A CLOUD BASED E-HEALTH SYSTEM FOR MOROCCAN CITIZENS

Capstone Design

Submitted April 2015

Meryem M’hamedi

Supervised by Dr. R. Abid
A CLOUD BASED E-HEALTH SYSTEM FOR MOROCCAN CITIZENS

Capstone Design

Submitted April 2015

Meryem M’hamdi

Supervised by Dr. R. Abid
ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to Dr. Mohammed Riduan Abid for his patient guidance, enthusiastic encouragement and immeasurable support.

I would like to thank Dr. Milton Duncan Rinehart and Dr. Khalid Sendide for their useful suggestions and support, which guided me and enriched the honors component of this capstone.

My special thanks are extended to my family for supporting me and believing in my abilities.
# CONTENTS

1 Introduction 1

2 Methodology 2

3 Planning 3

4 Main Text 4

   4.1 Functional Requirements 4
      4.1.1 Manage Account 4
      4.1.2 Manage Medical File 4
      4.1.3 Manage Appointment 4
      4.1.4 Manage Prescription 5
      4.1.5 Manage TimeSlot 5

   4.2 Design and Analysis 6
      4.2.1 Use Case Diagram 6
      4.2.2 Entity Relationship Diagram and Data Dictionaries 7

   4.3 Implementation 14
      4.3.1 Developing Web Server in Golang 14
         4.3.1.1 Motivation for Golang 14
         4.3.1.2 Building a Web Server using Golang 14
         4.3.1.3 Martini Web Framework 15
         4.3.1.4 Rendering Html Template 17
         4.3.1.5 Connection to Database 18
         4.3.1.6 Using Sessions 18
         4.3.1.7 Bootstrap Library 19
      4.3.2 Deploying Web Application into the Cloud 19
         4.3.2.1 Traditional Versus Cloud Hosting 19
4.3.2.2. Platform as a Service
4.3.2.3. Google App Engine
4.3.2.4. Go SDK for App Engine
4.3.2.5. Cloud SQL
4.3.2.6. System Architecture
4.3.3. Developing Android Version
   4.3.3.1. ADT Plugin to Eclipse Juno
   4.3.3.2. Android SDK
   4.3.3.3. Android Emulator
   4.3.3.4. MAMP Server
4.4. Cloud Web Based Application Screenshots
4.5. Android Application Screenshots
6 Conclusions
7 References
Appendix A: Capstone Specification
Appendix B: Code Snippets
Appendix C: Quick Golang Guide
Appendix D: Honors Research Project
ABSTRACT

This capstone project implements a 3-tier cloud based smart e-health system whose services are tailored to the needs of the Moroccan citizen. The application is also made accessible via mobile using android technology. It creates an intuitive and integral platform that connects patients to available doctors and enables them to manage their medical record. Patients can view and make appointments and view prescriptions once created by their doctors. This project makes use of go language and its net/http library for routing and serving, Cloud SQL for database creation and management and other packages to create a web application. The application is deployed on Cloud via Google App Engine. The android application makes use of android SDK, eclipse plugin for android and MAMP server for linking to remote database.
1. INTRODUCTION

E-health is a promising cyber application, and it is attracting substantial interest as it concerns the well being of every citizen worldwide. It has gained substantial amount of attention during the last few decades being the single-most important revolution in healthcare as it contributes efficiently to the enhancement of the whole clinical decision making process. Morocco is among the countries in which healthcare services are lacking and negatively impact its human development index. It is facing many challenges, which undermine its ability to provide accessible, efficient, and equitable services to its citizens. Those difficulties include the relatively low budget allocated to the health sector, structural weaknesses, inaccessible healthcare centers especially in rural areas, insufficient healthcare workforce allocated to every citizen, and insufficient medical awareness. Here comes the role of technology to serve this critical sector by providing solutions to optimize the resources at hand to provide better healthcare services.

There are many e-health innovations deployed as solutions in several developed countries and integrated into healthcare services. Those solutions include electronic health records which organize patient data and allow their exchange between different healthcare providers, telemedicine which promises a revolutionary treatment procedure by allowing instant online consultation through video conferencing and at distance tele-monitoring of patients, consumer health informatics, health knowledge management and many other applications which extend healthcare methods and possibilities. E-health deals with sensitive patient data whose disclosure can undermine the trust between patients and healthcare providers and whose alteration can alter treatment leading to serious health problems and possibly death. In the Honors research part of this capstone, I will be examining the future of e-health in Morocco and investigates the political, legal, social, economic, technological and ethical environment in order to better understand what kind of e-health solution is more adequate and realistic for the Moroccan citizen.
2. METHODOLOGY

In developing this system, I have followed an iterative approach because of the time constraint and the absence of a client. This approach allows for better flexibility and accelerates the evolution process of the implementation as results can be assessed thoroughly. I have used the method of creation of prototypes, which helped me adjust the functional and nonfunctional requirements of the project. Also since I am new to cloud and android technologies, it helped me realize what can potentially work later on when deployed using those technologies.

I have decided to achieve the honors requirements through a needs assessment study based on secondary data provided by ministry of health and other international organization health reports. The results of the study are to be taken into consideration during the implementation to maximize the value out of the deployed e-health system. In addition to that, a list of recommendations to address ethical issues is to be devised for future consideration. The honors part serves as a strong basis upon which requirements for the deployed system are to be set.
3. PLANNING

<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jan 19th – Jan 23rd</td>
<td>Brainstorming for capstone ideas</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project Selection</td>
</tr>
<tr>
<td>2</td>
<td>Jan 26th – Jan 30th</td>
<td>Initial Specification</td>
</tr>
<tr>
<td>3</td>
<td>Feb 2nd – Feb 6th</td>
<td>Feasibility Study</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setting up implementation tools</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Honors Research</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Getting familiarized with Cloud Computing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Getting familiarized with Android SDK</td>
</tr>
<tr>
<td>4</td>
<td>Feb 9th – Feb 13th</td>
<td>Honors Research</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Building 1st Basic Prototype and graphical user interfaces</td>
</tr>
<tr>
<td>5</td>
<td>Feb 16th – Feb 20th</td>
<td>Honors Research</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Building 2nd prototype and modeling for basic functionality and database</td>
</tr>
<tr>
<td>6</td>
<td>Feb 23rd – Feb 27th</td>
<td>Continue working on 2nd Prototype and modeling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Final Functional and Nonfunctional Requirements</td>
</tr>
<tr>
<td>7</td>
<td>Mar 2nd – Mar 6th</td>
<td>Mid Report Due</td>
</tr>
<tr>
<td>8</td>
<td>Mar 9th – Mar 13th</td>
<td>Implementation of part 1 functionalities for cloud app</td>
</tr>
<tr>
<td>9</td>
<td>Mar 16th – Mar 20th</td>
<td>Implementation of part 2 functionalities for cloud app</td>
</tr>
<tr>
<td>10</td>
<td>Mar 23rd – Mar 27th</td>
<td>Testing and debugging cloud app</td>
</tr>
<tr>
<td>11</td>
<td>Mar 30th – Apr 3rd</td>
<td>Implementation of part 1 functionalities for Android app</td>
</tr>
<tr>
<td>12</td>
<td>Apr 6th – Apr 10th</td>
<td>Implementation of part 2 functionalities for Android app</td>
</tr>
<tr>
<td>13</td>
<td>Apr 13th – Apr 17th</td>
<td>Testing and debugging Android app on android phones and with external users</td>
</tr>
<tr>
<td>14</td>
<td>Apr 20th – Apr 24th</td>
<td>Finalizing the test of cloud and android applications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Final cloud application and android application deployed on the Android App Store</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Final Report</td>
</tr>
<tr>
<td>15</td>
<td>Apr 27th – Apr 30th</td>
<td>PowerPoint Presentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project Defense</td>
</tr>
</tbody>
</table>

Table 3.1  Project Schedule
4. MAIN TEXT

4.1. Functional Requirements

4.1.1. Manage Account:

4.1.1.1. The patient shall be able to register to the system by creating an account based on his/her needs. The patient has to provide the necessary information including username, password, email address and other personal information.

4.1.1.2. The patient and doctor shall login to the system by providing his username and password.

4.1.1.3. The patient and doctor shall logout to the system

4.1.1.4. The patient and doctor shall modify their account information. The user can change their information including password, and personal information including full name, address.

4.1.1.5. The patient shall be able to delete his/her account

4.1.2. Manage Medical File:

4.1.2.1. The patient shall be able to create medical file. Each medical file consists of the file date and information related to disease. The patient is asked to respond to questions about his consultation motive, provide description of what he feels, record his history including previous surgeries, allergies, chronic diseases.

4.1.2.2. The patient shall modify medical file attributes.

4.1.2.3. The patient shall delete medical file.

4.1.3. Manage Appointment:

4.1.3.1. The patient shall be able to add an appointment for a specific date and time with a specific doctor. The patient is in the first stage asked to select the category of doctor based on his/her needs. He/she chooses between online and real consultation Then, the patient selects the appointment based on time and personal preferences.
4.1.3.2. The patient and doctor shall be able to see all his/her appointment. The patient can see the appointment information, the address of the doctor cabinet if it is a real consultation.

4.1.3.3. The patient shall be able to cancel an appointment within 24 hours from its occurrence. The patient should be asked for the reason for appointment cancellation.

4.1.3.4. The patient shall modify appointment information within 24 hours from its occurrence. The patient can select a different time and date for the appointment and choose another doctor.

4.1.4. Manage Prescription:

4.1.4.1. The doctor shall add a prescription form for a particular patient and particular visit. The doctor will enter the details about the prescription by specifying the particular patient condition, diagnosis results and adding medicines to the prescription and their dosage.

4.1.4.2. The doctor shall modify a prescription form. The doctor can delete or add medicine to the prescription. He/she can also modify dosage and usage cases.

4.1.4.3. The doctor shall delete a prescription form.

4.1.5. Manage TimeSlot:

4.1.5.1. The doctor shall be able to add a new time slot that the patient can select afterwards when creating an appointment.

4.1.5.2. The doctor shall be able to delete a time slot not already taken by patient.
4.2. Design and Analysis:

4.2.1. Use Case Diagram:

Figure 1 is the use case diagram for the system which illustrates the operations performed by the different types of users.

![Use Case Diagram](image)

Figure 4.2.1.1 Use Case Diagram
4.2.2. Entity Relationship Diagram and Data Dictionaries:

Figure 2 is the ERD diagram which describes the different entities and relationships between them in the database created for this system.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usr_id</td>
<td>Primary key</td>
<td>serial</td>
</tr>
<tr>
<td>Usr_fname</td>
<td>The user’s first name</td>
<td>Varchar</td>
</tr>
<tr>
<td>Usr_Lname</td>
<td>The user’s last name</td>
<td>varchar</td>
</tr>
<tr>
<td>Usr_type</td>
<td>The user’s type (patient, doctor or admin)</td>
<td>varchar</td>
</tr>
<tr>
<td>Usr_CIN</td>
<td>The user’s identification</td>
<td>varchar</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Description</td>
<td>Data type</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Usr_dob</td>
<td>The user’s date of birth</td>
<td>varchar</td>
</tr>
<tr>
<td>Usr_address</td>
<td>The user’s mailing address</td>
<td>varchar</td>
</tr>
<tr>
<td>Usr_phone</td>
<td>The user’s phone number</td>
<td>varchar</td>
</tr>
<tr>
<td>Usr_gender</td>
<td>Male or female</td>
<td>varchar</td>
</tr>
<tr>
<td>Usr_email</td>
<td>The user’s email address</td>
<td>varchar</td>
</tr>
</tbody>
</table>

**Table 4.2.2.1  Data Dictionary of the table User**

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usr_id</td>
<td>Primary key of table Patient references primary key of table user</td>
<td>serial</td>
</tr>
</tbody>
</table>

**Table 4.2.2.2  Data Dictionary of the table Patient**

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usr_id</td>
<td>Primary key of table Doctor references primary key of table user</td>
<td>serial</td>
</tr>
<tr>
<td>Doc_speciality</td>
<td>Doctor’s speciality falls into one the four categories: physician, psychologist, paediatrician, dentist</td>
<td>varchar</td>
</tr>
<tr>
<td>Doc_since</td>
<td>The year since which the doctor started practicing his job as a doctor: gives an idea about his experience</td>
<td>varchar</td>
</tr>
<tr>
<td>Doc_diploma</td>
<td>Gives an idea about the</td>
<td>varchar</td>
</tr>
</tbody>
</table>
## Table 4.2.2.3  Data Dictionary of the table TimeSlot

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slot_id</td>
<td>Primary Key</td>
<td>serial</td>
</tr>
<tr>
<td>Slot_beg_date</td>
<td>The beginning date of the slot</td>
<td>date</td>
</tr>
<tr>
<td>Slot_beg_time</td>
<td>The beginning time of the slot</td>
<td>time</td>
</tr>
<tr>
<td>Slot_end_date</td>
<td>The ending date of the slot</td>
<td>date</td>
</tr>
<tr>
<td>Slot_end_time</td>
<td>The ending time of the slot</td>
<td>time</td>
</tr>
<tr>
<td>Slot_taken</td>
<td>A flag that indicates whether the appointment has been taken by some patient or not</td>
<td>Bool</td>
</tr>
<tr>
<td>Usr_id</td>
<td>Foreign key that references the id of the doctor who has this time slot.</td>
<td>int</td>
</tr>
</tbody>
</table>

## Table 4.2.2.4  Data Dictionary of the table TimeSlot

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>App_id</td>
<td>Primary Key</td>
<td>serial</td>
</tr>
<tr>
<td>App_Creatdate</td>
<td>The date of creation of the appointment</td>
<td>date</td>
</tr>
<tr>
<td>App_Creattime</td>
<td>The time of creation of the appointment</td>
<td>time</td>
</tr>
<tr>
<td>patient_ID</td>
<td>Foreign key that references the id of the patient who has taken the appointment</td>
<td>int</td>
</tr>
<tr>
<td>slot_id</td>
<td>Foreign key that references the id of the slot corresponding to the appointment booked</td>
<td>int</td>
</tr>
</tbody>
</table>

**Table 4.2.5 Data Dictionary of the table Appointment**

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>The user’s account chosen username</td>
<td>varchar</td>
</tr>
<tr>
<td>password</td>
<td>The user’s account password</td>
<td>varchar</td>
</tr>
<tr>
<td>Usr_id</td>
<td>Foreign key references primary key of table user</td>
<td>serial</td>
</tr>
</tbody>
</table>

**Table 4.2.6 Data Dictionary of the table Account**

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Med_code</td>
<td>The medicine identification number and is the primary key of the table Medicine.</td>
<td>serial</td>
</tr>
<tr>
<td>Med_name</td>
<td>The medicine commercial name.</td>
<td>varchar</td>
</tr>
<tr>
<td>med_scientificName</td>
<td>The scientific description of the types and numbers of molecules included in the medicine</td>
<td>varchar</td>
</tr>
<tr>
<td>med_descript</td>
<td>The contents of the medicine</td>
<td>varchar</td>
</tr>
<tr>
<td>med_sideEffects</td>
<td>The undesirable effects that the patient can experience after</td>
<td>varchar</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Description</td>
<td>Data type</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>med_allergies</td>
<td>Allergies that the patient should not have before taking the medicine.</td>
<td>varchar</td>
</tr>
<tr>
<td>med_availability</td>
<td>Marks the availability of the medicine in the market (whether it is still sold in the market)</td>
<td>boolean</td>
</tr>
</tbody>
</table>

Table 4.2.2.7 Data Dictionary of the table Medicine
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Med_code</td>
<td>The identification number of the medicine that is prescribed by the doctor</td>
<td>int</td>
</tr>
<tr>
<td>pres_code</td>
<td>The identification number of the prescription in which the medicine is</td>
<td>int</td>
</tr>
<tr>
<td>dose</td>
<td>How much the patient should take from the medicine</td>
<td>varchar</td>
</tr>
<tr>
<td>takingTime</td>
<td>When the drug should be taken by the patient.</td>
<td>varchar</td>
</tr>
<tr>
<td>duration</td>
<td>Duration needed in taking the medicine</td>
<td>varchar</td>
</tr>
<tr>
<td>frequency</td>
<td>The number of times the medicine should be taking (how many times per day,</td>
<td>varchar</td>
</tr>
<tr>
<td></td>
<td>how many times per week…)</td>
<td></td>
</tr>
</tbody>
</table>
### Table 4.2.2.9  Data Dictionary of the table Prescribe

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>pres_code</td>
<td>Primary Key</td>
<td>serial</td>
</tr>
<tr>
<td>pres_date</td>
<td>The date at which the doctor created prescription for the appointment</td>
<td>date</td>
</tr>
<tr>
<td>pres_time</td>
<td>The time at which the doctor created prescription for the appointment</td>
<td>time</td>
</tr>
<tr>
<td>Pres_bloodPressure</td>
<td>Measured Blood Pressure at the time of the appointment occurrence</td>
<td>int</td>
</tr>
<tr>
<td>Pres_weight</td>
<td>Measured Weight in kg at the time of the appointment occurrence</td>
<td>int</td>
</tr>
<tr>
<td>Pres_height</td>
<td>Measured Height in cm at the time of the appointment occurrence</td>
<td>int</td>
</tr>
<tr>
<td>Pres_temp</td>
<td>Measured Temperature at the time of the appointment occurrence</td>
<td>int</td>
</tr>
<tr>
<td>app_id</td>
<td>Foreign key references appointment occurrence to make sure appointment that occurred</td>
<td>int</td>
</tr>
</tbody>
</table>

### Table 4.2.10  Data Dictionary of the table Prescription
4.3. Implementation:

4.3.1. Developing Web Server in Golang

4.3.1.1. Motivation for Golang:

The Web server was developed using “golang” as the event handler programming language and html5, css and javascript for styling and gui design. My motivation for this newly created programming language rather than using other ancient languages such as Java and C# comes from the following facts. It supports concurrency, modularity and ensures excellent integration with other services better than other programming languages. Concurrency is a highly desirable feature in building web application. Its role is to “increase the throughput of your web application by completing each incoming request faster, freeing up network and hardware resources for other end-users.” [1] Golang has this capability embedded as a built-in feature known as “goroutines”. The peculiarity of concurrency in golang compared to other languages is how go-routines make it incredibly easy to manage threads. Golang support for modularity is of extensive importance since large web services require a code that can grow easy beyond control. Therefore, debugging is very efficient to the organization of the code and building is incredibly speedy because of Go’s package smart build system. “If you are using packages A and B which both call a package C, package C only needs to be built once (instead of having the code being imported and built into each package separately.” [2] In addition to that, it is robust against bad code and copy-paste. “Golang is harsh, and not at all forgiving with sloppy code.”[1] It allows the efficient use of memory by not allowing the declaration of a variable or an imported package with actually using them in the code. It enforces good programming principles by making it impossible to proceed with running the program without making sure to check for code that returns error in case it was designed to do so.

4.3.1.2. Building a Web Server using Golang:

Go is an extremely clean and fun language to work with and has a bunch of handy, modular, well documented packages out of the box. We can easily build a web app using pure Go for handler functions and the net/http library for routing
and serving. After installing the Go programming language and setting up a GOPATH for our golang source project to be built from, a set of packages are required to import functionalities needed for this project. One of the most important packages for any web application is the **net/http** package. It allows you to build HTTP servers in go. The http.Handler interface encapsulates http request-response pattern by processing data coming from the object http.Request and writing response to http.ResponseWriter as illustrated in the following interface definition:

```go
type Handler interface {
    ServeHTTP.ResponseWriter, *Request)

While http.ResponseWriter object is described bellow:

```go
type ResponseWriter interface {
    Header() Header
    Write([]byte) (int, error)
    WriteHeader(int)
}
```

### 4.3.1.3. Martini Web Framework:

In handling http requests, I have decided to use martini go package which is an API that uses a thin layer of abstraction over the stblib’s net/http package and which has classy routing and middleware support. To make use of martini package to add router functions to my web application, we can use this code:

```go
m := martini.Classic()
```

This Classic Interface enables the use of some basic default middleware convenient for my purposes. “Classic creates a classic Martini with some basic default middleware - martini.Logger, martini.Recovery and martini.Static. Classic also maps martini.Routes as a service.”[3]

When an html form in my web application is submitted for processing, my golang web server renders the form by matching the form’s function to a function handler. The connection between my web interface and web interface is
done easily using martini Router interface using functions like Post and Get. For example, login function web interface defined by the following html code:

```html
<form name="LoginPatient" class="form-horizontal templatemo-login-form-2" role="form" action="/loginPatient" method="post">
  <div class="row">
    <div class="col-md-12">
      <h1>Login Form</h1>
    </div>
  </div>
  <div class="row">
    <div class="form-group">
      <div class="col-md-12">
        <label for="username" class="control-label">Username</label>
        <div class="templatemo-input-icon-container">
          <i class="fa fa-user"></i>
          <input type="text" class="form-control" id="username" name="username" placeholder="" required>
        </div>
      </div>
    </div>
    <div class="form-group">
      <div class="col-md-12">
        <label for="password" class="control-label">Password</label>
        <div class="templatemo-input-icon-container">
          <i class="fa fa-lock"></i>
          <input type="password" class="form-control" id="password" name="password" placeholder="" required>
        </div>
      </div>
    </div>
  </div>
</form>
```
The code that enables the processing of this form’s action upon its submission by my web server is:
```
func loginPatientHandler(req *http.Request, w http.ResponseWriter, s sessions.Session, db *sql.DB) {
    username := req.FormValue("username")
    // process the username
}
```

The function `func loginPatientHandler(req *http.Request, w http.ResponseWriter, s sessions.Session, db *sql.DB)` gets form values by calling `req.FormValue("username")` using the input name as an argument.

### 4.3.1.4. Rendering Html Template:

Go incorporates one of the best templating facilities for its simplicity and security. “Package template (html/template) implements data-driven templates for generating HTML output safe against code injection.”[5] Hence using this package, golang code can parse and execute HTML files safely and easily.

```
func (t *Template) Parse(src string) (*Template, error)
func (t *Template) ParseFiles(filenames ...string) (*Template, error)
func (t *Template) Execute(wr io.Writer, data interface{}) error
```

After parsing an html file or string into a template, data can be written in the rendered template by encapsulating it into structs. “Executing html template will
apply a parsed template to the specified data object, writing the output to wr.”

4.3.1.5. Connection to Database

I used golang “database/sql” package in the standard library to connect to a database driver namely CloudSQL (which I am going to tackle more deeply in later sections). Knowing the database driver name and driver-specific data source name, I used function func Open(driverName, dataSourceName string) (*DB, error) which opens a database.

```go
func SetupDB() *sql.DB{
    db, err := sql.Open("mysql", "root:C2jvxzqb@cloudsql(stately-sentry-88714:health)/HealthDatabase")
    PanicIf(err)
    err = db.Ping()
    if err != nil {
        fmt.Println(err)
    }
    return db
}
```

We make it ready for use by our martini middleware by calling function Map; m.Map(SetupDB()). Examples of the code used for selecting, inserting, updating and deleting data from and into database are provided as code snippets in appendix A.

4.3.1.6. Using sessions:

For handling web sessions we use "github.com/codegangsta/martini-contrib/sessions" which provides a middleware to handle all the session logic. This requires a specific type of storage. Accordingly, I used a new cookiestore to store my session information as in this code snippet `store := sessions.NewCookieStore([]byte("usersession"))`. Then, I needed to add my session store to my martini instance so that it starts using it. `m.Use(sessions.Sessions("user",store))`. Only after that, I could get sessions.Session injected in my function handlers as a service whenever I need it.
To store a new session value into my session store, I use `s.Set("userID",user.Id)` and to retrieve the value for the same key “userID”: `userId := s.Get("userID")`

4.3.1.7. **Bootstrap Library:**

The most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. I used it for styling my web application as it is the most efficient way to reuse code and to get a good and smooth stylish design. I accessed most of the CSS styles and classes and HTML templates Resources from [http://getbootstrap.com/](http://getbootstrap.com/)

4.3.2. **Deploying Web Application into the Cloud**

The main purpose of this project is to get familiarized with the cloud platform as a solution for hosting user’s applications, which are made accessible to Internet users. Compared to traditional hosting solutions, cloud hosting provides higher reliability, scalability and affordability. To see more concretely cloud advantages, I will examine some well-known traditional hosting methods and compare their efficiency to cloud.

4.3.2.1. **Traditional Versus Cloud Hosting:**

There are two well-known types of traditional hosting: shared and dedicated hosting. For shared hosting, “the client pays for a set amount of space (storage) on a single server, and that server’s resources are shared by a number of other websites.” [8] This is the economic approach to hosting since it requires only a small portion of space and less maintenance; however, it is a low reliability and availability approach since if the only server dedicated breaks down, the whole service is not accessible. Also, with shared hosting, storage of resources proper to one user is done on the same server as other users and applications. The biggest disadvantage of this approach is that “the servers running shared hosting accounts are almost always fully loaded” [9] meaning that if a single resource takes a huge space of the server, accessing all resources in the same server is subject to low speed. Compared to shared hosting, cloud hosting offers the possibility of hosting one resource on several servers which work as one entity instead of just one server. Figure 4.1.1.3.1.1 illustrates this difference.
The other traditional hosting approach known as dedicated hosting is similar to cloud hosting in the fact that it uses multiple servers and the control is leased to the web hosting providers who take care of technical support, backup and maintenance. However, the key difference is in the genuine pay per use plan of cloud hosting. You only pay for what you use. “Cloud hosting companies provide virtual space on an on-demand, as-needed basis”[8]. On the contrary, dedicated hosting is less cost efficient. “The resources of a dedicated server are usually used below 60%, so it never runs at full capacity.”[9]

4.3.2.2. **Platform as a Service:**

It is a cloud-based solution to host user’s applications, which are made accessible via Internet. Using this service provides users with a high level of simplicity and flexibility while creating their software applications since among all features provided they have the choice to include only the features needed. It all depends on the user’s expectations and level of expertise. PaaS services are paid for on subscription basis. Other key benefits of PaaS include for developers:
• No need to invest in physical infrastructure, the focus is diverted to the development of applications and nothing more.
• No advanced expertise is required to take advantage of PaaS services
• Adaptability
• Efficient Team work supported
• Data security, backup and recovery are better supported

In sum, “a PaaS offering supplies an operating environment for developing applications by providing an architecture, an overall infrastructure to support application development including networking, storage, software support and management services”.[10]

4.3.2.3.  **Google App Engine:**
Google App Engine is a Platform as a Service (PaaS) offering whole role is to build and run applications on Google’s infrastructure. App Engine applications are easy to build, maintain, and scale as traffic and data storage needs change. With App Engine, there are no servers for you to maintain. The reason why I picked this solution is because it provides a rich set of options for reliable and efficient data storage ranging from NoSQL datastore, Google Cloud SQL, for pertinent pay plan and implementation and testing features thanks to its interactive Developers Console.

The following description explains Developers Console work environment.

![Google Developers Console](image)

**Figure 4.1.1.3.3.1**

This figure shows the console’s home page, which contains the set of projects created and deployed into Google App Engine.
Upon selecting a project, the different options for managing the project are made available.

The dashboard shows the web address for which the deployed application is made available along with its version number. In my case, the web address is http://stately-sentry-88714.appspot.com/ and it is version one. It also shows the usage duration per day and the average memory used.

**Figure 4.1.1.3.4.2**

4.3.2.4. **Go SDK for App Engine:**

Deploying my web server written in golang, I installed Go SDK, which includes the Go compiler and standard library for easy deployment. “App Engine builds and executes Go application code using a safe "sandboxed" environment.”[11] The used set enabled by Go SDK comes with a reduced set of libraries but make use of “scalable services provided by App Engine to store data and communicate over the Internet.”[11]
After the Go SDK is installed, my golang web server along with its static javascript, html and css files are deployed following this steps I created an app.yaml file inside the project containing my healthApp.go program and static html files. The role of this file is to link a project instance created in my app engine developers console to the go runtime environment. As per modular golang project, an app.yaml is required inside the same directory of every single module. In my case, I used only one program, which is healthApp.go so I had to create only one app.yaml file inside the same directory of my golang program.

Here are the contents my app.yaml file:

```
application: stately-sentry-88714
version: 1
runtime: go
api_version: go1

handlers:
- url: /css
  static_dir: public/css

- url: /images
  static_dir: public/images

- url: /js
  static_dir: public/js

- url: /.*
  script: _go_app
```

The runtime variable selects the Go runtime environment and the api_version used here is go1 which is the latest golang version supported by Google App Engine. The version variable shows that many versions of the deployed
application in cloud could be used. To load my html, css and javascript files as static files, I declared directories for these respective files under static_dir. I made use of appcfg.py, the multipurpose tool that handles all command-line interaction with my application running on App Engine. Using goapp serve from my terminal, I can build, install my application locally and run it on localhost:8080. Using goapp deploy from terminal as well, after I am asked to provide my credentials, all files are loaded and compiled, the application is deployed and can be accessed from Internet using address http://stately-sentry-88714.appspot.com/. Following are screenshots of those two actions run from terminal and their log messages.
Figure 4.1.1.3.4.1

Latest SDK:
api_versions: ['1']
release: 1.9.19
supported_api_versions:
go:
    api_versions: [go1]
java7:
    api_versions: ['1.0']
python:
    api_versions: ['1']
python27:
    api_versions: ['1']
timestamp: 1424415497

----------

Your SDK:
api_versions: ['1']
release: 1.9.18
supported_api_versions:
go:
    api_versions: [go1]
java7:
    api_versions: ['1.0']
python:
    api_versions: ['1']
python27:
    api_versions: ['1']
timestamp: 1422656281

----------

Please visit https://developers.google.com/appengine/downloads for the latest SDK
4.3.2.5. *Cloud SQL:*

It is an object-relational highly and performant open source database management system. I used database/sql package in golang to connect, read and write from the database created using CloudSQL. To create my cloudSQL database, I created a database instance `stately-sentry-88714:health`, which is linked it to my project instance name. I configured the access Control parameters by adding authorization to my own ip address and requesting the IPv4 address. All other database and tables creation actions were then made from my terminal. I connect to my cloudSQL database instance using IPv4 address as host address, root as user and password as root password. The following screenshot illustrates the connection to cloudSQL from terminal:
After connecting to HealthDatabase, the tables contained in that database are shown as follows and queries are done the usual way.

```
mysql> show databases;
+--------------------------+
<table>
<thead>
<tr>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>information_schema</td>
</tr>
<tr>
<td>Health</td>
</tr>
<tr>
<td>HealthDatabase</td>
</tr>
<tr>
<td>mysql</td>
</tr>
<tr>
<td>performance_schema</td>
</tr>
</tbody>
</table>
+--------------------------+
```

Figure 4.1.1.3.5.1

After connecting to HealthDatabase, the tables contained in that database are shown as follows and queries are done the usual way.

```
mysql> show tables;
+--------------------------+
<table>
<thead>
<tr>
<th>Tables_in_HealthDatabase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
</tr>
<tr>
<td>Appointment</td>
</tr>
<tr>
<td>Doctor</td>
</tr>
<tr>
<td>MedicalFile</td>
</tr>
<tr>
<td>Medicine</td>
</tr>
<tr>
<td>Patient</td>
</tr>
<tr>
<td>Prescribe</td>
</tr>
<tr>
<td>Prescription</td>
</tr>
<tr>
<td>TimeSlot</td>
</tr>
<tr>
<td>UserTable</td>
</tr>
</tbody>
</table>
+--------------------------+
10 rows in set (0.18 sec)
```

```
mysql> select * from Doctor;
+------------+--------------+-------------+--------------------------+
| usr_ID     | doc_speciality | doc_since   | doc_diploma              |
|------------+--------------+-------------+--------------------------|
| 2          | Dentist      | 1970        | Got his Doctorate in dental surgery from the University o |
| 3          | Dentist      | 1980        | Got his Doctorate from school of medicine in Rabat        |
| 4          | Physician    | 1990        | Studied in Algeria and got his diplomate of doctor end sp |
| 5          | Psychologist | 2000        | Graduated from McGill University                           |
| 6          | Dentist      | 2006        | Graduated from Faculty of Medicine in Casablanca          |
+------------+--------------+-------------+--------------------------+
```

Figure 4.1.1.3.5.2
4.3.2.6. **System Architecture:**

The following figure summarizes the whole system architecture involving google app engine and the way it connects to Google CloudSQL and to end users through thin web or android client application.

![System Architecture Diagram](image)

**Figure 4.3.2.6.1**

4.3.3. **Developing Android Version**

In developing the android native application for my e-health project, I have opted for a start from scratch implementation approach since web and android are two different platforms that cannot be linked directly. However, I made use of the same database design and information by migrating it to MAMP server. Installing and getting familiarized with the environment for developing android application was the first step. I used the following tools:
4.3.3.1. **ADT Plugin to Eclipse Juno:**

It is a plugin for the Eclipse IDE that extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute your application. I opted for this approach since I have already worked on eclipse and I am well familiarized with its development features.

4.3.3.2. **Android SDK:**

This is an Android Development Kit that links to eclipse. It is a software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries/APIs to build Android applications. I used Android SDK version of 5.1.1 in developing my application.

4.3.3.3. **Android Emulator:**

The emulator is a virtual Android mobile device. It simulates the functionalities of a real phone gives the exact outputs that a real phone is supposed to give. I used it for application testing to simulate target device.

4.3.3.4. **MAMP Server:**

This is the mac version of Wamp server. As an android application can never access a database directly, the use of Web Server for remote access to the database is necessary. So, I implemented PHP files that would form a sort of intermediary between my application and the MySQL database I used for implementation and testing. These files get data from or post it to the database and return JSON (JavaScript Object Notation) objects. The MAMP Server encapsulates an Apache Tomcat Server that hosts the php files and a MySQL database server that hosts the database.

My MAMP server configuration is shown in the following figures, in which ports for Apache server and Mysql server are set respectively to 8888 and 8889.
Figure 4.3.3.4.1

The apache and mysql server are shown to be working by turning it green colour after starting them.

Figure 4.3.3.4.2
Accessing MySQL phpMyAdmin page leads me to an interactive work environment, which enables database importation, exportation, creation, executing select, insert, delete, update actions and queries on different tables and so on. Thus, it allows me to monitor, see and manage changes done to the database after execution of my android app.

My database structure as seen from MySQL phpMyAdmin is shown in the figure below:

![MySQL phpMyAdmin Interface](image)

Figure 4.3.3.4.3
I have written php scripts to carry out read, update, write and delete actions from and on database located in my mamp server. The interested reader may refer to Appendix A to see some example of code snippets that connect to MAMP server database using php. I have stored all my php scripts performing operations on e-health database needed by my android application in htdocs folder to make them accessible from http://localhost:8888/. Here a screenshot of some of my php scripts accessible from http://localhost:8888/ which constitutes a way to test that my scripts work as expected before linking my android to them.
I developed my android application using modular programming by creating classes with each encapsulated method making use of a single or more php scripts. Again, the interested reader may refer to Appendix A which contains code snippets that show how android activities are linked to php scripts using JSON encoded objects.

4.4. Cloud Web Based Application Screenshots:

The following screenshots show the execution of some of the application’s main functionalities.

The home page contains five sections: image slider, about us, our team, our services and contact us section.

![Figure 4.3.2.1 Home Page](image-url)
The contact us form enables a user to send an email to the e-health system administrator.
To be able to use the application functionalities including booking an appointment with a doctor with specialty of choice, manage medical file and view prescriptions online, the patient needs to create an account. Here, we consider that only administrator can create the doctor account, for security reasons.
Three types of users can login into the application: patient, doctor and admin. User should choose the option that corresponds to their status.
Figure 4.3.2.7 Login Form for Patient

Figure 4.3.2.8 Update Personal Information Form
Figure 4.3.2.9  Patient Home Page
Figure 4.3.2.10  View Appointment Form

Figure 4.3.2.11  View Medical File Form
Figure 4.3.2.12  Choose Doctor

Figure 4.3.2.12  Select TimeSlot Form
Figure 4.3.2.13  Create Medical File Form

Figure 4.3.2.14  Doctor Login
Figure 4.3.2.15 Add Time Slot for Doctor

Figure 4.3.2.16 Time Slot Added for Doctor
Figure 4.3.2.17 Delete Time Slot for Doctor

Figure 4.3.2.18 Cancel Appointment
4.5. Android Version Screenshots:

![E-Health App Home Page](image)

Figure 4.5.1. Home Page
Figure 4.5.2. Create Account
Registration Success

Account successfully created. You can now login!

Figure 4.5.3. Successful Creation of Account
Figure 4.5.4. View Appointment

Figure 4.5.5. Login Patient

Username
meryemmhamdi

Password
**********

LOGIN

New to E-Health App? Register here
5. CONCLUSIONS

This capstone provided me with the great opportunity to develop a web application using the new smart programming language golang which is gaining its place quickly and becoming more powerful compared other old languages such as Java and C#. I also had the chance to get familiarized with cloud deployment. I was able to make use of Google App Engine and CloudSQL and the deployment using golang as the client end language made it easier to switch to the cloud platform. My capstone journey was rich with the usage of new languages and technologies and getting familiarized with different deployment platforms as I had to build the android version of my application as well. I used xml and java to develop the client side server and MAMP server along php scripts and mysql to connect the client app to the backend server. Above all, it was a challenging and rewarding experience in
which I got a hand on new technologies and enforced my knowledge of software engineering and database management.

6. REFERENCES


E-health is a promising cyber application, and it is attracting substantial interest as it concerns the well being of every citizen worldwide. The objective of this capstone project is to produce a cloud based smart e-health system whose services are tailored to the needs of Moroccan citizens. The basic idea is to create an intuitive and integral platform that connects patients to available doctors for online consultation and to conceive a smart health prediction system that provides end user support.

The requirement phase of this project will consist of researching the Moroccan market and investigating the different deployments that do exist in the domain of e-health to work out a fitting towards a cloud deployment with specificities relevant to the Moroccan citizen. By the end of two or three weeks, the requirements will be clearly set and we can move to the deployment of the system.

It has been decided to create a 3-tier application deployed in cloud using Google App Engine. The implementation will consist of two parts. The first part will consist of implementing the back end including the application server and the database. The programming language that we have chosen to use is Go which is a low level language developed by Google and which is designed specifically to fit cloud deployment thanks to its speed in handling concurrency and its built-in lightweight framework for testing. The second part will consist of the front end, which will be designed using a HTML, CSS, and JS framework like Twitter Bootstrap Library, Foundation or SproutCore. The application is made accessible via mobile through an android app.

The testing of the application will be performed alongside with the implementation through a series of local unit tests to check the quality of the code written to improve the development process. In addition to that, the application will be tested as a whole, during the last two weeks, to inspect unforeseen anomalies and to further tune the system.

The system has important social implications, as it will contribute effectively to the well being of every citizen, improve health surveillance, health education and clinical decision-making. It is helpful to have direct access to the different health services needed, especially in case of emergencies that cannot reach the hospital or with no doctors in the area. The patients can consult doctors or get support from other services online with no further financial burden associated with displacement and with a better productivity. The challenge is to avoid any integrity, confidentiality and privacy issues and to comply to legal policies protecting consumer rights by providing accurate and up-to-date information and respecting the confidentiality of information shared by patients with real and trusted doctors.
APPENDIX B: CODE SNIPPETS

```go
err := db.QueryRow("select usr_fname, usr_lname, usr_email, Account.usr_ID from Account join UserTable on Account.usr_ID = UserTable.usr_ID where username = ? and password = ? and usr_type = 'doctor' ", username, password).Scan(&user.Fname,&user.Lname,&user.Email,&user.Id)
if err != nil {
    flash := &Flash {Message: "Invalid Username/password",Type: "Error"}
}
tmpl, err := template.ParseFiles("public/LoginDoctor.html")
if err != nil {
    http.Error(w, err.Error(), http.StatusInternalServerError)
    return
}

tmpl.Execute(w,flash)
return
```

**Code Snippet 1**

To check that the user’s username and password pattern match one of the ones existing in the database, we use QueryRow function:

```go
func (db *DB) QueryRow(query string, args ...interface{}) *Row
```

This function executes a query that is expected to return at most one row. QueryRow always return a non-nil value. Errors are deferred until Row's Scan method is called. In case of errors (err!= nil), the html template is rendered to include an error flash message with message “Invalid Username/Password” popping up as a javascript alert dialog.

```go
rows, err := db.Query("select slot_id, slot_beg_date, slot_beg_time, slot_end_date, slot_end_time, slot_taken, usr_id from TimeSlot where usr_id = ? ",s.Get("userID")
if err != nil {
    fmt.Println("select time slot error")
    http.Error(w, err.Error(), http.StatusInternalServerError)
}
```

---

51
return
}

**Code Snippet 2**

This code snippet shows an example of how a select statement returning multiple rows is done. I used Row function:

```go
func (db *DB) Query(query string, args ...interface{}) (*Rows, error)
```

Query executes a query that returns rows, typically a SELECT. The args are for any placeholder parameters in the query. The

```go
_,err1 := db.Exec("INSERT INTO Appointment
(patient_id,app_create_date,app_create_time,slot_id) values
(?,?,?,?)",s.Get("userID"),todayDate,todayTime,s.Get("slotID"))
if err1 != nil {
    fmt.Println("insert appointment error")
    http.Error(w, "app"+err1.Error(), http.StatusInternalServerError)
    return
}
```

**Code Snippet 3**

To insert a new row into my database table Appointment, I used Exec function:

```go
func (db *DB) Exec(query string, args ...interface{}) (Result, error)
```

This function Exec executes a query without returning any rows. The args are for any placeholder parameters in the query. The query being executed here is "INSERT INTO Appointment (patient_id,app_create_date,app_create_time,slot_id) values (?,?,?,?)" where ? marks are replaced in order by s.Get("userID") (the session value corresponding to the user ID of the user currently logged on), todayDate, todayTime, s.Get("slotID").

```go
_, err := db.Exec("update TimeSlot set slot_beg_date= ? where slot_id = ?",beg_date,slot)
if err != nil {
    http.Error(w, err.Error(), http.StatusInternalServerError)
    return
}
```
Code Snippet 4
This code snippet shows the method used to execute an update on a database table.

_, err2 := db.Exec("delete from Appointment where app_id = ?", app_ID)
    if err2 != nil {
        fmt.Println("delete appointment error")
        http.Error(w, err2.Error(), http.StatusInternalServerError)
        return
    }

Code Snippet 5
This code snippet shows the method used to execute the deletion of a row from a database table.

$conn = new mysqli("127.0.0.1", "root", "C2jvxzqb", "HealthCare");
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

Code Snippet 6
This code snippet shows php code for connecting to database giving host address (localhost), user (root) , password and database name. An error is returned in case the connection doesn’t work.

if (isset($_GET["patient"])&&isset($_GET["date"])) {
    $patient = $_GET["patient"];  
    $date = $_GET["date"];  
    $sql = "select app_id, slot_beg_date, slot_beg_time, usr_id, slot_end_date, slot_end_time from appointment inner join TimeSlot on TimeSlot.slot_id=appointment.slot_id where patient_id=$patient and slot_beg_date='$date'";
    $result = $conn->query($sql);
if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        $response["success"] = 1;
        $app = array();
        $app["app_id"] = $row["app_id"];
        $app["slot_beg_date"] = $row["slot_beg_date"];
        $app["slot_beg_time"] = $row["slot_beg_time"];
        $app["slot_end_date"] = $row["slot_end_date"];
        $app["slot_end_time"] = $row["slot_end_time"];
        $app["usr_id"] = $row["usr_id"];
        // success

        $response["message"] = "Success";

        // user node
        $response["app"] = array();

        array_push($response["app"], $app);

        // echoing JSON response
        echo json_encode($response);
    }
} else {
    // no product found
    $response["success"] = 0;
    $response["message"] = "Error";

    // echo no users JSON
    echo json_encode($response);
die();

} else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    // echoing JSON response
    echo json_encode($response);
    die();
}

**Code Snippet 7**

This shows php code can be used to check for provided input and stop in case required field is missing returning a JSON encoded response with success value 0 and message “Required field(s) is missing”. In case required fields are given which are in this case the patient id, and the beginning date of the appointment, a read query is executed to get results in the form of an array named app whose app_id, slot_beg_date, slot_beg_time, slot_end_date, slot_end_time get filled with query row results. In case, this goes without error, a success status of 1 is returned, a success message and the whole app array are returned as a JSON Object.

```java
JSONObject jsonParser = new JSONObject();
private static final String urlviewApp = "http://10.0.2.2:8888/viewApp.php";
private ArrayList<TimeSlot> timeSlotList = new ArrayList<TimeSlot>();

public ArrayList<Appointment> viewAppPat(int user_id, String slot_beg_date){
    int success;
    String message;
    StringBuilder sb = new StringBuilder();
    sb.append(""");
    sb.append(user_id);
    String id = sb.toString();
    try {
        // Building Parameters
        List<NameValuePair> params = new ArrayList<NameValuePair>();
```
params.add(new BasicNameValuePair("patient",id));
params.add(new BasicNameValuePair("date",slot_beg_date));
JSONObject json = jsonParser.makeHttpRequest(urlviewApp, "GET", params);
success = json.getInt("success");
if (success ==1){
    Log.d("Success","1");
    JSONArray appObj = json.getJSONArray("app");
    for (int i = 0; i < appObj.length(); i++) {
        JSONObject appointment = appObj.getJSONObject(i);
        Appointment app = new Appointment();
        app.setAppId(appointment.getInt("app_id"));
        app.setSlotBegDate(appointment.getString("slot_beg_date"));
        app.setSlotBegTime(appointment.getString("slot_beg_time"));
        app.setSlotEndDate(appointment.getString("slot_end_date"));
        app.setSlotEndTime(appointment.getString("slot_end_time"));
        Account acc = new Account();
        StringBuilder sb1 = new StringBuilder();
        sb1.append(appointment.getInt("usr_id"));
        String usrid = sb1.toString();
        Log.d("***************usr_id",usrid);
        Account acc1 = acc.getPersonalInfo(appointment.getInt("usr_id"));
        app.setDocFname(acc1.getFname());
        app.setDocLname(acc1.getLname());
        appList.add(app);
    }
}
} catch (JSONException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
return appList;

---

**Code Snippet 8**

As this code snippet shows, the link between php script and android activity class is done through the use of the php script url [http://10.0.2.2:8888/viewApp.php](http://10.0.2.2:8888/viewApp.php).

Parameters for GET variables in the php script are done through

```java
List<NameValuePair> params
```

by adding new BasicNameValuePair for each parameter.

Then those params are passed into JSONParser method

```java
JSONObject json = jsonParser.makeHttpRequest(urlviewApp, "GET", params);;
```

Response values are then retrieved from this json object.
// mysql update row with matched pid
$result = mysql_query("UPDATE usertable SET usr_lname = '$lname', usr_fname = '$fname',
usr_CIN = '$CIN',usr_dob='$dob', usr_address='$address',
usr_phone='$phone', usr_email='$email' WHERE usr_id = $usr_id");

// check if row inserted or not
if ($result) {
    // successfully updated
    $response["success"] = 1;
    $response["message"] = "Account successfully updated.";

    // echoing JSON response
    echo json_encode($response);
} else {
    $response["success"] = 0;
    $response["message"] = "Account not updated successfully!";
    echo json_encode($response);
}

**Code Snippet 9**

This php script extract shows an example of the execution of an update statement on Account. Again, the response is returned in the form of a JSON Object.

// array for JSON response
$response = array();

//Fields
$med_code = $_POST['med_code'];
$pres_code = $_POST['pres_code'];
$dose = $_POST['dose'];
$duration = $_POST['duration'];
$sfrequency = $_POST['frequency'];
$stakingTime = $_POST['takingTime'];

//Check for required fields
if( trim($med_code)==='' || trim($pres_code)==='' || trim($dose)==='' || trim($duration)==='' || trim($frequency)==='' || trim($stakingTime)==='' ){
   $response['success'] = -1;
   $response['message'] = "Required field(s) is missing";
   // echoing JSON response
   echo json_encode($response);
   die();
}

//All required fields are provided, so insert
else {
   $sql= "INSERT INTO Prescribe (med_code, pres_code, dose, duration, frequency, takingTime) values ($med_code, $pres_code, '$dose', '$duration', '$frequency', '$stakingTime')";
   if ($conn->query($sql) === TRUE) {
      $response['success'] = 1;
      $response['message'] = "Prescribe created successfully";
   } else {
      echo "Error: ". $sql . "<br>" . $conn->error;
      $response['success'] = 0;
      $response['message'] = "Prescribe not created successfully!";
   }
}

Code Snippet 10

This php script extract shows an example of the execution of an insert statement on table Prescribe. Again, the response is returned in the form of a JSON Object.
// array for JSON response
$response = array();

// check for required fields
if (isset($_POST['file_id'])) {
    $file_id = $_POST['file_id'];

    // include db connect class
    require_once __DIR__ . '/db_connect.php';

    // connecting to db
    $db = new DB_CONNECT();

    // mysql update row with matched pid
    $result = mysql_query("delete from medicalFile where file_id = $file_id");

    // check if row deleted or not
    if (mysql_affected_rows() > 0) {
        // successfully updated
        $response['success'] = 1;
        $response['message'] = "MedicalFile successfully deleted";
    } else {
        // no product found
        $response['success'] = 0;
        $response['message'] = "MedicalFile not deleted";
    }

    // echo no users JSON
    echo json_encode($response);
} else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    // echoing JSON response
    echo json_encode($response);
}

Code Snippet 11

This php script extract shows an example of the execution of a delete statement on table MedicalFile. Again, the response is returned in the form of a JSON Object.

APPENDIX C: QUICK GOLANG GUIDE

Go Setup goes through the following steps:

1. Download the latest release Go 1.4.2 package that corresponds to your operating system: https://golang.org/dl/ and save it to your drive
2. Create a workspace directory in which all your future golang code and code imported will go to.
   $ mkdir $HOME/go
   Create three folders inside it: $mkdir bin src pkg
   Bin is where go programs are installed, pkg is where package files are installed and src is where source files are kept
3. Set the GOPATH from the terminal by editing the ~/.bashrc file and assigning to it the workspace directory just created.
   $ export GOPATH=$HOME/go
4. Set the workspace’s bin subdirectory as the value of your PATH variable
   $ export PATH=$PATH:$GOPATH/bin
After setting up the go environment, you can start creating your programs inside src folder. If your program needs to import a non-standard library package, you need to get the code and install it in your src folder by running the following statement into your terminal:

```
go get -u <import_path>
```

The majority if not all packages needed are open source but make sure to install git in your machine and have a git account before proceeding. The easiest and most efficient way to take full advantage of go’s high performance is to use a text editor instead of considering the usage of an IDE. After writing the program inside a folder in src, you can build the program from the terminal using go build and run it using `./programName`. You can use go install to put your program in bin folder so that you can run it afterwards like any command line app.

Documentation for writing golang web applications is available in [https://golang.org/doc/articles/wiki/](https://golang.org/doc/articles/wiki/). Also, please refer to the comprehensive e-book on [http://codegangsta.gitbooks.io/building-web-apps-with-go/content/index.html](http://codegangsta.gitbooks.io/building-web-apps-with-go/content/index.html). Interesting web application frameworks that can be used to build web apps easily and quickly include Beego, Revel, Martini and Traffic. Since I already covered Martini earlier in the implementation part, I will focus on Beego to encourage future work on it since it is believed to be more advantageous than Martini. “Beego integrates features belonging to Go and great mechanisms from other frameworks.”[6] The biggest flexibility enabled by this framework is enabling modular design and giving to the developers the freedom the choice of which modules are interesting for the purposes of their applications. The interested reader may refer to [http://beego.me/](http://beego.me/) to get started on this methodology.

Using Beego requires the installation of Beego and the Bee dev tool from the terminal:

```
$ go get github.com/astaxie/beego
$ go get github.com/beego/bee
```

[7] explains the process of using this methodology after completion of installation, to write your first program: “

1. We import package github.com/astaxie/beego. As we know, Go initializes packages and runs init() in every package ([more details](http://beego.me/)), so beego initializes the BeeApp application at this time.
2. Define the controller. We define a struct called MainController with an anonymous field beego.Controller, so the MainController has all methods that beego.Controller has.

3. Define some RESTful methods. Due to the anonymous field above, MainController already has Get, Post, Delete, Put and other methods, these methods will be called when user sends a corresponding request (e.g. the Post method is called to handle requests using POST. Therefore, after we overloaded the Get method in MainController, all GET requests will use that method in MainController instead of in beego.Controller.

4. Define the main function. All applications in Go use main as their entry point like C does.

5. Register routers. This tells beego which controller is responsible for specific requests. Here we register MainController for /, so all requests to / will be handed by MainController. Be aware that the first argument is the path and the second one is pointer to the controller you want to register.

6. Run the application on port 8080 as default” [7]

Appendix D: Honors Research Project

E-Health in Morocco: Challenges, Opportunities, Implications and Recommendations for the Implementation of a National E-Health Strategy

Abstract

E-health is a promising cyber application, and it is attracting substantial interest as it concerns the well being of every citizen worldwide. E-health applications have gained substantial amount of attention during the last few decades being the single-most important revolution in healthcare. It encompasses a wide variety of functions related to the medical field, to the interaction between its different actors and the delivery of clinical information in a timely and efficient manner leading to the enhancement of the whole clinical decision making process. Morocco is among the countries, in which healthcare services are lacking,
which negatively impacts its human development index. It is facing many challenges, which undermine its ability to provide accessible, efficient, and equitable services to its citizens. In this paper, I will be investigating the context of healthcare in Morocco, its challenges and strategic implications in the long run to better understand what kind of e-health solution is more adequate and realistic for the Moroccan citizen. This is to be achieved through a needs assessment study based on secondary data provided by ministry of health and other international organization health reports. Then, those e-health solutions and strategies are analyzed in terms of their economic efficiency, social and ethical implications. A list of recommendations to address and reduce the risks of ethical issues is to be devised for future consideration.

Background

Morocco is ranked as 129 among 187 countries in total in 2014 with a human development index of 0.617. [14] This low index is impacted by the low quality of services provided by the Moroccan healthcare sector. The difficulties, which impede the development of healthcare in Morocco, include structural weaknesses, insufficient human resources and low budget allocated to this vital sector. With “a hospitalization rate reaching only 5%, 0.62 doctors allocated per every 1000 habitants and 0.97 nurses per 1000 habitants” [4], Morocco is among 57 countries that suffer from an acute shortage on Human Resources for Health. [5] This situation calls for the exploration of emergency plans to deliver efficient healthcare services, especially in remote places in which the highest rates of infant and maternal morbidity and mortality are recorded. “The principal causes of maternal deaths are hemorrhage (54%), eclampsia (24.2%), infection (8.1%), abortion (5.6%)”. [9] According to the same source, of the total of audited maternal deaths, 48.3% never consulted before giving birth and only 5.4% of them benefitted from four antenatal visits. This data shows that enhancing the means of interaction and communication between patients and physician is required to improve Moroccan citizens healthcare quality.

E-Health is the set of processes that make use of technological means to improve the quality of the medical care chain. It concerns a wide range of applications, which include
electronic health record systems, telemedicine, consumer health informatics, health knowledge management, mobile health and so on. The use of e-health allows the efficient management of patient information and their medical records, which reduces the amount of delays and disperses the access level among benefiters belonging to different geographic distributions. E-health aims at improving healthcare services through an improved diagnosis and reduction in medical errors through patients records, continuity of care and improved patient safety (BCMA, drug-drug interaction, allergies alerts). [7] In addition to that, it allows for the use of integrated preventive health care tools (clinical reminders), optimization of facility workflow (paperless documentation), minimization of patient’s waiting time (queuing and appointment systems), enabling regional referral centers, improving resources utilization (archiving patient records and X-ray images), controlling waste and duplication (eliminating duplication in dispensing drugs, repeat of lab tests). It is regarded as an efficient tool that helps with supporting research and decision making (comprehensive database of patients, supporting the decision making process by providing necessary and periodic statistics), developing necessary policies for the advancement of healthcare, providing the latest references and research.[7]

Although e-health seems to be a promising venue to be considered by leaders of the health sector in Morocco and despite all the challenges that Morocco is facing and which call for urgent strategic solutions, following e-health strategy in Morocco calls for a calculated approach to integrating this strategy. Hence, needs based assessment along with a comparative analysis of e-health strategies that address those needs are required to mark a surer step towards the implementation of an adequate e-health strategy in Morocco.

**Materials and Methods**

In conducting my research, I started with an initial investigation of the progress achieved so far in terms of e-health strategies already thought of. This includes the study of ICT (Information and Communication Technology) indicators and efforts of e-government plans, which make up obstacles and opportunities to the implementation of an e-health solution. Those serve as restrictors by limiting our scope of thinking about realistic solutions. Then, I opted for needs based assessment to come up with a deeper
understanding of Moroccan expectations. Data for this study was collected from secondary sources namely the websites of Moroccan Ministry of Health, “Ecole Nationale de Santé Publique” and the Union for the Mediterranean Digital Agenda. The analysis of the results of this assessment allowed me to see what is the best approach to integrate e-health in Morocco’s e-government strategy. This helped in conceiving which concept makes more sense to be adopted for Morocco and which maximizes the benefits in the long run. Then, the e-health solution is evaluated by foreseeing its social, ethical and economic implications in the long run. Recommendations for enhancing the privacy and the security of clinical records from theft and manipulation are suggested based on literature review combined with my own knowledge and experience.

Analysis

Maroc Numeric is an e-government strategy which “started in 1997 and aims to promote the information society in Morocco”. [15] It targets different sectors by financing innovative applications and projects. “The government has budgeted 2.5 billion Dirham (0.22 billion Euros) between 2006 and 2010”. [15] In addition to improving administration’s performance, helping with decision-making, this strategy aims at providing quality services to citizens with diminished delays, more transparency in the administration management and to decrease distances between the government and the citizen. This trend is growing in Morocco and concerns healthcare sector as well. “The 2020 health vision produced by the Ministry of Health of Morocco with the World Health Organization support […] identified eight areas of action.”[15] The particularity of those action plans is that none of them clarifies an e-health strategy to be adopted. Some efforts have been made to integrate medical informatics as a tool to be used by health professionals to increase work efficiency and to help in taking decisions. The Laboratory of Medical Informatics has been established in Casablanca in 2005 and it is the first academic center in Morocco, which trains doctors specialists in medical informatics.[8] This institute first and unique of its kind in Morocco aims at shortening the bridge between health and ICT professionals through effective, multidisciplinary cooperation which focuses on Mobile Health (mHealth) more than any other approach to e-health. Some applications
already implemented include an antibiotic guide on Android and iOS plate-form and an application for measuring stress.

So, in terms of e-health opportunities in Morocco, Maroc Numeric strategy encourages efforts oriented towards innovation in different sectors. Academic training of medical informatics specialists has gained the interest of healthcare sector actors. However, in terms of obstacles, Morocco is faced with the restriction of illiteracy and the low personal rate of computer penetration especially in rural areas. Morocco is not prioritizing health sector. “Morocco only dedicates 1.1% of its GDP to public health expenditure, versus 2% to 25% in countries with similar per capita income.” [15] Also, despite all the efforts invested so far, the usage of ICT in healthcare is still in its earliest stage compared to other industries.

**Results**

The critical domains of action in the Moroccan healthcare sector include infant and maternal health. Analysis of the factors responsible for mortality among this category shows that lack of communication between future mothers and their doctors plays an important role in the deterioration of quality of healthcare services delivered. This applies to women in remote areas where they cannot have access to medical attention. The factor of time plays a critical role here since any delay can have serious consequences. For this reason, it seems that what this category needs is a platform through which they connect to professionals to seek their advice and get answers for their questions without delays and from anywhere they happen to be at the moment. Another need is to have an organizer of appointments that the pregnant woman needs to attend before giving birth, which sends reminders when the time of the appointment is close. Since Morocco has a low workforce rate allocated per citizen, the most desirable need of Moroccan citizen is to have remote access to health services and to be able to manage their appointments online.

Implementing a global and national e-health solution in Morocco needs time and funds. Important collaboration between healthcare sector actors and ICT actors is needed to sustain activities related to that field. A global information management system is needed
to ensure the smooth flow of information and firm monitoring of patient records. This could apply to all healthcare stakeholders in the healthcare chain. To take part in the elaboration of a new paradigm for healthcare, Morocco should think about a quick solution instead of waiting for all obstacles to be erased. Indeed, nowadays going for an e-health approach requires only a personal computer connected to Internet to work on clinical care, health services, and information research.

This relatively inexpensive solution could build upon a better future by not wasting time and taking advantages of tools at hand. It is also going to encourage the gradual adoption of this practice as its benefits are going to be lived and witnessed closely during daily operations. A lot of health practitioners will feel reluctant at first to use technology because of its complexity; although, it is not a matter of complexity but of familiarity. Getting started with the usage of e-health in its simplest forms can make huge difference into building the acceptance and credibility of this approach in healthcare. Thus, they can start by having access to updated and complete information of their patients and his current situation, as well as the history of treatment. This allows communicating the information via Internet to other healthcare services such as laboratories and the results could be transferred directly to the prescribing physician and the patient. After a short period of familiarization, the efficiency and productivity of this approach compared to the traditional paper based approach are observed and it will become hard to give up on the new approach. This solution assumes that all practitioners are equipped with a personal computer with an access to the Internet and a number of other basic software such as Word, image processor, spreadsheet software applications and so on. Practicing with this approach will allow practitioners to form a closer idea of their real needs in terms of user interfaces and functionalities.

Another important consideration is to encourage the cooperation between healthcare sector actors and ICT professionals from one side and between Morocco and other international organizations. This will allow for elaborate supervision and organization of efforts towards global development.
Implementing an e-health strategy that satisfies those needs has many implications economically, socially and environmentally. On the long run, an e-health strategy has economic implications. Reliance on electronic medical record can result in tremendous savings and increased productivity. The ability to get access to all information needed on a specific patient quickly anywhere and with low error probability improves without any doubt the quality of healthcare. “The ability to receive health maintenance alerts and notification of potential drug interactions, to graph results, to instantly access information anywhere at any time, to improve communication between providers, and to have the assurance of a complete patient record, all will play a major role in improving patient outcomes.”[13] Socially, e-health strategy can enhance the communication between patients and doctors by providing a common platform in which information is shared efficiently and patients can contact their doctors anytime and anywhere. Healthcare scope is extended beyond its conventional geographic boundaries. However, on the other hand, this strategy can create inequalities between people based on their level of knowledge of how to use technologies. The biggest challenge is that the illiteracy rate is high in Morocco, which compromises the efficiency of this solution. It could emphasize the disadvantage of poor people who don’t have the money and access to computers and smartphones; those are the least likely to benefit from this strategy. Environmentally, keeping data stored in computers relieves health practitioners from the necessity to use papers. Paper work is responsible for environmental crisis as it wastes natural resources. However, storing data in servers generates great amount of heat, which is responsible for environmental issues as well, which creates an environmental dilemma.

The legal and ethical implications of an e-health strategy are summarized in the expression “Hippocratic Bargain” which means that efficiency of services is gained at the expense of privacy being jeopardized. “The patient-physician relationship has expanded to the point that private patient information passes through countless hands from treatment to billing to insurance filing.”[10] Loss of privacy can be reached through a number of ways:

- “Poorly designed user interface account for unintended adverse consequence leading to decreased time efficiency, poor quality of care and increased threat to patient safety.
• Inaccurate representation of the patient’s current condition and treatment occurs due to improper use of options such as “cut and paste”.

• Drop down menu and disposition of relevant information in the trash. Such menus limit the choices available to the clinician who in a hurry may choose the wrong one leading to major errors.

• Loss or destruction of data during data transfer; this raises concerns about the accuracy of the database as patient care decisions are based on them: medical identity theft.”[12]

Recommendations

To reduce the impact of social inequity caused by the adoption of e-health strategy, the following recommendations should be followed:

• Organizing Training plans to educate people from different ages and backgrounds on how to use computers.

• Forming personnel responsible for helping people having a difficulty to access a service in e-health application, those personnel can be accessed through a help service either online or via telephone.

• Dedicating some budget to distribute computers and/ or smartphones to be used by citizens with critical health conditions.

Some recommendations to overcome or at least reduce the risk of ethical and legal implications of this e-health strategy are as follows:

• Making sure to grant access to only authorized individuals to information by enforcing good encryption practices. This includes a login system with password encrypted using strong encryption algorithms such as 3-DES, AES and so on. “Usage of two-factor authentication system with security tokens and password are helpful in securing EHRs.”[12]

• Using antiviruses and firewalls to protect data from modifications. “Security measures such as firewalls, antivirus software, and intrusion detection software must be included to protect data integrity.”[12]
• Using cloud computing to store portable medical records enables the usage of remote servers which guarantees more security since data management under the control of cloud providers far from the reach of people of interest.

• Adopting certain intuitive practices and measures, for example, practitioners and patients should not share their own login information with anybody, they should always log off after leaving their workstation.

• Conducting routine random audits regularly to check any discrepancies between real data and data provided by electronic information system and to generate detailed listings of modifications, their date, time and logs for every entry inserted or deleted. “The HIPAA Security Rule requires organizations to conduct audit trails, requiring that they document information systems activity and have the hardware, software, and procedures to record and examine activity in systems that contain health information.”[12]

• Setting up a system, which sends up notifications whenever inappropriate access to medical information is recorded to give logs about the date, location and material accessed which could help in detecting the attackers.

Conclusions

All in all, health sector in Morocco is lacking in terms of human resources, infrastructure and dedicated budget. Maroc Numeric is an e-government strategy that encourages promising and innovative projects, which aim at providing quality services to citizens with diminished delays, more transparency in the administration management and decreasing distances between the government and the citizen. E-Health is among the most promising venues, which Morocco needs for its economic efficiency and social benefits. E-Health in Morocco is still under development with no clear venues or genuine dedication. I have conducted a study of Moroccan needs in terms of health in general, since studies of its needs in terms of e-health specifically are still not taken seriously. The common needs include the prioritization of easy and quick access to health services and the increased need to care of pregnant women to reduce the prenatal morbidity among infants. My own vision to implementing a national e-health strategy is to get started quickly by satisfying basic needs and making use of available resources because the main aim is to get health
practitioners familiarized with e-health applications and feel the positive difference between e-health and traditional paper work. E-health has social and social, ethical and legal implications, which require training, education and strong encryption practices. Implementing an e-health solution in the cloud can solve a lot of limitations of a national e-health strategy since this approach guarantees more security and less environmental problems.

References