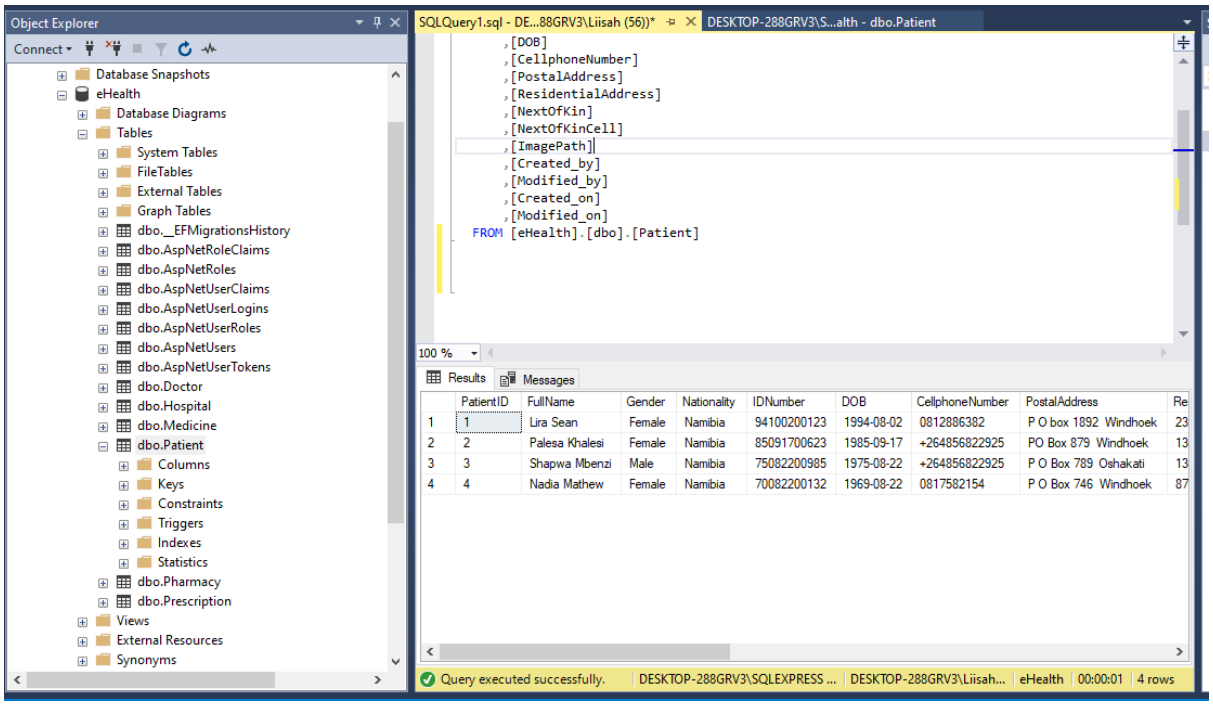


Figure 25: A query made after table migration and update.

After a proper and successful table migration and update, the database is fully created. When the web application is deployed, data can be added. Figure 25 shows a query done after a successful migration and data have been entered. The tables for patients can be observed with its fields and records.

4.4 API Implementation

The Web API was implemented to allow authorized users to securely connect to the central database. The API provides standard methods for data access that can be consumed by various other systems. Authorized users (or applications that consume API) are firstly authenticated, then issued a session token by the API. Subsequent requests to the API can then be sent from the client to API with this session token so that the application can be granted access to the data being requested.

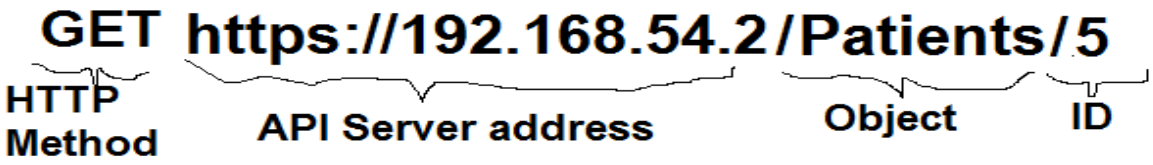
4.4.1 Security

To prevent man-in-the-middle attacks and also ensure data integrity between API and client, the API redirects all HTTP requests to HTTPS. In the startup.cs file, the Configure method, where the HTTP request pipeline is set up, we configure the application to use HTTPS Redirection middleware. The middleware is set up with the command **(app.UseHttpsRedirection)**.

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env) {
        if (env.IsDevelopment()) {
            app.UseDeveloperExceptionPage();
        } else {
            app.UseHsts();
        }
        app.UseHttpsRedirection();
        app.UseMvc();
    }
```

4.4.2 API Endpoints

The API has several methods for external systems to use for reading, creating and modifying data. The Restful API was built with GET, PUT, PATCH, DELETE and POST methods for each data object.



The API endpoints are in intuitive format. When sending a request to the API, the HTTP method needs to be provided, then the server address (or domain when the application is finally deployed), Object or data model (Patients, Prescriptions, etc) and then an additional data which in this application is usually an ID for a record. To implement such an API endpoint ID in C#, the following example of code was written in the Patients Controller class:

```
[HttpGet("{id}")]
    public async Task<IActionResult> GetPatient([FromRoute] int id) {
      if (!ModelState.IsValid) {
        return BadRequest(ModelState);
      }
      var patient = await _context.Patient.FindAsync(id);
      if (patient == null) {
        return NotFound();
      }
      return Ok(patient);
    }
```

NotFound, Ok and BadReqests are built in .Net Framework HTTP Response methods that reply to the client with appropriate 404, 200 and 400 status codes respectively. The Ok method takes an optional parameter which is data sent back to the client. In this case, for the GET patient request, we are sending back a Patient object with an ID specified in the request. The same applies to all the other endpoints for each data model.

5 Results and Testing

The previous chapter explained how the design of web applications and web APIs are implemented. In this chapter, the implemented designs are evaluated and proof of concept is presented. At the primary stage, API is tested and the web applications are deployed. Once the web applications are deployed smoothly, the analysis and demonstration of results in then presented. The presentation includes demonstration of the patient's records logging, new file creation, new prescriptions can be added and the sharing of information.

5.1 Application performance profiling

To conduct the performance profiling, the platform used for application development, Visual Studio Community edition contains a built-in performance profiler to measure the application performance while the application is running . This is done to make sure the application is running rapidly and responding perfectly in order to retain the consumers interests. The performance test focused on ensuring there are no possible bottlenecks that may hinder the application performance. The bottlenecks metrics conducted were to acquire information on how the application is using the devices memory and CPU resources.

Figure 26 shows a snapshot on a test analysis done for the CPU usage for a diagnostic session of around 4 minutes. The session includes both idle and interactive sessions when a user (doctor) is entering patient details and when a user is prescribing medicine. The two session are considered to the targeted application usage. The test result can than be used for CPU optimization consideration.

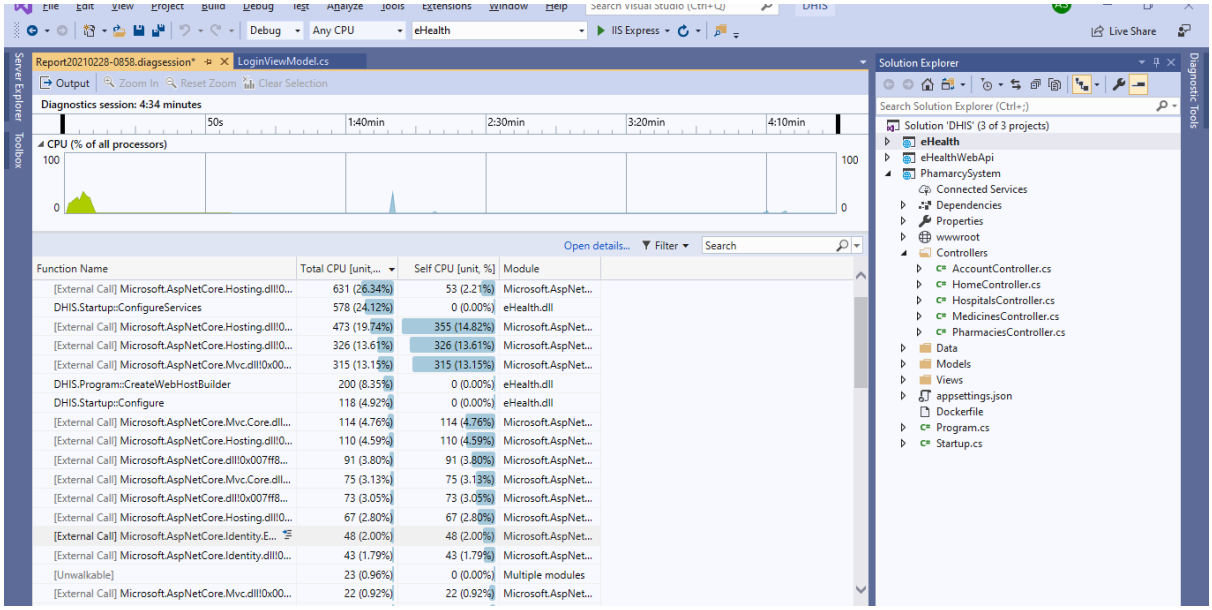


Figure 26 CPU usage results

From the graph in Figure 26 the spikes in CPU usages can be observed and drop down shows the active functions sorted by the size in bytes. This will help identifying function where the CPU optimization is required. The CPU usage is recorded at 2% for the idle session at which is very reasonable as it does not incur unnecessary CPU usage when not in use. The highest usage us

recorded at 27% for the data capturing session which is also considered to be a reasonable CPU usage to process the data capturing requests.

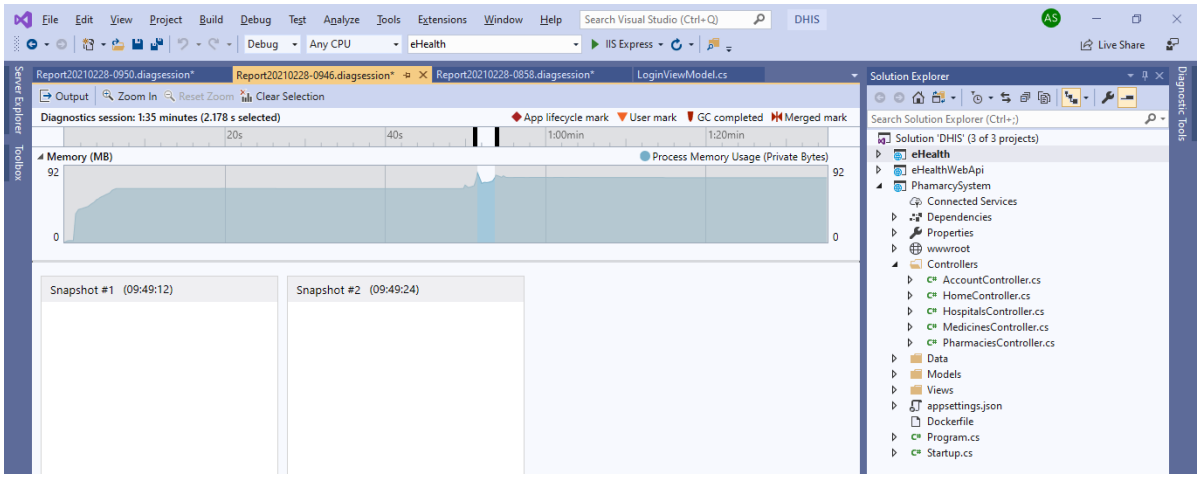


Figure 27: RAM diagnostic results

Figure shows a different snapshot on memory usage in Mega Bytes when processing requests. The diagnostic session of 3.35 minutes for active sessions of data capturing and prescription issuing. The session results indicate a usage of a maximum of 92 MB. This will help with setting breakpoints for the application and identifying the right hardware resources.

5.2 Integration Test

The Web application objectives include functioning properly while communicating with other components in the system such as the database and most important the API. The API is created to access the Prescription model only (the only authorized data) through the patients' identification number. The integration test was performed by testing the APIs' connectivity and response with the server.

To test the API connection, third party tool, a software Swagger is used as explained in chapter 5.1. A manual test is done for demonstration purposes. For robust and larger application, automatic testing is recommended. In this thesis, an End to end testing is performed. The API works with two models which are the Patients ID and Prescription as illustrated in Figure 28.

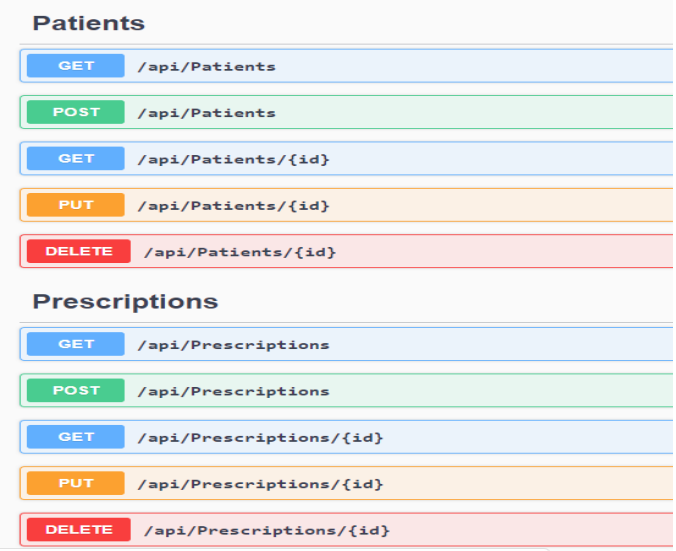


Figure 28: API endpoints with their requests.

To test the APIs' connection to the server, one endpoint is selected and send a request method. An example is shown in Figure 29, it illustrates a GET request done manually by indicating the get response requirement which is a patient ID. The URL in the GET request is executed and the server responds with status code. In this request, one can either get a NotFound, Ok or BadReqests response. In the first attempt, a test was made with a known ID number taken from the E-health application to verify if the response would correspond.

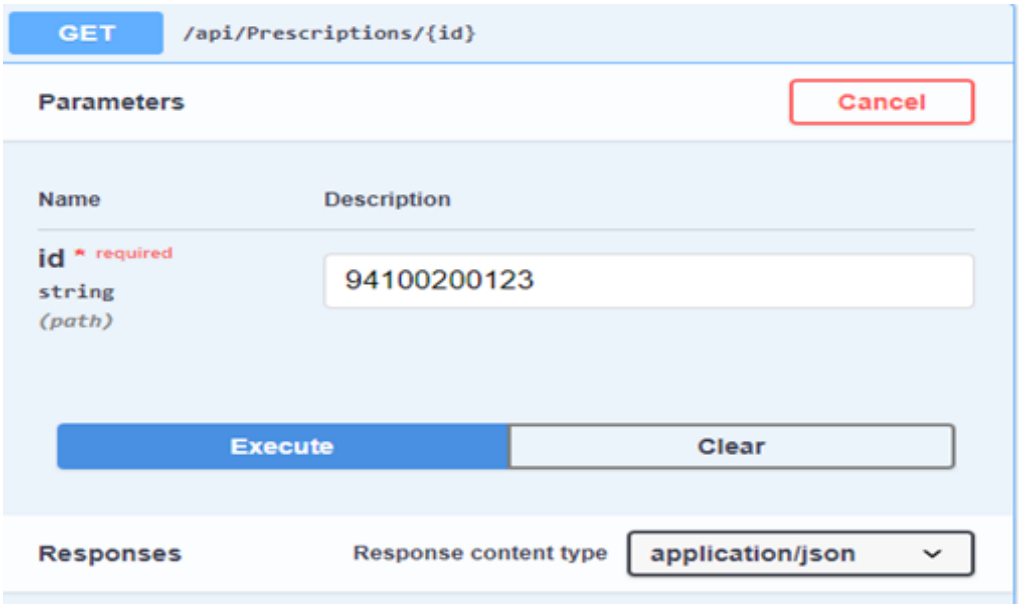


Figure 29 SWAGGER GET request.

Figure 30 shows a response display after executing a GET prescription for the indicated patients' ID number. In the server response, one can than see the status of the connection between the two applications. If the response come with a status code of 200 that represent an OK, the connection is made and the request have returned all the authorised attributes in the patients table. The prescriptions details are in the response body.

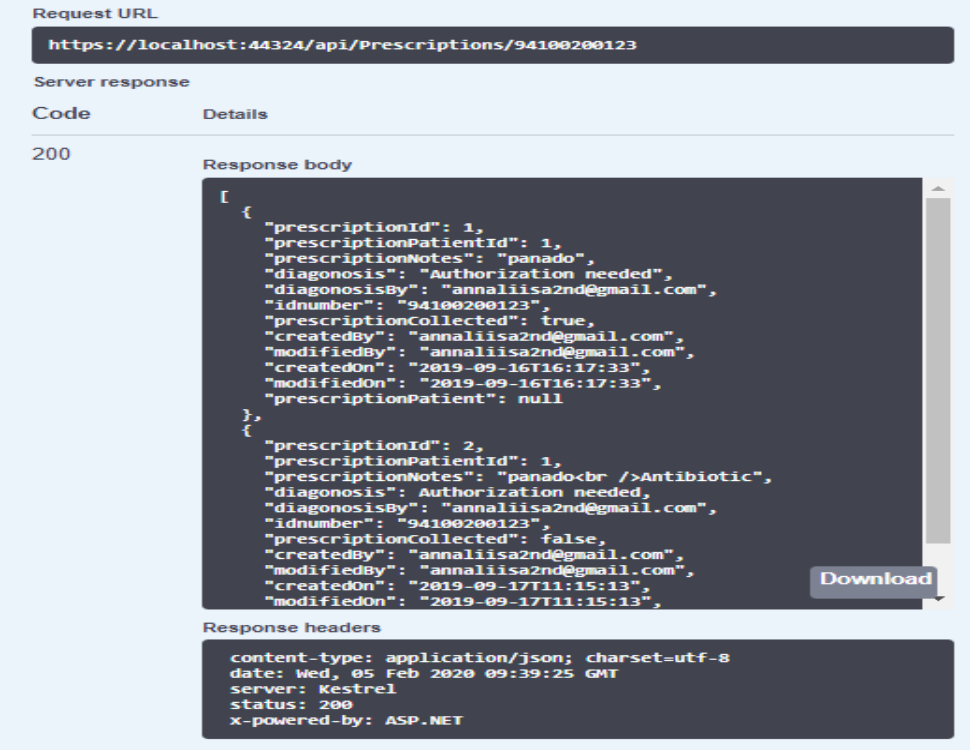


Figure 30: SWAGGER GET response.

To validate concept, a different delete request was performed with a random ID number. The connection was established with the server but it could not find what was requested, hence the 404-error message.

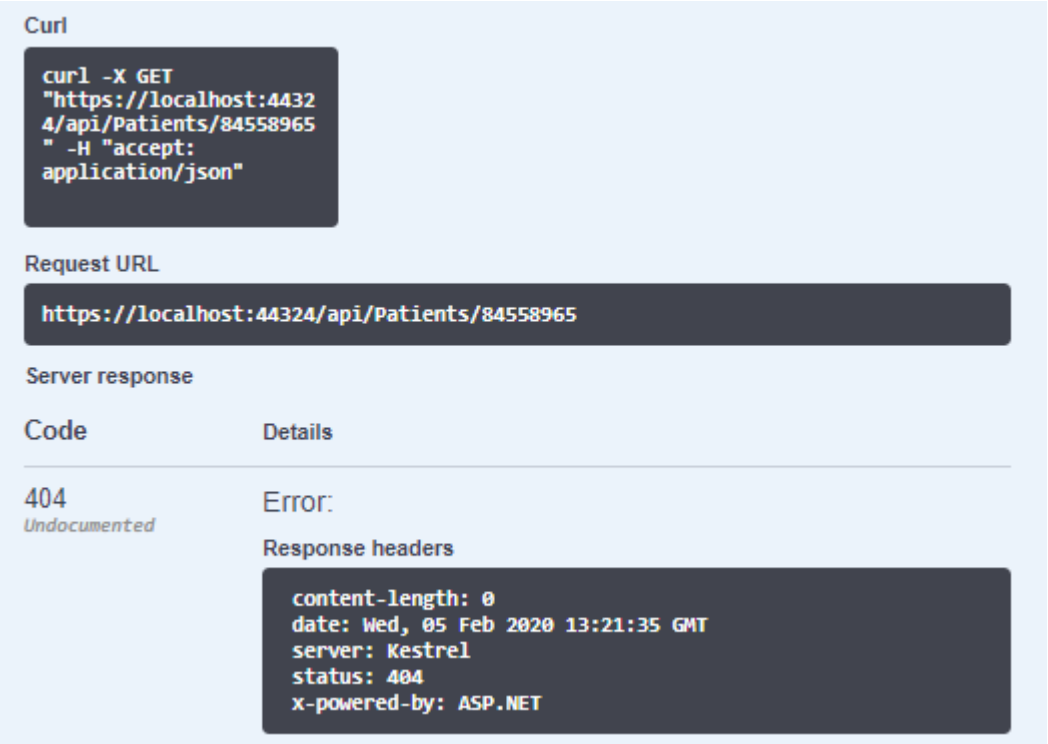


Figure 31: GET request not found.

Figure 31 shows a server response with a 404-error response message. Observation can be made that the response is returned without a response body.

5.3 Deployment:

In the deployment phase before running the application, the SQL Server Configuration Manager has to be launched (previously installed and configured) and the first database table migrations done as explained in Chapter 5. (the tables will update automatically only the first migration is required).

All the web applications (the e-health and Electronic Dispersal system) are first built and run from the Microsoft Visual Studio with IIS Express (Internet Information Services) which is the Web server that powers window servers. The port is then assigned for local hosting deployment (free and accustomed) and Google Chrome as a browser (Other browsers can also be configurated). The web page will open in Google Chrome with a user authentication page as the first point of contact.

5.4 Authentication

The prototype systems constrained to the features to be used in the integration demonstration. The user interface firstly allows authorised users to authenticate for data integrity and security reasons. The user authentication interface is shown in Figure 32. This serve to restrict access.

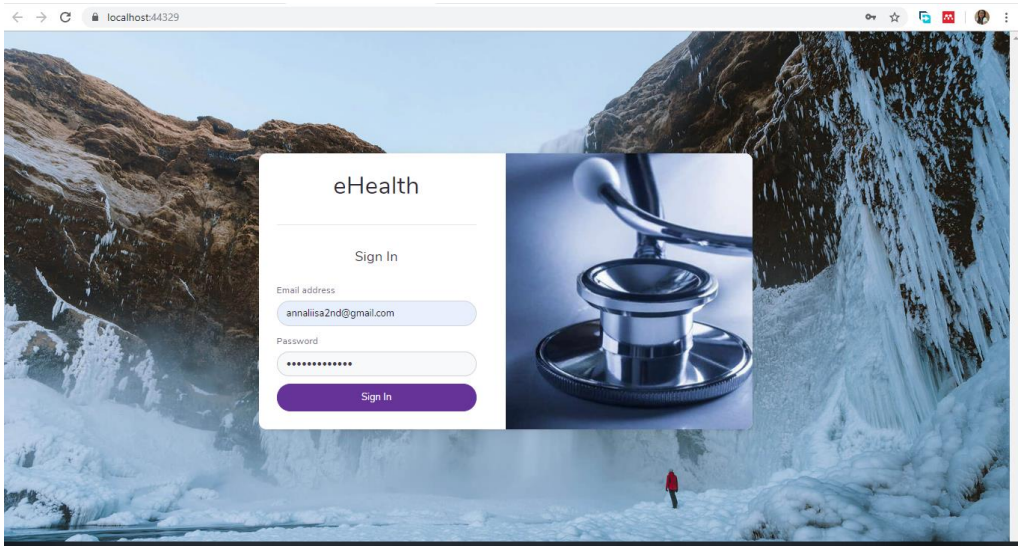


Figure 32 The user interface for authentication corresponding to the E-health web application.

For security and privacy reason, the access restriction is also applied in the EDT application; Figure 33 shows the authentication window to log in authorised personnel.

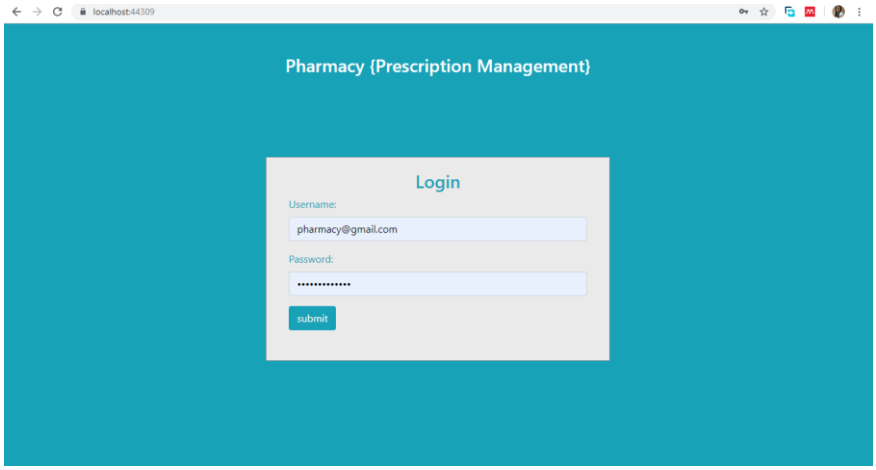


Figure 33: Dispersal tool authentication interface

Scenario 1

If an unregistered user attempt to log in the application, he will be denied access as illustrated in Figure 34. The registered user login will however be grant access and lead to the applications homepages.

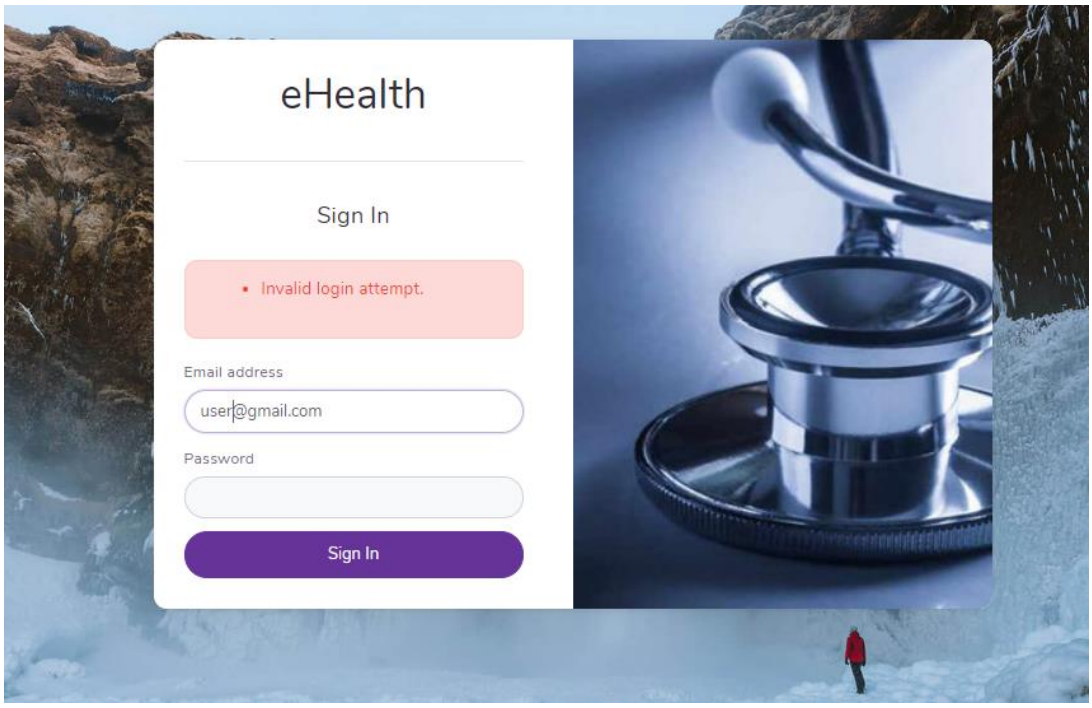


Figure 34: Unauthorised user lockout.

5.5 Data entry and Patient Record Management.

Before the data entry is done, the system administrator first creates system users (authorized hospital staff). The system administrator also has the right to revoke user rights. The main data entry point is then done by the system users as patients come for consultation and the autonomy of the system increase as per the number of patients and system users.

Initially, the e-health system allows the capturing of data as it is. One of the main aims of the e-health is to keep a medical record of all the patient that visit the hospital. The application keeps a record of the prescription and diagnosis but for the record-keeping purpose only. It also generates reports on the record of patients and their prescriptions. The integrated system using the designed API allows the E-health system to receive the prescription collection feedback from the Electronic dispersal system. The data entry is only done at the E-health application which enable the single data entry requirement. This is illustrated in Figure 35.

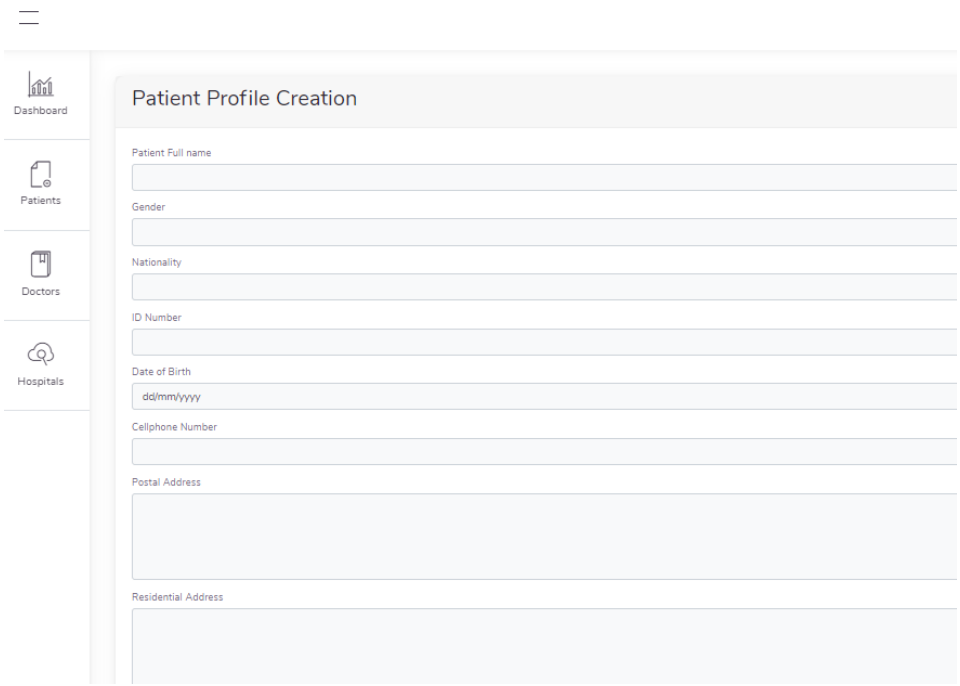


Figure 35: Interface to create patients' profiles.

The system as a medical health record system allows doctors upon patient first consultation to be the first entry-level of the patient demographic information (Name, gender, ID among others), diagnosis and prescription. Figure 36 show the patients list for the previous registered and the search button allow the user to filter among the number of registered patients, the search can only be done by ID as it's the only thing that uniquely identifies patients.

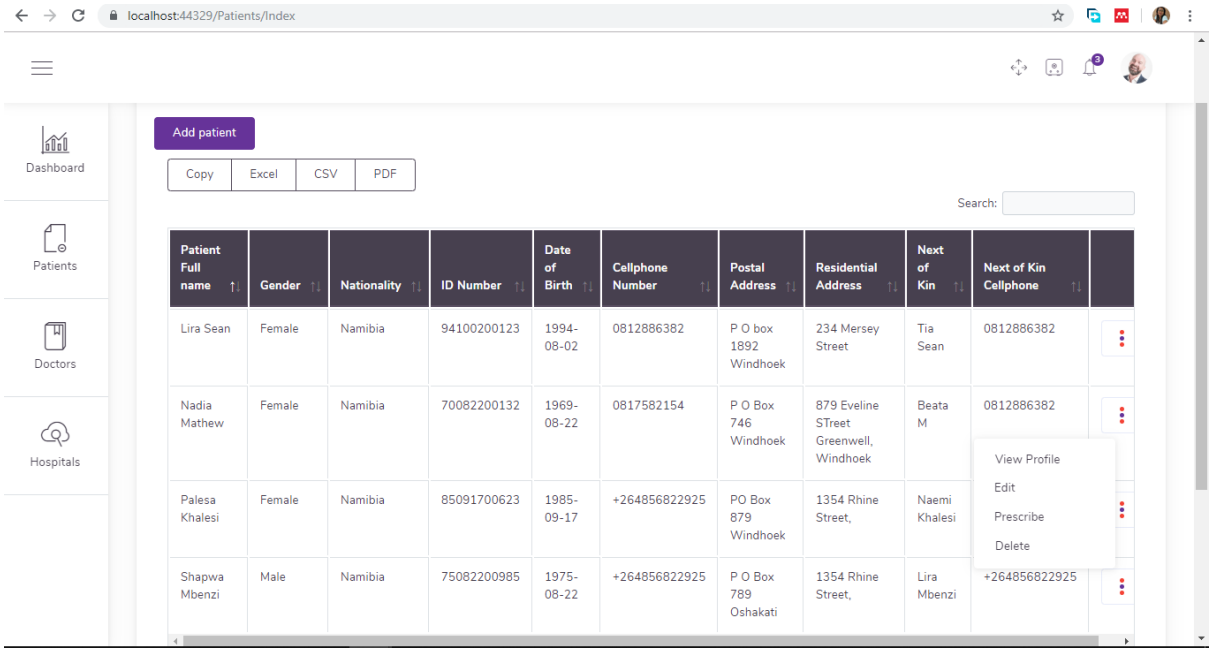


Figure 36: Interface to view patients list and add new profiles.

The three dots allow the user to extend and view the patient profile as illustrated in for more information on the previous description and patient medical history, or to edit the patient details, to write down the patient's diagnosis and prescriptions and lastly to delete if necessary.

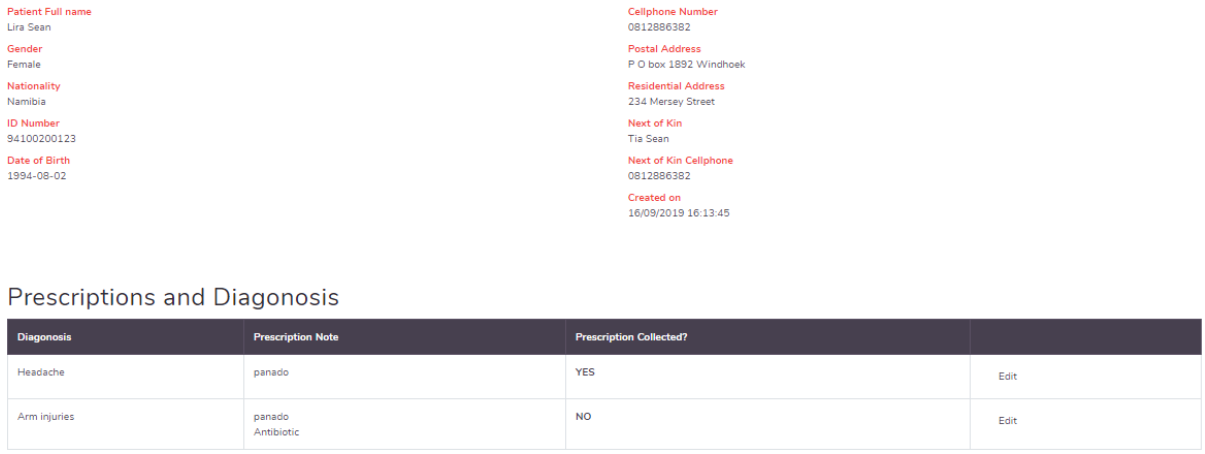


Figure 37: Patient profile example.

Patient record management, whereby a record of all the patients together with their medical history, demographic information, etc. is kept electronically other than the booklets that are still currently in use in most of the hospitals in Namibia especially in the rural areas.

5.5.1 Descriptions and Diagnosis

Upon patient registration by the authorized personnel follows the record of patient diagnosis and prescription. A patient can only be registered once, their national ID serves as their unique identification and does not allow another entry into the system. To add more diagnoses to an already registered patient, the user tap on the patients' profile and select diagnose and prescribe. An interface for the diagnosis will appear as shown in Figure 38.

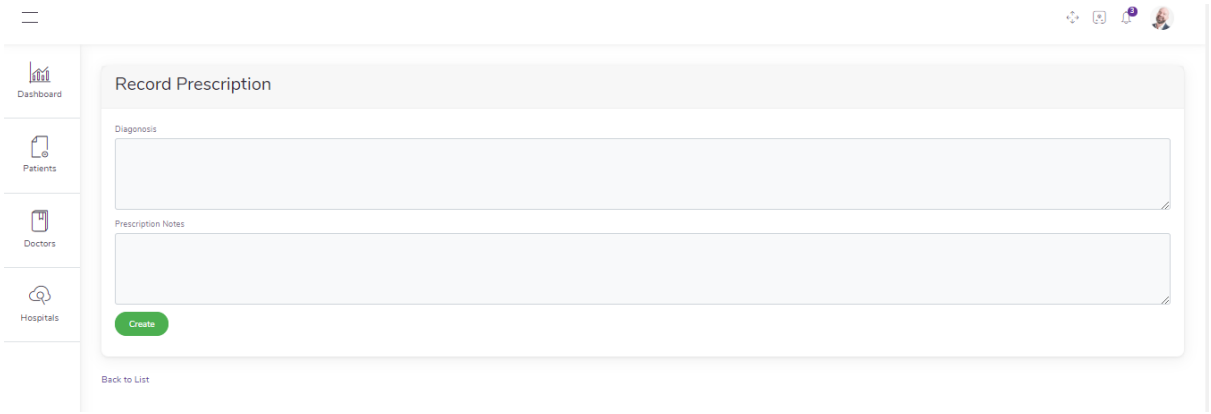


Figure 38: Interface for adding patient's prescription and diagnosis.

Once the patient is registerd, they are captured in the system permanently and everytime they go for a conultation their medical history increases. Figure 39 show patient profile with multiple diagonisis and their perspective perscriptions.

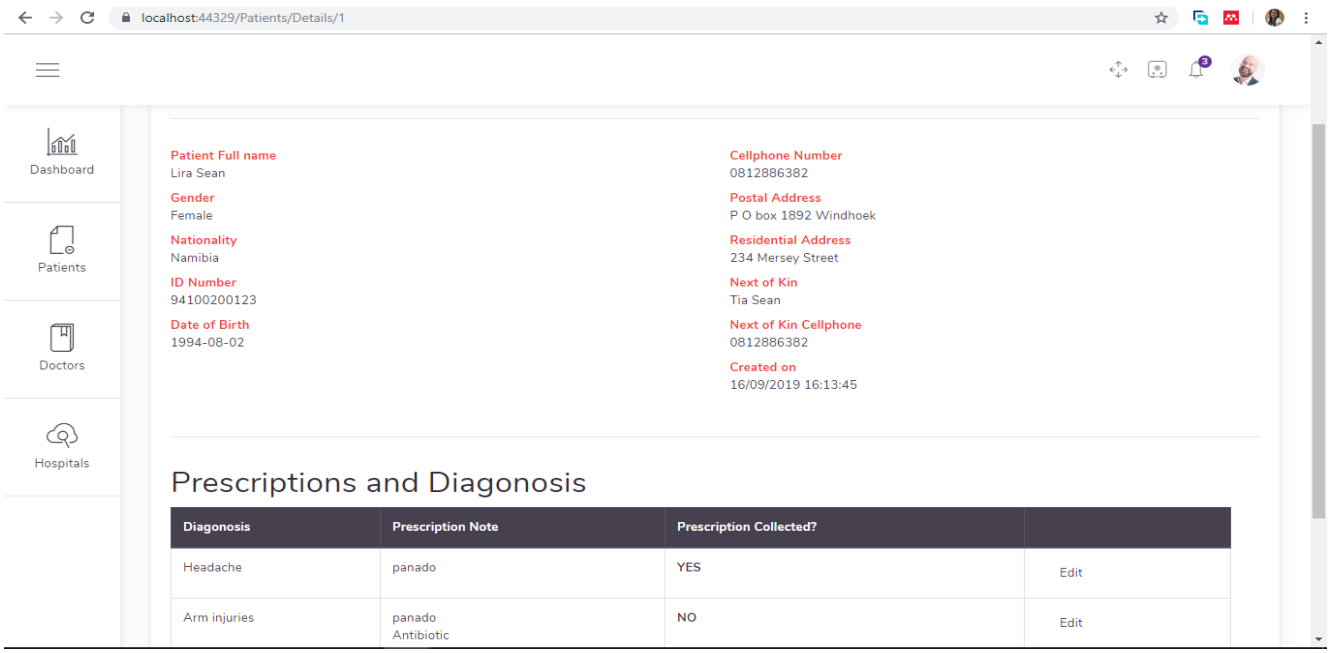


Figure 39: An example of the patient profile

5.5.2 Medicine Dispersal

The pharmaceutical prototype like the e-health system is simplified for demonstration purposes. The two are independent systems run in the hospital for different operations. The replicated systems run on different servers and with different databases. Its functions are discussed in the literature review of the Pharmaceutical Management Information System in Chapter 2.2.2. The prototype designed is only restricted to the area of interest for integration which is the prescription management unit, the Electronic Dispensing tool. The prototype does not have its own database as it opposes the purpose of the integration demonstration that as of the aims is to remove data redundancy. The tool is part of a complete Pharmaceutical Management Information System that connects all the pharmacies and their inventory management. The get prescription section allows the pharmacist to search for pending prescription using the patient's unique identification number. Figure 40 shows the interface for prescription searching and dispensing.

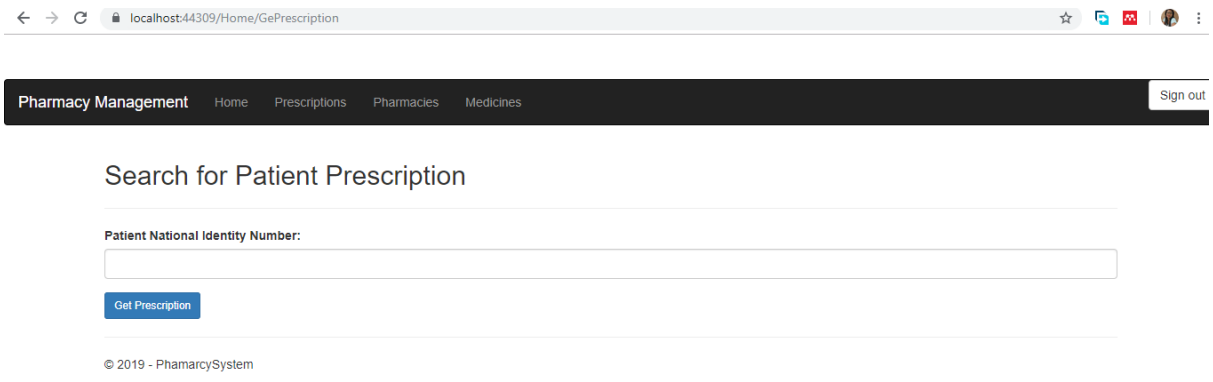


Figure 40: Get prescription controller button.

Scenario 2

If an unregistered patient ID is entered, the API query will return an errored response as the request cannot be processed. The illustration is given in Figure 41 for demonstration.

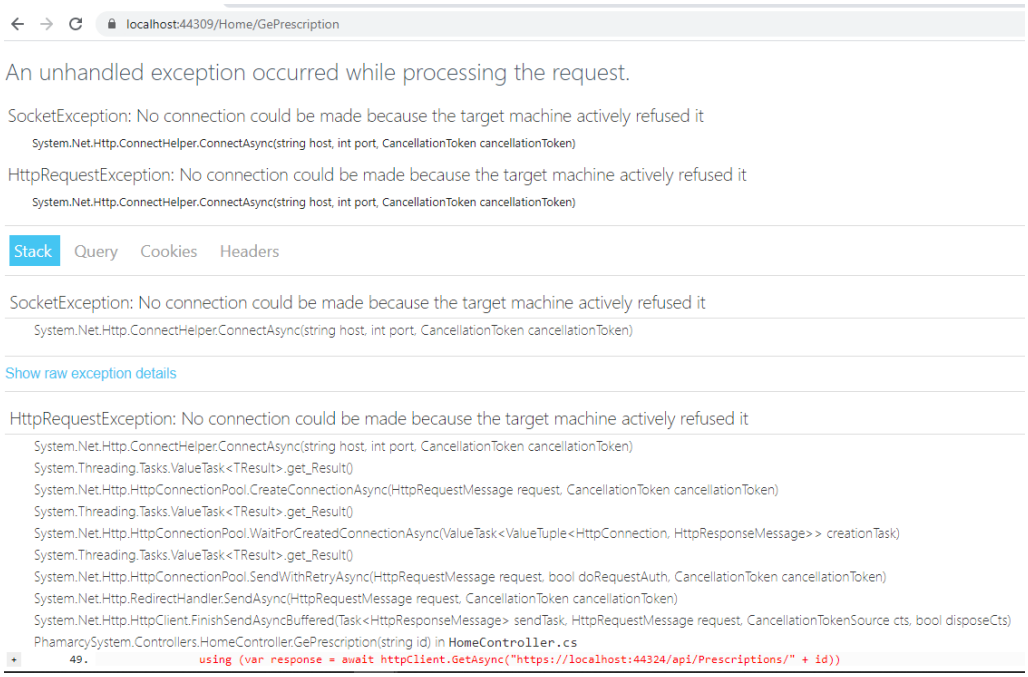


Figure 41: Unregistered patient query response.

Once the correct ID is entered, the API will connect the EDT to E-health system and fetch the information corresponding to the entered patient ID. Figure 42 show the details of an example of patient ID number 94100200123 as observed recorded and diagnosed. The response will include the patient's prescription note.

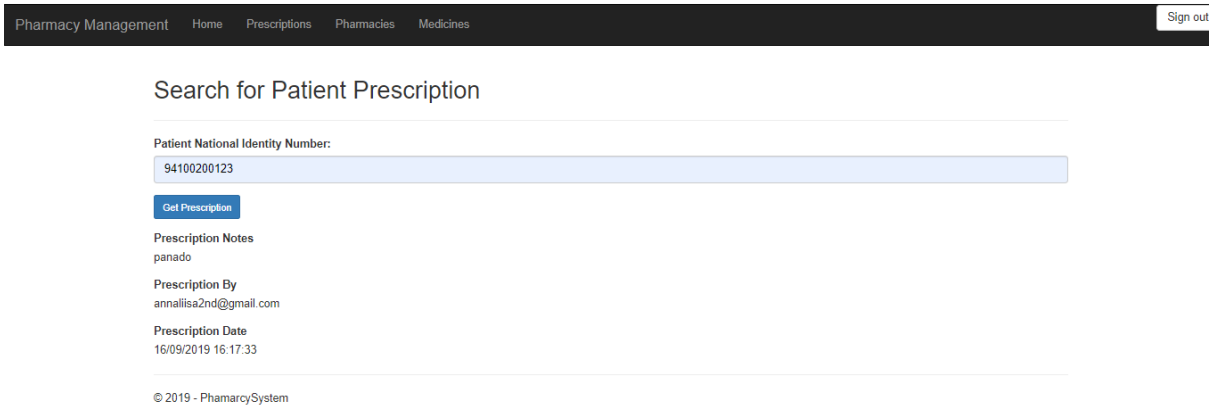


Figure 42: Successful patient prescription search.

5.5.3 Feedback

The feedback interface serves to allow communication from the EDT to the E-Health system. It allows the system E-health system to keep a record of all the prescriptions served and the doctor to know the status patient medicine intake especially in case of a follow up. It is only used in case of acknowledgment button for medicine collection approval feedback. Figure 43 shows the interface for such.

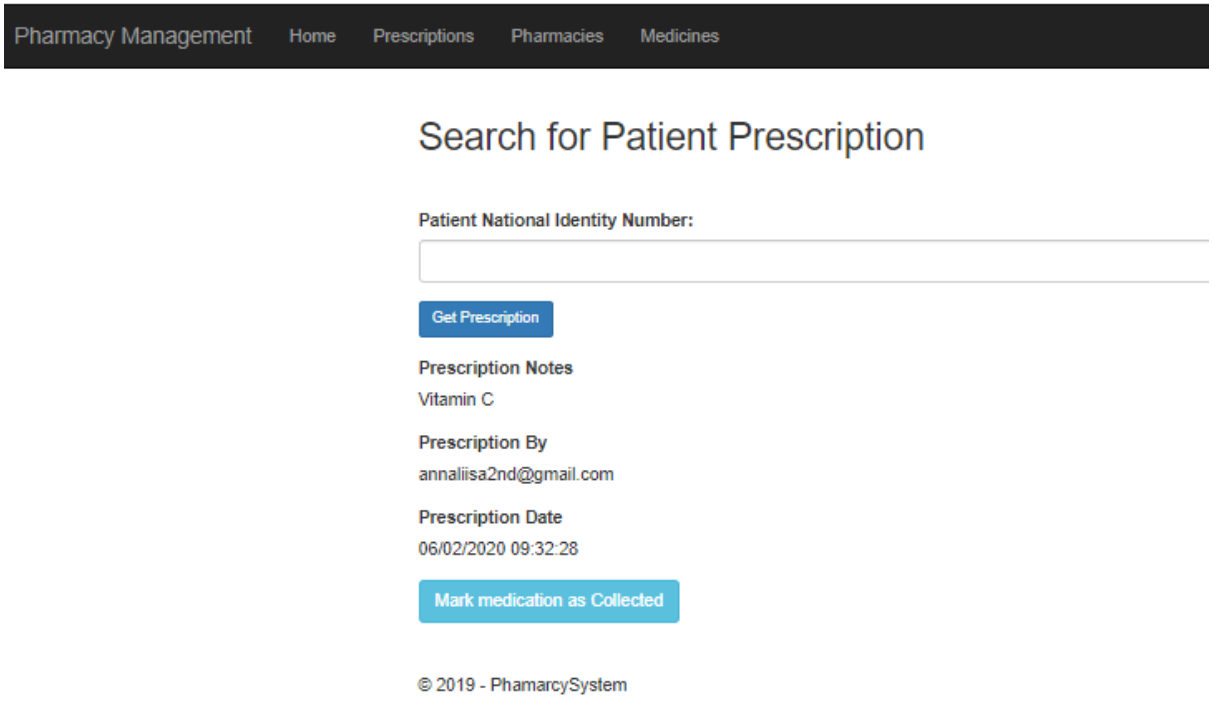


Figure 43: Medicine collection feedback button.

Scenario 3

In case the pharmacist did not send the medicine collection approval, the patient record file will indicate medicine collection pending collection, given that the patient has collected their prescription, their files will show prescription collected.

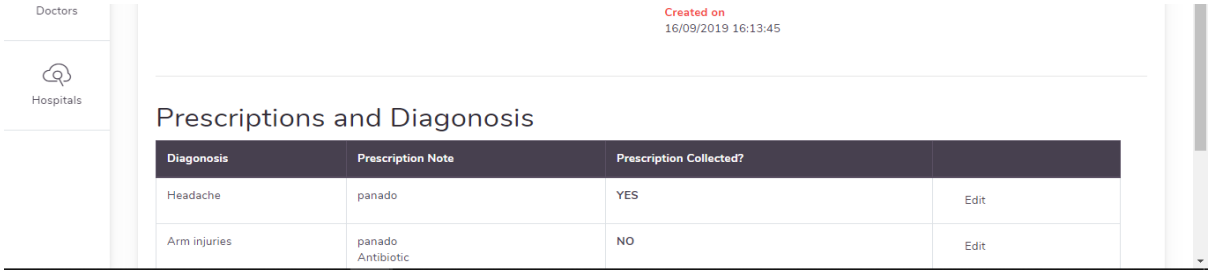


Figure 44: Prescription collection feedback.

5.5.4 Dashboards

The participant interface to view all registered patients and users

The dashboard allows the system to generate statistics on the number of patients registered, their prescription collected as well as pending one. It also shows the number of doctors registered on the system. Details are shown in Figure 45. These reports are meant to be sent to the DIHS2 or as raw data depending on the data format the system supports.

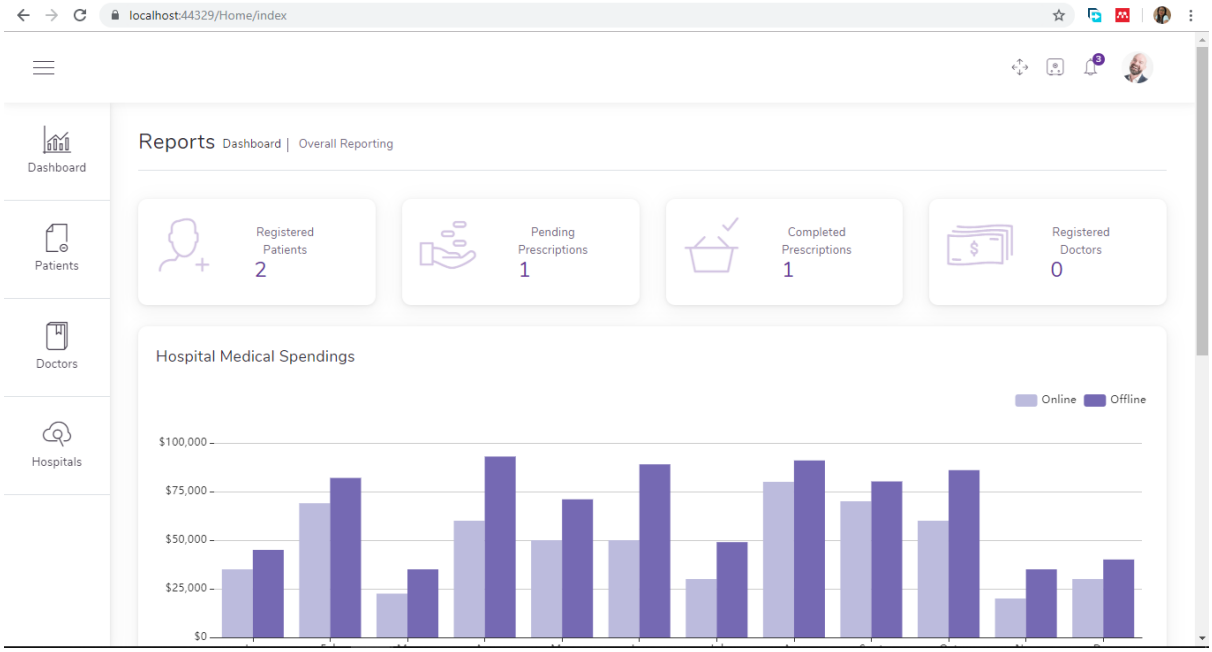


Figure 45: An example of the dashboard created by the E-Health application.

The whole system's idea is to keep a tab of the hospital consultation. The features that are not included in the design include: patient management (day and hospitalised patient,), billing (medical plan cares and aid) patient portal and the main reason for their omission is because they are not related to the integration demonstration and their presence or absence does not impact the current presentation.

Figure 46 shows the home page for the system upon authentication dashboard for statistic used in decision making and stock taking card system all described in the literature review, it includes the amount of the registered patients and information on whether their prescription is collected or not.

The features extend to the list of pharmacies linked and the medicines in stock as part of the mother system although not fully elaborated.

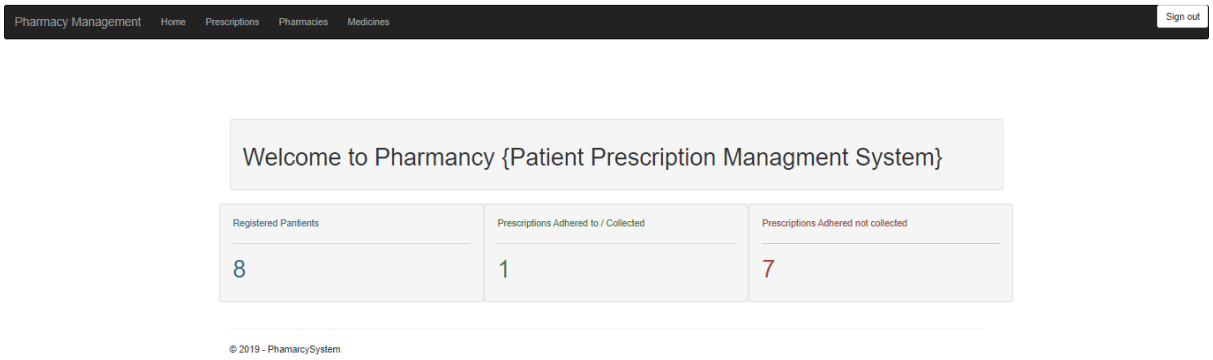


Figure 46: EDT system dashboard.

5.6 User's Feedback

To get insight of the user's point of view, an online survey was conducted using my.survio.com. My survio is a survey web application that allow developers to create web evaluation surveys and view results online. End users are sampled volunteering hospital staff members from both ends of the integrated system. The questionnaire is designed the same for both the data capturers (doctors and nurses) on the e-health application and the pharmacists on the pharmacy system. The link to the questionnaire; https://www.survio.com/survey/d/V5X2A9L5A1Q9F6L9W is shared to the participants to allow them to complete the survey online.

5.6.1 User interface:

The first questions 1 &2 of the questionnaire were designed with multiple choice rating scale to see if the web application is user friendly. Question on ease of use: "How easy is it to navigate the web application?" and "How visually appealing is the web application?" were used. Figure 47 and Figure 48 shows the snapshots from the survey results of the user answers for user interface questions. More than 80% of the users find the application very easy to extremely easy to use and moderately appealing to very appealing.

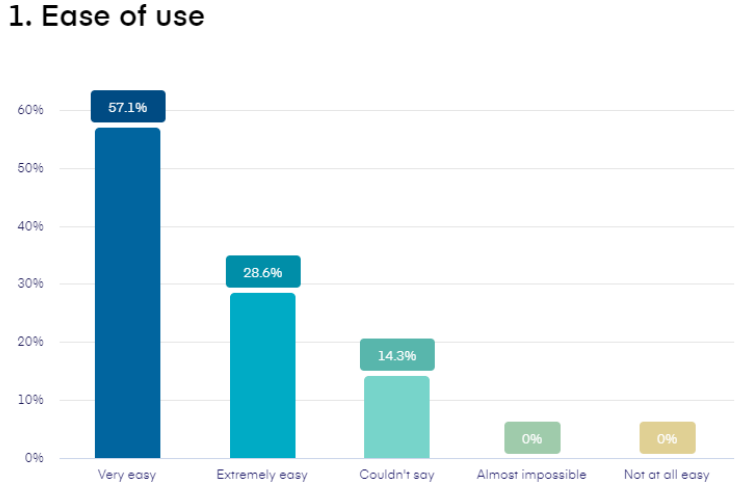


Figure 47:Ease of use results

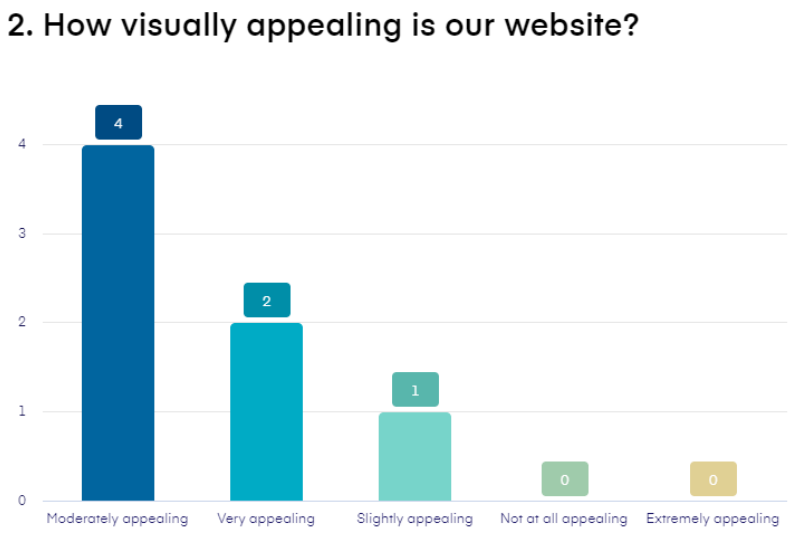
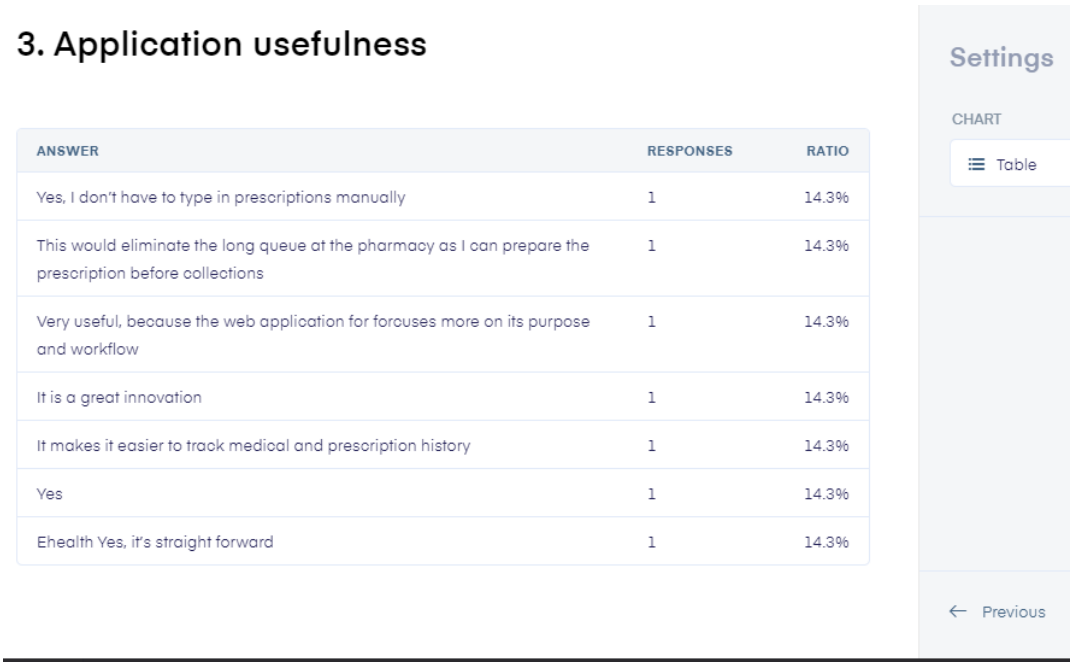


Figure 48: Application presentation results

5.6.2 Usefulness

This question is designed to see if the application is meeting the targeted utility and if the system really improves the user's service delivery. Question used on application usefulness: "did you find the application relevant to the circumstances?" Results of survey answers can be observed in Figure 49. Users contributed positively on how the integration impacted their day today activities.



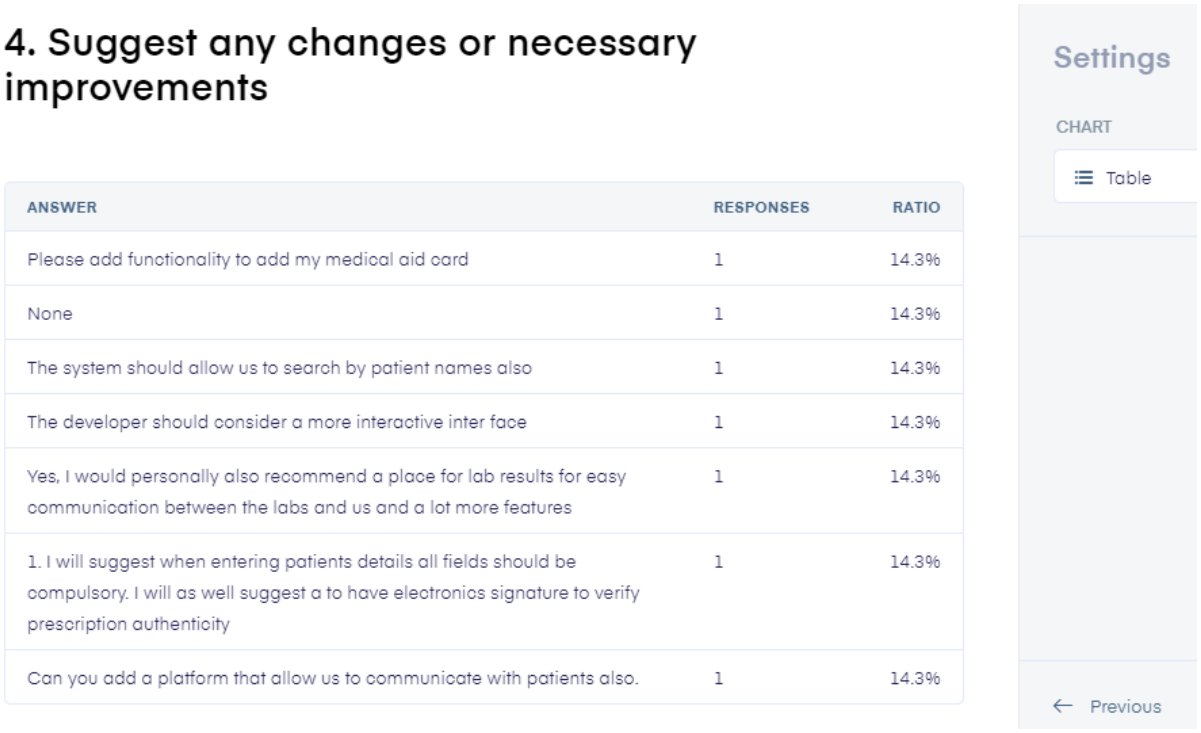
ANSWER	RESPONSES	RATIO
Yes, I don't have to type in prescriptions manually	1	14.3%
This would eliminate the long queue at the pharmacy as I can prepare the prescription before collections	1	14.3%
Very useful, because the web application for forcuses more on its purpose and workflow	1	14.3%
It is a great innovation	1	14.3%
It makes it easier to track medical and prescription history	1	14.3%
Yes	1	14.3%
Ehealth Yes, it's straight forward	1	14.3%

Figure 49:Application usefulness results

5.6.3 User's recommendations

This evaluation allows users to air their suggestion on any improvements and additional features they think are crucial for the system. Question used: " Suggest any changes or necessary improvements" .

Results are observed in Figure 50. Users suggestions are focused on further integration to be done as well with other departments such as laboratory services and new functionalities to be added to the system.



4. Suggest any changes or necessary improvements

ANSWER	RESPONSES	RATIO
Please add functionality to add my medical aid card	1	14.3%
None	1	14.3%
The system should allow us to search by patient names also	1	14.3%
The developer should consider a more interactive inter face	1	14.3%
Yes, I would personally also recommend a place for lab results for easy communication between the labs and us and a lot more features	1	14.3%
1. I will suggest when entering patients details all fields should be compulsory. I will as well suggest a to have electronics signature to verify prescription authenticity	1	14.3%
Can you add a platform that allow us to communicate with patients also.	1	14.3%

Figure 50: User's recommendation results

Chapter 7

6 Conclusion

This research aimed to demonstrate the possibility of having standalone systems that operate with same data but different functionalities to interoperate. A case study of health information systems from hospitals in Ministry of Health and Social Services in Windhoek was used. Based on a qualitative research conducted, it was observed that patient demographic data entry is first done at the consultation stage. This same data is used at other departments within the hospital where they have to be entered again in other systems. It was concluded that having one single data entry would solve several issues within the facilities such as data redundancy, labour cost, and preserve data integrity.

Following the above observations, the dream of sharing the authorised data would be unrealistic since all the systems in place are independent and cannot communicate. To invent a new health information system that covers all operations of the hospital (laboratories, pharmacies, radiography and consultations all in one) may be costly, excessive and inordinate procedure for the ministry considering the current economic situations. Facilitating hospital operation and delivering an excellent quality of service is however still and remains the goals of the ministry. A challenge to create a solution on system integration and allowing them to interoperate hence sharing necessary information, In Chapter 1, the following objectives were set:

- Design sub-systems from different departments that interoperate in their day to day processes as prototypes health systems
- Model and implement a mechanism to combine the sub-systems into an integrated system.
- Evaluate the integrated system to validate the connectivity between the two subsystems in a proposition of allowing proper information exchange and collective data storage.

Following the given objectives, two prototypes; an Electronic Dispersal Tool from the pharmacy department and E-health system from the doctor's consultation department were used to test theory. The prototypes were designed as web applications together with an API and used to demonstrate the possibility of the flow of data (In this case the doctor's prescriptions are the necessary information the pharmacy system would require) between the two systems.

The E-Health is the mother application that is centred on all the applications. The E-Health application alone can run independently. It serves a purpose of recording, store and process patients' data electronically.

The dispensing tool can run independently, provided that it does not have to tap into the e-health database through the connection enabled by the APIs. The aim is however to remove the problem of data redundancy and have one data entry point which is the E-health system.

To demonstrate the interconnection between systems, the proof of concept is implemented through the frameworks. The results were obtained and characterized in Chapter 5, through the results presentation; it is proved that API can be of great use when it comes to systems integrations. They allow Multi directional exchange of information in independent systems or applications, APIs support the communication in parallel applications where different departments operate different applications independently but to achieve a common goal.

6.1 Recommendations

During the process of conducting the research and design for this research topic, there were various reflections that arises. While the research only focused on the relation between the consultation rooms and pharmacies, there are whole lot of other departments in the hospital with their independent systems. This can be narrowed to the Laboratory and the radiology department, Other operational department like hospital management department that manages inpatients and other specialised departments such as the Cardiology, neonatal, paediatric, surgical departs etc. can be added to the list.

Just like demonstrating the possibility of e-prescribing, the ability to effortlessly share data with other departments electronically, the following attributes can be looked into for further developments.

- ✓ Integration software:

Apart from using API to integrate, there are various commercial enterprises systems that can be used for integration, software frameworks that are custom made for system interoperability, data exchange through parallel communication, and provide necessary environment that are often achieved through end point integration.

- ✓ Big Data

Integrated systems will produce a pool of integrated data and new data storage have to be implemented. Research on how to make use of related data can be done to enable proper usage of data through Artificial Intelligence, Machine to machine communication etc.

- ✓ Integration with the DHIS

The DHIS is considered as the centre of information in the entire Ministry of Health and Social services. As explained in Section 2.2.3, it collects data from different health information system dashboards then processes the data for presentation and decision making. As most of the systems feed the data manually, the integration of the systems dashboards with DHIS could be of great use.

References

[1] E. Nyella, “Challenges in health information systems integration: Zanzibar experience,” *2009 Int. Conf. Inf. Commun. Technol. Dev. ICTD 2009 - Proc.*, pp. 243–251, 2009.

[2] C. T. Nengomasha, R. Abankwah, W. Uutoni, and L. Pazvakawambwa, “Health information systems in Namibia,” *Inf. Learn. Sci.*, vol. 119, no. 7–8, pp. 358–376, 2018.

[3] MoHSS, “Dashboard - Namibia Master Health Facility List.” [Online]. Available: https://mfl.mhss.gov.na/. [Accessed: 25-May-2019].

[4] S. H. Dlodlo N, “The Status of Integration of Health Information Systems in Namibia.,” *Electron. J. Inf. Syst. Eval.*, vol. 20, no. 2, pp. 61–75, 2017.

[5] T. Lippeveld, R. Sauerborn, and C. Bodart, “Design and implementation of health information systems.,” *WHO*, p. 280, 2000.

[6] MoHSS, “STRATEGIC PLAN 2017/2018 – 2021/2022,” 2017.

[7] L. A. Tawalbeh, R. Mehmood, E. Benkhlifa, and H. Song, “Mobile Cloud Computing Model and Big Data Analysis for Healthcare Applications,” *IEEE Access*, vol. 4, pp. 6171–6180, 2016.

[8] N. Sabooniha, D. Toohey, and K. Lee, “An evaluation of hospital information systems integration approaches,” no. August, 2012.

[9] S. Kitsiou, V. Manthou, and M. Vlachopoulou, “A Framework for the evaluation of integration technology approaches in healthcare,” *Proc. Int. Spec. Top. Conf. Inf. Technol. Biomed.*, no. October, pp. 26–28, 2006.

[10] Word Health Organization, “HEALTH INFORMATION SYSTEMS,” 2008.

[11] G. Eysenbach, “What is e-health?,” *J. Med. Internet Res.*, vol. 3, no. 2, p. E20, 2001.

[12] N. SMIT, “Ministry launches E-Health system - The Namibian,” *The Namibian Newspaper*, WIndhoek, 09-Aug-2011.

[13] G. M. W. Tjaronda, E. Sagwa, H. Kagoya, “E-health Innovation Expedites Patient Dispensing Services in Namibia,” *MSH*, 2018. [Online]. Available: https://medium.com/@MSHHealthImpact/e-health-innovation-expedites-patient-dispensing-services-in-namibia-8a8956a42412. [Accessed: 01-Jul-2019].

[14] D. Mabirizi *et al.*, “Implementing an Integrated Pharmaceutical Management Information System for Antiretrovirals and Other Medicines: Lessons From Namibia.,” *Glob. Heal. Sci. Pract.*, vol. 6, no. 4, pp. 723–735, 2018.

[15] H. Kong, D. Feng, W. Fan, and Q. Li, “Design and Implementation of Pharmaceutical Logistics and Supply Chain Management System for Hospital,” no. Ameii, pp. 1149–1152, 2015.

[16] V. Sumbi *et al.*, “Developing and Implementing a Nationwide Electronic Pharmacy Dispensing System in Low-Resource Settings: The Electronic Dispensing Tool (EDT) in Namibia’s ART Program,” 2013.

[17] J. Å. Haugen, G. Hjemås, and O. Poppe, *Manual for the DHIS2 quality tool*. 2017.

[18] R. Dehnavieh *et al.*, “The District Health Information System (DHIS2): A literature review

and meta-synthesis of its strengths and operational challenges based on the experiences of 11 countries," *Health Inf. Manag.*, vol. 48, no. 2, pp. 62–75, 2019.

[19] J. Braa and S. Sahay, "The DHIS2 Open Source Software Platform: Evolution over time and space," *Glob. Heal. Informatics*, no. 6, pp. 67–72, 2017.

[20] K. Bigten, "Health Information System Integration Approaches," 2018. [Online]. Available: https://www.grin.com/document/437199. [Accessed: 19-Jun-2019].

[21] D. Smith, "System Integration and Interoperability - Healthcare Informatics," *Healthcare Informatics*, 2016. [Online]. Available: https://sites.google.com/site/healthcareinformatics456/home/system-integration-and-interoperability. [Accessed: 30-Jul-2019].

[22] K. Viakom, "10 New Requirements for Modern Data Integration - Database Trends and Applications," *Database Trends and Applications*, 2016. [Online]. Available: http://www.dbta.com/Editorial/Trends-and-Applications/10-New-Requirements-for-Modern-Data-Integration-109146.aspx. [Accessed: 29-Jul-2019].

[23] B. Lau, "5 Clinical Information Systems (CIS) every clinic owner should know," *2019 MIMS Pte Ltd*, 2016. [Online]. Available: https://today.mims.com/5-clinical-information-systems--cis--every-clinic-owner-should-know. [Accessed: 26-Jun-2019].

[24] N. R. Dissanayake, K. A. Dias, and S. Lanka, "Web-based Applications : Extending the General Perspective of the Service of Web-based Applications : Extending the General Perspective of the Service of Web," no. March 2018, 2017.

[25] L. Shklar and R. Rosen, "Web Application Architecture: Principles, Protocols and Practices."

[26] S. R. Singh and S. Kumar, "An Overview of World Wide Web Protocol (Hypertext Transfer Protocol and Hypertext Transfer Protocol Secure)," 2016.

[27] D. Raggett, A. Hors, and I. Jacobs, "HTML 4.0 Specification," 1998.

[28] B. Duperrat, N. van Ut, and N. the Huy, "Hypertext Transfer Protocol.," *Netw. Work. Gr.*, vol. 119, no. 6, pp. 328–332, 1999.

[29] K. Kazantsev, "Development of a Web Application for Management of Public Events," Helsinki Metropolia University of Applied Sciences, 2018.

[30] S. Smith, "Overview of ASP.NET Core MVC," *Microsoft Documents*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-3.1. [Accessed: 27-Nov-2019].

[31] "Stack Overflow Developer Survey 2019," 2019. [Online]. Available: https://insights.stackoverflow.com/survey/2019. [Accessed: 27-Nov-2019].

[32] S. Kilicarslan, "ASP.NET Core MVC Web Application (Project Structure)," 2019. [Online]. Available: https://medium.com/net-core/asp-net-core-mvc-web-application-project-structure-3ccaa244fa66. [Accessed: 27-Nov-2019].

[33] R. Anderson, S. Luttin, and D. Roth, "Introduction to ASP.NET Core | Microsoft Docs," *Microsoft Documents*, 2019. [Online]. Available: https://docs.microsoft.com/en-za/aspnet/core/index?view=aspnetcore-3.0. [Accessed: 27-Nov-2019].

[34] R. Morales-Chaparro, M. Linaje, J. C. Preciado, and F. Sánchez-Figueroa, "MVC web design patterns and rich internet applications," *Proc. Jornadas Ing. del Softw. y Bases Datos*, no. January, 2007.

[35] R. Nor-Mohd, J. Razi, A. A. Zakaria, A. Fakhri, and S. Abdu, "Cloudemy: Step into the cloud," *J. Telecommun. Electron. Comput. Eng.*, vol. 9, no. 3-5 Special Issue, pp. 135–139,

2017.

[36] S. Smith and L. Latham, “Views in ASP.NET Core MVC | Microsoft Docs,” *Microsoft Documents*, 2019. [Online]. Available: https://docs.microsoft.com/en-za/aspnet/core/mvc/views/overview?view=aspnetcore-3.0. [Accessed: 05-Dec-2019].

[37] S.Smith & S.Addie, “Handle requests with controllers in ASP.NET Core MVC | Microsoft Docs,” 2017. [Online]. Available: https://docs.microsoft.com/en-za/aspnet/core/mvc/controllers/actions?view=aspnetcore-3.0. [Accessed: 06-Dec-2019].

[38] G. Vossen, “Databases and database management,” *Handbooks Oper. Res. Manag. Sci.*, vol. 3, no. C, pp. 133–193, 2015.

[39] F. Almeida, *Practical SQL Guide for Relational Databases*, no. September. ISSUU Publishing, 2017.

[40] J. Lerman and R. Miller, *Programming Entity Framework: Code First*. 2012.

[41] T. FitzMacken, “Getting Started with Entity Framework 6 Database First using MVC 5 | Microsoft Docs,” 2014.

[42] M. Meng, S. Steinhardt, and A. Schubert, “Application programming interface documentation: What do software developers want?,” *J. Tech. Writ. Commun.*, vol. 48, no. 3, pp. 295–330, 2018.

[43] S. Castellani, “What are the different types of APIs? Categorization and approaches,” 2017. [Online]. Available: https://apifriends.com/api-creation/different-types-apis/. [Accessed: 08-Dec-2019].

[44] Y. Qiu, “The openness of Open Application Programming Interfaces,” *Inf. Commun. Soc.*, vol. 20, no. 11, pp. 1720–1736, 2017.

[45] OECD, “Unlocking the Digital Economy – A Guide to Implementing Application Programming Interfaces in Government,” p. 70, 2019.

[46] BBVA, “101: Introduction to the world of APIs.” BBVAOpen4U, 2015.

[47] S.Kilicarslam, “Rest Api – .NET Core – Medium,” 2019. [Online]. Available: https://medium.com/net-core/tagged/rest-api. [Accessed: 08-Dec-2019].

[48] L. Li, W. Chou, W. Zhou, and M. Luo, “Design Patterns and Extensibility of REST API for Networking Applications,” *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 1, pp. 154–167, 2016.

Appendix A

Appendixes

- Codes Repositories

https://github.com/LiisaSh/thesis-integrated-his

- Survey link