Automatic Generation of the History of Present Illness in a Telemedicine System

a thesis presented to
the Faculty of the College of Computer Studies of
the Science and Technology Complex of De La Salle University

in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

by

DAYUPAY, Jan Michael I.

Arnulfo P. AZCARRAGA, Ph. D
Adviser

December 8, 2015

Abstract

People in rural areas around the Philippines typically do not have access to a doctor for a medical checkup or complaint, as doctors usually prefer to work in the more urban areas in the cities and large towns. Fortunately, telemedicine is an emerging trend in medical technology that harnesses the power of ICT to diagnose and treat patients at a distance. Indeed, with the introduction of telemedicine, doctors can now provide health-care to patients in remote locations without coming face to face with them. GetBetter is an example of such a telemedicine system that is currently being developed and eventually to be deployed in the Philippines. The GetBetter system serves as an infrastructure that will provide health care to Filipino patients in rural areas. This study seeks to provide a specialized feature of the GetBetter system, with the main objective of designing a medical diagnosis question-and-answer subsystem. The subsystem aims to methodically probe the different possible impressions for the patient's illness, by asking about the various symptoms that are present. Based on the answers, a one paragraph summary, referred to as the History of Present Illness (HPI), is automatically generated. The research applies Natural Language Generation tools, decision-support systems, and database techniques in order to implement the subsystem. The system has been tested using several dozens of inter-related illnesses, some with shared symptoms. The system runs on Android-based tablets.

Keywords: Natural language processing, natural language generation, telemedicine, GetBetter, mobile application, expert system, medical decision support system

Table of Contents

List of Figures

List of Tables

List of Code Segments

Chapter 1 – Research Description

1.1 Overview of the Current State of Technology

Technology has been applied in various fields such as entertainment, business, education, and medicine. Moreover, technology influences the new generation to invent and develop more applications each day. The field of medicine which has been proven to be vital to the lives of many people is one of the most significant application of technology.

All branches of medicine benefit from the fast development of technology and the impact of medical technology on the lives of the people is important and still progressing (Nusslin, n.d). With the help of technology, vast array of equipments and tools have been created for medical purposes, such as stethoscope, x-ray machines, MRI machines, LASIK surgical machines, etc. There are a lot of medical technologies being researched and developed such as biomedical engineering, cell and molecular imaging, minimally invasive surgery and computer aided diagnosis (Nusslin, n.d). Another medical technology being researched nowadays is telemedicine.

Telemedicine provides a structure and technology to bridge the gap between patients, that are located at far flung areas, and medical care. With the help of this technology, doctors can now diagnose their patients online. Also, patient's records will be saved in an online database for the doctor's reference.

An example of telemedicine is the GetBetter system, a cloud-based database and web application where the records of patients are stored for the doctors to see and diagnose. It is an application wherein Filipino patients from far away provinces will be taken cared of by a doctor using the web (Azcarraga, 2014). The GetBetter system will have two types of accounts, one account for the doctor, and another for the nurse/midwife that will be with the patient at a health center. The primary purpose of the account of the nurse/midwife is to gather the information about a patient that will be sent to the doctor. On the other hand, the primary purpose of the account of the doctor, is to display all the information about the patient and do some actions such as video chat, send a diagnosis, or even send the information to a specialist.

The GetBetter system relies on the nurse/midwife's expertise in gathering information regarding the patient's complaint which may miss out certain important information needed by doctors. In line with this problem, GetBetter needs a subsystem that will emulate how doctors interview their patients in order to extract needed information for diagnosis and treatment. Moreover, it will also need a feature that will summarize all the information gathered from a patient to give doctors the ability to judge and quickly address those patients who needed medical care as soon as possible.

With the recent developments of technology, computers are able to interact with humans in a more natural manner. This has become possible with the help of Natural Language Processing or better known as NLP. NLP is a method of communicating with a computer through speech/text. It has been researched and developed for over sixty years now and it has been successful in the following applications: Natural language understanding, natural language generation, speech or voice recognition, machine translation, information retrieval, summarization, spelling correction and grammar checking (Bierman, et al, 1992). Although there are applications of NLP, it is still hard to design due to its complexity and the ambiguity of natural language (Bose, n.d). NLP has a great future ahead of it which will focus more on human interaction to create more user-friendly systems.

NLP has been used in the medical field for many years. Applications of NLP in the medicine field includes the following: transformation of a massive amount of unstructured medical records to a structured one, speech recognition, and machine translation. One major problem of developing NLP systems in relation to medicine is the large number of medical concepts and words (Friedman and Hripcsak, 1999). NLP will play a major role in the field of medicine in the future to come.

1.2 Research Objectives

1.2.1 Main Objective

To design a subsystem of the GetBetter Telediagnosis system that methodically generates diagnosis and constructs the “History of Present Illness” (HPI) based on the question and answer session.

1.2.2 Specific Objectives

1. To gather information about different common illnesses in the Philippines especially among the poor in rural areas of the country;

2. To apply expert system techniques in probing for the possible illness based on the symptom manifested by the patients;

3. To implement various database techniques in extracting important information from a patient database;

4. To utilize various NLG techniques in creating the HPI in a one-paragraph format;

5. To build an intuitive interface for the expert system and HPI generator and;

6. To insert, revise, delete diagnosis questions and answers in the database.

1.3 Scope and Limitations of the Research

The HPI subsystem includes an expert system that will methodically ask a series of questions to the patients regarding their chief medical complaint. It will also provide a Natural Language Generation module that will use the information gathered from the patient to generate the HPI. The HPI will be written as a single paragraph, maximum of 150 words, in grammatically correct English.

The research will not include the diagnosis of doctors for the patient or any other actions of the doctor but it will be stored in the database together with the HPI for future reference.

The questions in the expert system must be related to the chief complaint of the patient and succeeding questions should be based upon the previous answers. Furthermore, the questions will be displayed in English as well as in the Filipino language. It will focus on illnesses common to people living in rural areas in the Philippines. Moreover, it will include the standard for determining the history of present illness which will asks questions such as the severity, duration, quality, etc. The answers to these questions will be probing an impression of the patient's illness and determine the notion of severity.

NLP techniques and tools will be utilized to summarize all the answers and convert it into a well formed paragraph. In addition, if a patient already has a case record in the GetBetter database, all information in the past case records that are related to the current complaint will be extracted and will be added to the paragraph. The paragraph will be written in English and will contain all the information needed by doctor about the condition of the patient. This paragraph will be displayed in the account of the doctor in the GetBetter system.

The subsystem will be implemented as an Android application that can run offline in order to accommodate weak Internet signals in rural areas. In implementing the mobile application, Java, SQLite, and the Android Software Development Kit (SDK) will be used to program the user interface and functions. On the other hand, PHP and MySQL will be used in getting data from the GetBetter cloud database.

There will be two databases that the subsystem will be using, which are the existing GetBetter database and the local database of the device. The existing GetBetter database will be altered minimally by adding new database tables and adding a connection from the subsystem's new database tables to the existing tables particularly the *tbl_case_records*.

The research will assume that the nurses/midwives or the patients will not be computer literate so the user interface of the mobile application will be intuitive or user-friendly. There will be instruction messages that will guide the users throughout the mobile application. Moreover, the title headers, buttons, and other objects will be properly labeled and colored. The question and answer module will be simple by letting the user answer the questions by tapping on buttons.

There are many NLP techniques that exist such as machine learning, POS tagging, NER tagging, parsing, decision trees, etc. Not all of the present techniques will be applied in this system but only those that can yield the best results in extracting the most important information. The two most important major tasks of NLP that will be used in this research will be Natural Language Understanding and Natural Language Generation.

1.4 Significance of the Research

The GetBetter project aims to provide medical care to people in remote areas where there are few hospitals and even possibly no doctors at all. As a subsystem to the existing GetBetter system, is meant to aid doctors in diagnosing the illness of the patients. The HPI system includes an expert system, in the form of a medical decision-support system that will ask patients about their current health conditions and methodically probe about related symptoms. The expert system will emulate how doctors interview their patient thereby not depending on the expertise of the nurse/midwife in asking questions. As such, the patient would be able to provide important information that the doctor needs. Furthermore, the expert system would provide training to the nurses/midwife on how to interview patients to extract needed information without being conscious of it. It will also teach them what are the signs and symptoms of certain diseases (e.g. Dengue, Measles, Typhoid, etc.). With the system as it is designed, not only the patients “get better” but so would the nurses or midwives in terms of improved skills and increased in medical knowledge.

The collected answers to the questions, the medical impressions that are formulated by the expert system, and related information from past case records, serve as the basis for creating the HPI paragraph. The paragraph will provide doctors with an organized and brief description of the patient's problem. The research will thus help the doctors in determining immediately the kind of medical attention needed by the individual patients.

The data stored in the database particularly the generated HPI can be used for data analytics in the future. These data could be studied and analyzed in order to discover useful information, conclusions, and solutions. Using data analytics in the medical field could provide better health care solution.

1.5 Research Methodology

1.5.1 Data Gathering and Consultation

During this phase, information regarding on how medical specialists or doctors ask their patients regarding their conditions will be researched. Moreover, information on how to write the history of present illness will be gathered. Lastly, information regarding the different common illnesses of people in rural areas are suffering from will also be gathered. The use of different Internet sources and medical books will be used as reference. Doctors and medical practitioners will also be consulted to gather all the

information needed.

In addition, different NLP techniques and existing NLP systems that could aid in the implementation of the system will be researched. Past and related works on NLP will be utilized as reference. Experts in the area of Natural Language Processing from the faculty of the College of Computer Studies will be consulted.

1.5.2 System Design

The overall design of the system will be drawn. This includes the database design, user interface design, and the program flow. The possible NLP techniques that will be used in the system will be chosen in this phase. Furthermore, the most vital part to be done in this phase is the design on how this research will be integrated to the existing GetBetter system.

The database design of the research is an addition to the existing database design of the GetBetter system. The existing database will be minimally altered and will be connected to the database of this research. The database of this research will contain the following:

1. HPI plus the diagnosis of the doctor and treatments;
2. The questions for the expert system in English and Filipino language;
3. The impressions table that will contain all the diseases
4. The symptoms table that will contain all the symptoms for the diseases listed on the impressions table
5. The answers to the expert system;

The user-interface of the expert system and HPI will follow the existing design of the GetBetter system. Figure 3.1 shows the initial design for the expert system while Figure 3.2 shows the HPI design.

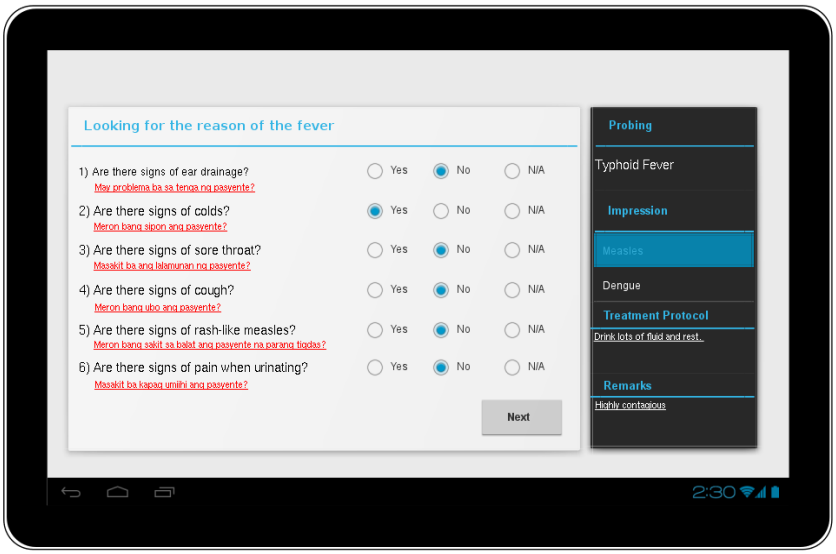

Figure 1 Expert System Initial Design

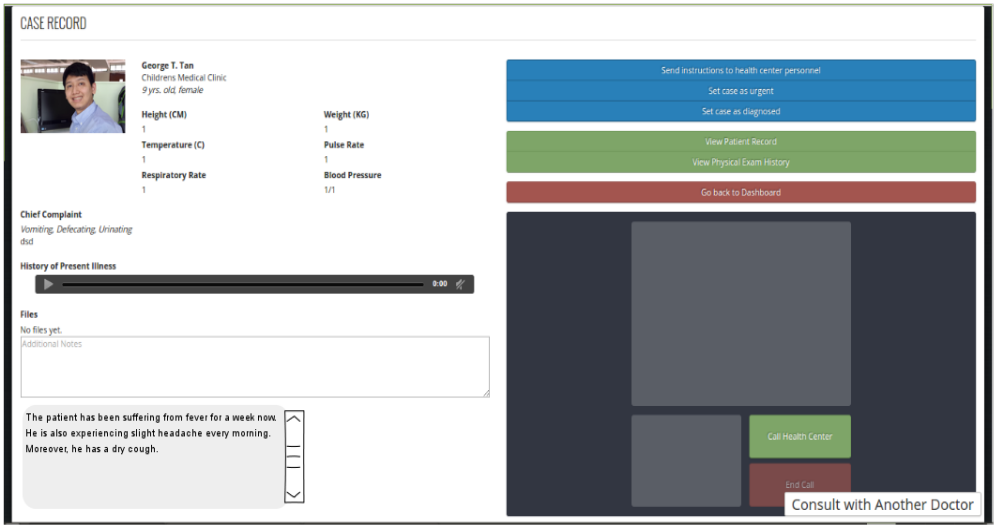

Figure 2 HPI Initial Design

1.5.3 Implementation

The mobile application will be developed during this phase using Java, SQLite, and Android SDK. PHP and MySQL will be used in the server side parsing of data from the GetBetter cloud database. Unit testing and integration testing will also be done during this phase in order to ensure that the system has no errors when integrated to the existing GetBetter system. Regular consultations and updates will me be made with the developers of GetBetter in order to improve the system

The research will follow the Agile Software Development Model. In following this model, the implementation will be incremental and the releases will be tested every iteration to ensure that the objectives of the research are being accomplished. Furthermore, doctors, medical practitioners, and Dr. Arnulfo Azcarraga will be constantly consulted.

1.5.4 Validation and User Testing

The system will be tested by at least 3 volunteer doctors every version release. The volunteers will be asked to answer a survey regarding the performance and features of the system. The answers to the survey will be noted for the revision of the system and then it will be tested again. An average score of 80% on the survey will be the basis of success of the research.

Chapter 2 – Review of Related Literature

2.1 Expert Systems

This section discusses different existing Expert Systems mostly related to the medical field. This section will also discuss the design, implementation, and output of each expert system.

ONCOCIN is a rule-based medical expert system for oncology protocol management that aids physicians in the treatment of cancer patients that was developed at Stanford University. It has two principal modules which are the Reasoner and the Interviewer. The reasoned is a rule-based expert system while the Interviewer is an interface program that interacts with the physicians (Shortliffe, et al., 1979).

Question Answering System to Answer Procedural and Causal Questions in Medicine or QA4MED is a system that gives a straight answer in natural language to a question about medical knowledge (Daella, Dimayacyac, Liao, Lim-Cheng, & Velarde, 2011). With a lot of medical information found in the Internet, searching through them and finding the information needed takes a lot of time which is a bad thing especially in emergency situations. QA4MED helps users by returning an answer, that is retrieved from different medical documents online, as quick as possible. Moreover, this system deals with commonly asked questions to medical professionals in the emergency room and in the barrios (Daella, Dimayacyac, Liao, Lim-Cheng, & Velarde, 2011). QA4MED has four major modules: 1) Question Analysis, 2) Passage Retrieval, 3) Answer Extraction, and 4) Answer Representation.

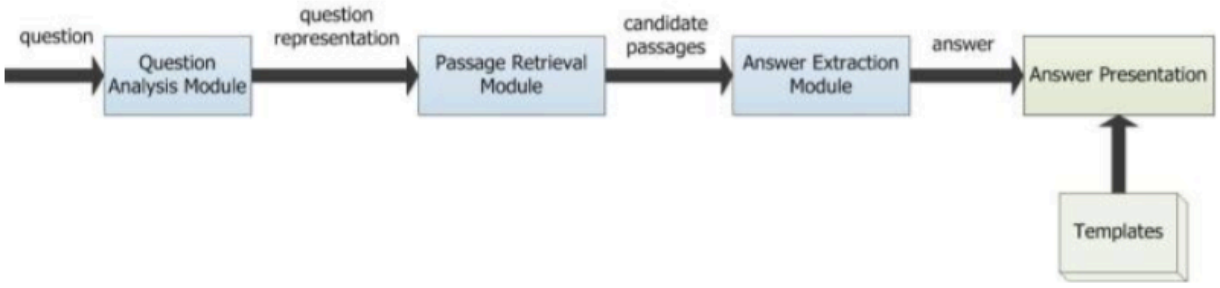

Figure 3 System Architecture of QA4MED (Daella, Dimayacyac, Liao, Lim-Cheng, & Velarde, 2011)

The Question Analysis module is tasked to produce a representation that contains all information needed from the question of the user. This representation is then passed to the Passage Retrieval module, which queries the indexed documents database to retrieve candidate documents which are related to the question. It will then rank each passage by computing a score to retrieve relevant passages. The Answer Extraction module extracts ranked answers and then passes it to the Answer Presentation module to display the answer to the user (Daella, Dimayacyac, Liao, Lim-Cheng, & Velarde, 2011).

2.2 Natural Language Generation

This section discusses different Natural Language Generation systems.

Author	Description of Work	Techniques Used	Results	Future Work
An Ontology-Based Approach to Natural Language Generation from Coded Data in Electronic Health Records [Arguello M., et al. (2011)]	Shows the benefits of an ontology-based approach to generate natural language for the history of present illness from structured clinical texts for diagnosis and clinical management of diseases within the domain of Acute Red Eye.	Ontology, content determination, discourse planning, sentence aggregation, lexicalalisation, referring expression generation, linguistic realisation	The system can reduce the delay in creating consultation notes.	Extending the ontological model to include the predefined templates and aggregation rules
A Novel Approach for Construction of Sentences for Automatic Story Generation Using Ontology *[Jaya A. & Uma G. V. (2008)]*	Automatic construction of sentences for story telling using ontology	Ontology, Conceptual wording, syntactic selection, ordering, morphological generation		Construction of all possible types of sentences with different tenses and avoiding ambiguity.

Table 1 Natural Language Generation System Summaries

Natural language generation or NLG is a subfield of artificial intelligence and computational linguistics that produces natural language texts in English or any other languages from a given knowledge base (Reiter & Dale, 2000). It is the opposite of natural language understanding but both applies almost the same techniques. NLG systems follows a popular architecture that consists of a three – stage pipeline which are “Text Planning”, “Sentence Planning”, and “Sentence Realisation” (Arguello, et al, 2011). In addition, there are six basic kinds of activity that needs to be done in order to generate a natural text from an input and these are: content determination, discourse planning, sentence aggregation, lexicalisation, referring expression generation, and linguistic realisation (Reiter & Dale, 2000).

2.2.2 Template-Based Generation

This section discusses template-based generation, a Natural Language Technique that uses pre-defined template in generating natural language text.

Medical Consultation Annotation and Summary (MedCAS) is template-based generator of medical information relevant to the doctor during a consultation for the personal health information system, Healthbook (Abad, et al, 2013). MedCAS has four major modules which are “Data Retrieval”, “Content

Determination", "Sentence Aggregation", and "Realization".

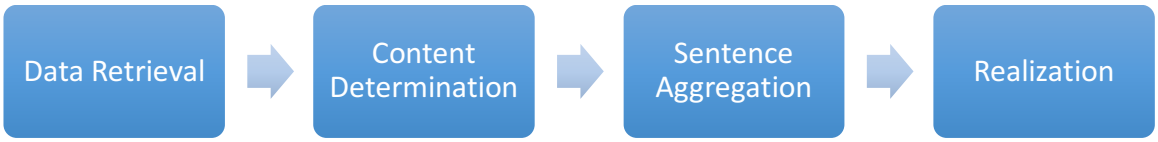

Figure 4 MedCAS Process

MedCAS starts with the Data Retrieval module and gathers data needed for the consultation summaries and annotations from the database. After gathering data needed, the system enters the Content Determination. This module determines the content to be displayed which is decided between three cases: 1) most recent consultation, 2) returning patient's current consultation, and 3) user specified range. After determining the content, the generator enters the Sentence Aggregation module, which merges similar sentences to avoid redundancies and discard sections with missing information. The last step of the MedCAS process is the Realization module. This module handles the varying of sentences, tensing of verbs, and appropriate pronouns. The output sentences is displayed in the consultation area of Healthbook (Abad, et al, 2013).

Chapter 3 – Theoretical Framework

3.1 Expert Systems

Expert system is a branch in Artificial Intelligence that makes computer systems emulate or behave like a human expert in a particular field (Agarwal & Goel, 2014). Expert systems are subdivided into three parts, the inference engine, knowledge base, and the user interface. The knowledge base forms the human expert's knowledge consisting of the sets of rules and facts. The inference engine is in charge of the algorithm of deriving new knowledge from the current knowledge by searching the knowledge base. In addition, the user interface lets the user communicate with the system. (Lewis & Griffin, 1989)

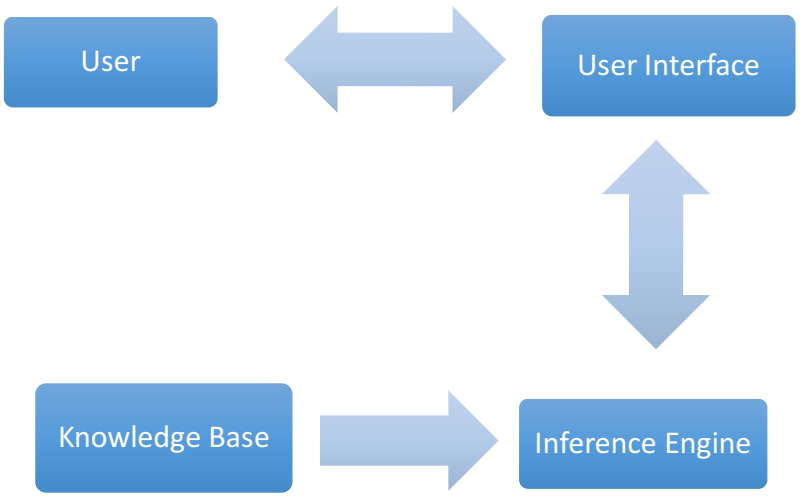

Figure 5 Expert System Architecture

3.2 Natural Language Generation

Natural language generation is a sub branch of artificial intelligence and computational linguistics that concentrates on computer systems that can generate understandable text in a variety of human languages (Reiter & Dale, 2000).

According Reiter and Dale (2000), the general concept of Natural language generation is divided into three modules with each having specific tasks. These three modules are the document planning module, microplanning module, and the realisation module.

Document Planning Module

1. Content Determination is responsible for deciding what information should be communicated in the output document.
2. Document Structuring is responsible for deciding how the contents should be grouped or structured in a document and how each content should be related to each other.

Microplanning

1. Lexicalisation is responsible for deciding what words to use to describe the content given by the

content determination component.

2. Referring Expression Generation is the component which is responsible for deciding what expressions should be used to refer to entities.
3. Aggregation is the component which is responsible for deciding how the structures created by the document planning module should be mapped onto sentences and paragraphs.

Realisation

1. Linguistic Realisation is the component which is responsible for converting abstract representations of sentences into the real text.
2. Structure Realisation is responsible for converting paragraphs and sentences into mark-up symbols understood by the presentation component (Reiter & Dale, 2000).

Figure 6 Natural Language Generation System Architecture (Reiter & Dale, 2000)

3.2.1 Template–based Natural Language Generation

A Template-based NLG system maps the raw data input directly to the linguistic surface structure without intermediate representations (Deemter, Krahmer, & Theune, 2005). This kind of system creates a fixed template wherein the system will just feed the inputs to generate a human understandable text. Samples of template-based natural language generators are YAG (Channarukul, 1999), DEXTOR (Narayan, Isbell, & Roberts, 2011), D2S (Theune, Klabbers, Odijk, de Pijper, & Krahmer, 2001), and many more.

3.2.2 SimpleNLG

SimpleNLG is a realisation engine Java API for natural language generation systems. It contains methods in generating natural language that offers users direct programmatic control over the realisation process (Gatt & Reiter, 2009). The SimpleNLG Java API can generate grammatically correct simple sentences, by just specifying the subject of the sentence, verb to be used, and the object of the verb as the input to the methods described in the API library. In addition, the SimpleNLG API automates natural language generators' common tasks such as proper placing of white spaces, proper placing of punctuation marks, using the right tense, using the right pronouns, noun-verb agreement, and verb groupings. A sample code for generating a simple sentence is shown on Listing 3.2.2.

```
SPhraseSpec p = nlgFactory.createClause();

  p.setSubject("Mary");
```

```
    p.setVerb("chase");
    p.setObject("the monkey");

    String output2 = realiser.realiseSentence(p); //Realiser created earlier
    System.out.println(output2);
```

The resulting output is:

Mary chases the monkey.

Code Segment 1 SimpleNLG Generating a Simple Sentence

3.3 Android Development

The Android Operating System is the mobile operating system that was created by Google (Todd, 2014). Android was first launched into public in 2007 by Google. It is based on the Linux kernel and is designed to be installed on touch screen mobile devices such as smart phones and tablets. As of December 2015, the Android Operating System has 13 versions and Android version 6.0 or Marshmallow is the most recent version. All versions of Android are named after desserts such as Ice Cream Sandwich, Kit Kat, Jelly Bean, and many more. Most people have Android as the operating system of their mobile device wherein they install Android applications downloaded from the Google Play Store.

Developing applications, apps in short, for Android requires knowledge of Java Programming Language and XML or Extensible Markup Language. It also requires the Android Software Development Kit (SDK), and an IDE or Integrated Development Environment such as Eclipse or Android Studio. Moreover, an Android device or emulator is needed in order to test the application being developed.

Web technologies, such as HTML5 and Javascript, can also be used to develop Android apps. Moreover, programming languages such as C and C++ can be combined with Java to develop Android apps with the help of Native Development Kit (NDK). These programming languages can create great Android apps but Java is still the best language to use in developing Android apps because it is the native language of Android and it can maximize the potential of Android. Java is used to develop the back end of an Android app while XML is used to create the user interface.

```
<LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
     xmlns:android=”http://schemas.android.com/tools”
     android:layout_width=”match_parent”
     android:layout_height=”match_parent”
     android:orientation=”horizontal”>

     <TextView android:layout_width=”wrap_content”
            android:layout_height=”wrap_content”
            android:text=”Hello World!”
            android:textSize=”24sp” />

</LinearLayout>
```

Code Segment 2 Sample Android XML Layout

```
package com.example.android;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);



    }
}
```

Code Segment 3 Sample Android Activity Java Code

Chapter 4 – Medical Diagnosis and HPI Generator Subsystem for GetBetter

4.1. System Overview

The Medical Diagnosis and HPI Generator, a subsystem for the existing GetBetter Tele-Medicine System, which will handle the chief complaint impression diagnosis and the generation of a one paragraph History of Present Illness or the HPI. This system is implemented as a mobile application for Android tablets. The system stores all its data, from patient information to the results of the expert system, in the device's local database storage. The system uses existing database tables from the existing GetBetter system and added new database tables for the two major modules of the system.

The first module of the system is to provide the user medical questions that are answerable by a 'yes' or 'no'. The questions and answers is displayed in English but there is a button to toggle between English and Tagalog language. Depending on the answers to the questions, the system probes and displays plausible impressions that the patient has. This module is part of the midwife/nurse account of the GetBetter system.

The second module of the system is the generation of the one paragraph History of Present Illness. All the positive symptoms, chief complaints, and patient details are combined to generate to HPI and is stored in the database. The HPI generator uses a template-based natural language generation technique to generate a one paragraph History of Present Illness.

The system also includes an account for an admin user. The main function of the admin account is to provide an interface for adding new impressions and symptoms to the database of the system. The impressions and symptoms are added for the medical diagnosis expert system.

4.2. System Objectives

The following objectives must be met by the system in order to function properly:

1. To add new symptoms to the database to be included in the expert system,
2. To add new impressions to the database to be included in the expert system,
3. To display the questions and answers in English and Tagalog,
4. To probe an impression of what disease the patient has based on the answers to the questions,
5. To summarize all answers, impressions, and patient information into one paragraph,
6. To display the one paragraph HPI, plausible impressions, ruled out impressions, and
7. To create new Patient Records and store it in the database;

4.3. System Scope and Limitations

The whole system is operated offline most of the time in order to accommodate the poor Internet connection. The system has two accounts, one for the nurse/midwife and one for an admin. The user account includes the two major modules of the system, which is the medical diagnosis expert system and the HPI generation. On the other hand, the admin account is in charge of the interface for adding new impressions and symptoms to the database. Impressions and symptoms added when added will automatically be included in the medical diagnosis expert system.

The system displays the questions and answers in English but there is a button that will toggle the language of the questions and answers to Tagalog. Users is given two options for their answers to each questions, specifically “Yes” and “No”. Moreover, the user could select more than one chief complaint. The possible impressions that will be listed will be based on the selected chief complaints. The symptoms asked is based on the impression that is being probed by the expert system. If the question has been answered already, it will not be repeated. The impressions that will be probed by the system will be displayed in English. All results, from answers to the generated HPI, is stored on the device’s database storage.

The one paragraph HPI has a limit of 150 words and is displayed in the patient case record page after answering the medical diagnosis expert system. The HPI paragraph that is generated includes the name of the patient, age, gender, his/her chief complaint, and the positive symptoms.

4.4. Architectural Design

4.4.1. System Model and Major Modules

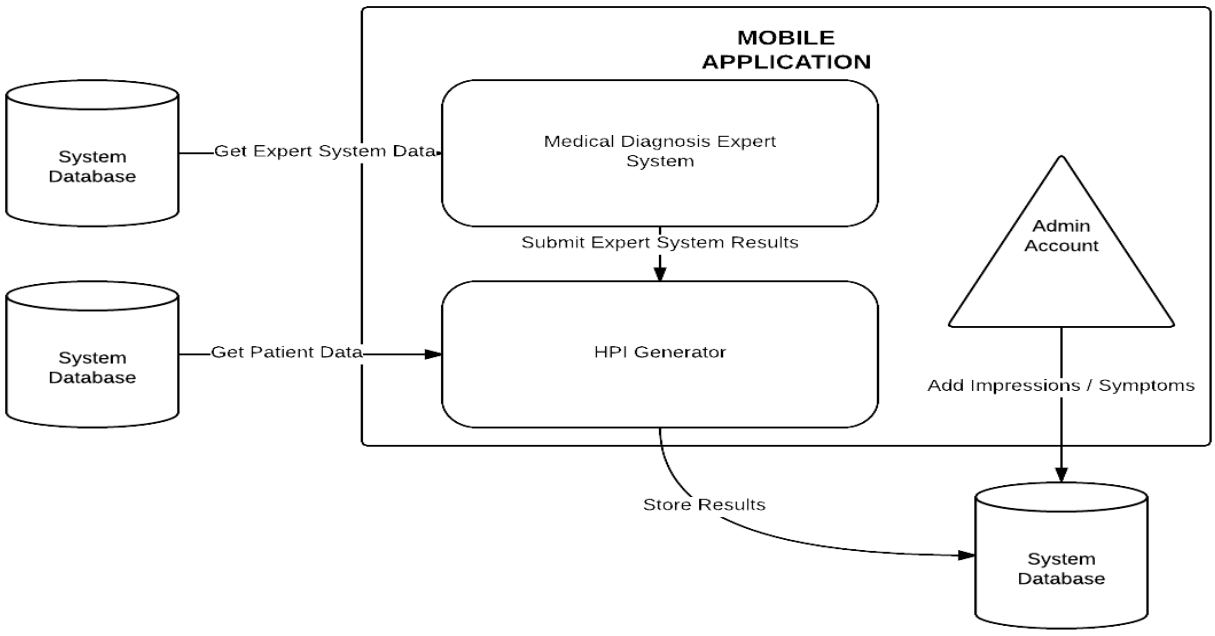

Figure 7 System Overview

The system has two major modules, the medical diagnosis expert system and the HPI generator. The two modules access the database to gather data needed for their functionalities. The medical diagnosis expert system module handles all the user interactions including the selecting of patients, selecting of chief complaint, and answering the questions. After answering all possible questions of the expert system, the system generates the one paragraph HPI and is displayed. All the information, answers to all the questions, plausible impressions, and History of Present Illness, is stored in the local database.

4.4.2. System Flow

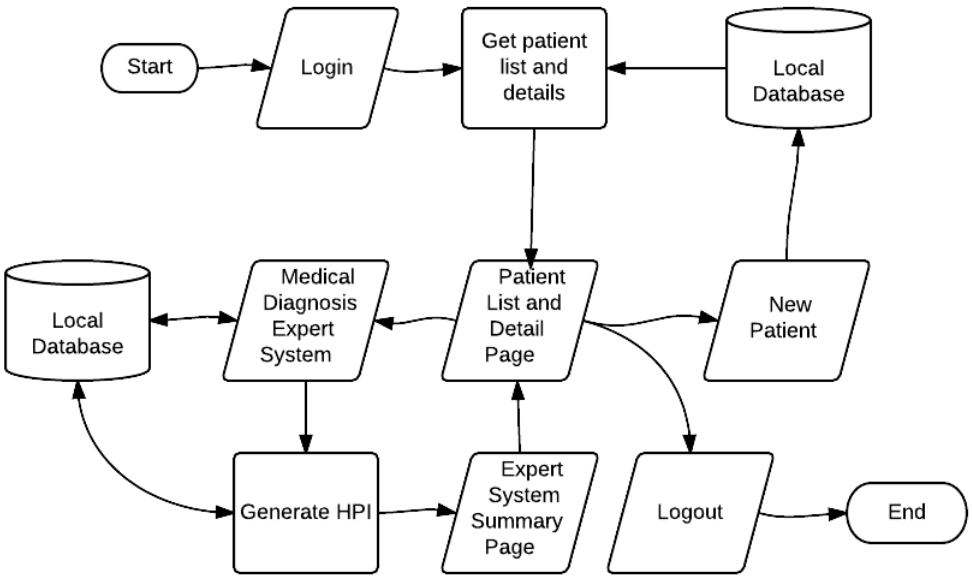

Figure 8 System Flowchart

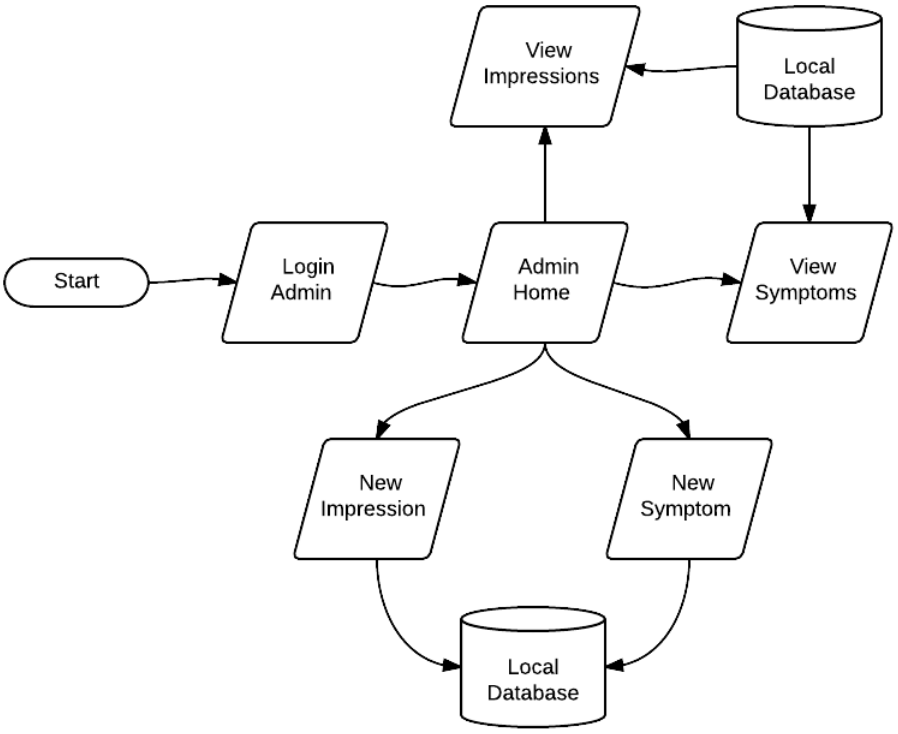

Figure 9 Admin Account Flowchart

4.4.3 Database Design

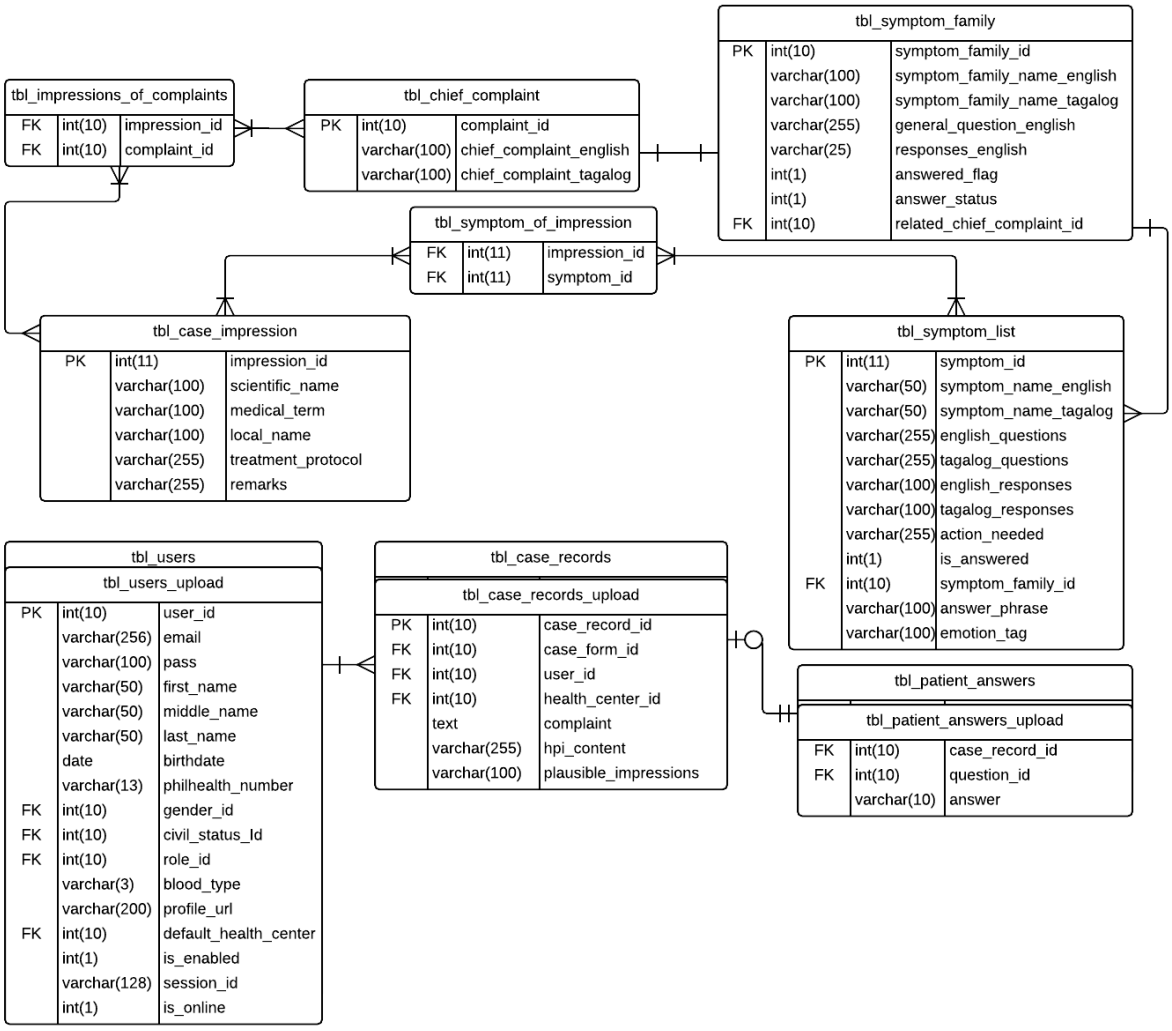

Figure 10 Database ERD

The system is using the existing GetBetter database plus the addition of seven more tables. The added tables are tbl_symptom_of_impression, tbl_symptom_list, tbl_patient_answers, tbl_chief_complaint, tbl_symptom_family, tbl_impression_of_complaints and tbl_case_impression. The *tbl_case_records* from the GetBetter database is altered, adding two more columns which are the hpi_content, and plausible impressions.

Description of each tables:

1. ***tbl_case_impression*** – This table is the storage of all the possible diseases that a patient can have. Stored in it are the medical term, scientific name, local name, treatment protocol, and additional remarks of a particular disease.

2. ***tbl_symptom_list*** – This table is the storage of all the possible symptoms a particular disease can have. Stored in it are the English and Tagalog name of the symptoms. In addition, the questions to ask and actions needed for diagnosing the symptoms are also stored in this table. A column “is_answered” is used to check whether the symptom question has been asked already therefore not repeating it. The column “answer_phrase” is used for the HPI Generation module, wherein this phrase is combined with a subject to create a sentence. Moreover, the column “symptom_family_id” connects a specific symptom to its symptoms family.

3. ***tbl_symptom_of_impression*** – This table contains the primary keys of both tbl_*case_impression* and tbl_*symptom_list*. The purpose of this table is to map an impression and its possible symptoms. An impression can have one or more symptoms. An additional column named, “hard_symptom”, is used for checking if the impression is going to be plausible or ruled out. If at least one hard symptom is ruled out, then the corresponding impression is ruled, otherwise it is a plausible impression.

4. ***tbl_chief_complaint*** – This table contains the chief reason for visit. E.g. Fever, Generally unwell, problem in the skin, bowel movement, and many more.

5. ***tbl_patient_answers*** – This table contains all the patient's responses to the questions in the medical diagnosis module of the system. In addition, included in this table are the foreign keys for the *id for the tbl*_symptom_list and tbl_case_record tables.

6. ***tbl_symptom_family*** – This table contains the symptom family name and question in English and Tagalog language. The symptom family is the parent symptom of a specific symptom. Before asking a specific symptom question, the system checks if its symptom family is answered. If the symptom family is answered, the system checks if the answer is positive or negative. The system asks the specific question if the symptom family is positive and skips the specific symptom if it is negative.

7. ***tbl_impressions_of_complaints*** – This table contains the matching of impressions and chief complaints. The data stored here is the basis for populating the possible impressions list that is going to be probed by the expert system.

8. ***tbl_users*** – This table contains the details of a user. Stored here are the user accounts for admin and nurse/midwife. Also stored here are the patients created in the system. Each user is uniquely identified by the “user_id”. Columns included in this table are the email, password, name, birthdate, gender, civil status, and many more.

9. ***tbl_case_records*** – This table contains the case records of each patient after finishing the medical diagnosis expert system. Each case record is uniquely identified by a randomly generated “case_record_id”. Stored here are user’s id, the hpi, chief complaints, and the plausible impressions.

4.5 System Functions

4.5.1 System Login and Logout

The login screen will be the first screen when the mobile application is opened. This function will restrict access to the mobile application in order to provide security to all the data of the patients. Only registered users are allowed to access the mobile application through this function.

The nurse/midwife will select first the health center that will be accessed before logging in. Once a health

center has been selected, the nurse/midwife will input the username and password then click the login button to access the mobile app. The nurse/midwife can logout of the system by tapping the logout button located at the patient list page. Logging out of the system will end the session of the nurse/midwife and will redirect them to the login page. This function also checks if the credentials that is entered is an admin account. If the login credentials that is entered is an admin, the system redirects the user to the admin page of the system.

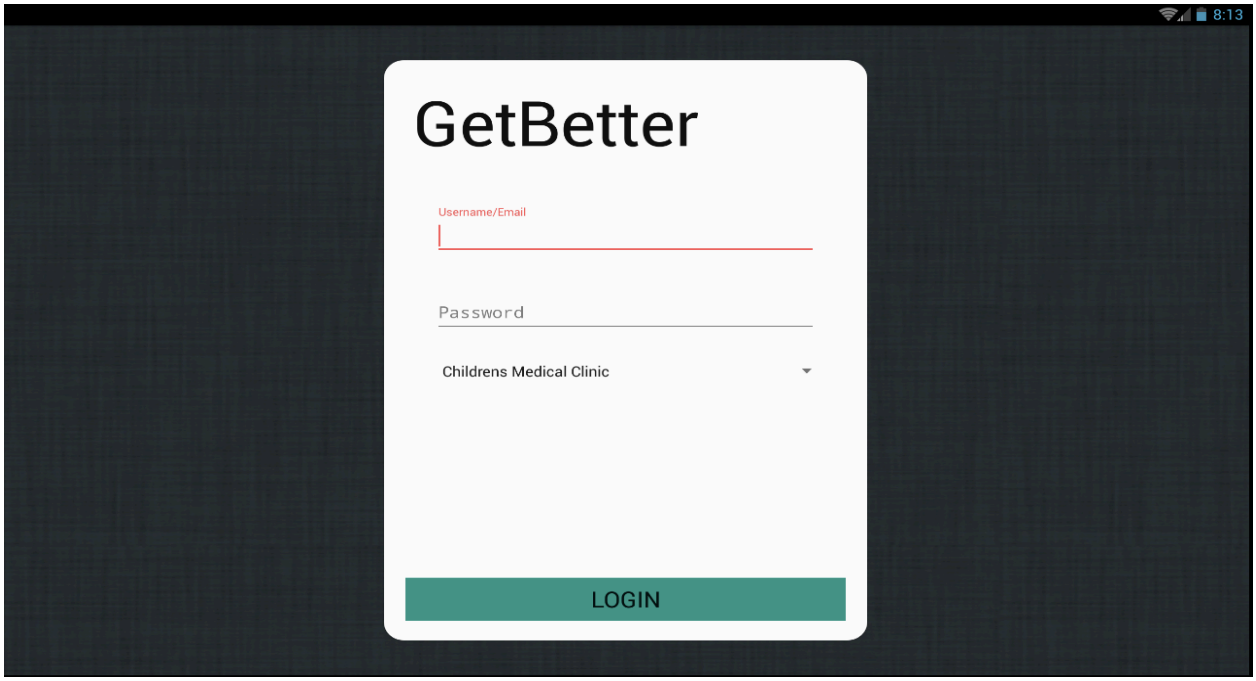

Figure 11 Login Screen

4.5.2 Patient List and Patient Select

This function lists all the patient records created in the system. The nurse/midwife selects a patient from the list on the left side and the selected patient's personal details is displayed on the right. Personal details of the patient include the name, age, gender, civil status, blood type, and address. Moreover, two buttons are displayed when selecting a patient. These two buttons are "New Case Record", and "View Case Record/s". The "New Case Record" button starts the medical diagnosis expert system for the selected patient. On the other hand, the "View Case Record/s" button displays a list of case records of the patient selected.

Tapping the button on the upper right side of the screen, a list of actions is displayed. The actions are "New Patient", "Settings", and "Logout". The nurse/midwife can add new patient record by tapping "New Patient" that redirects to the new patient page. The nurse/midwife can logout from the system by tapping "Logout".

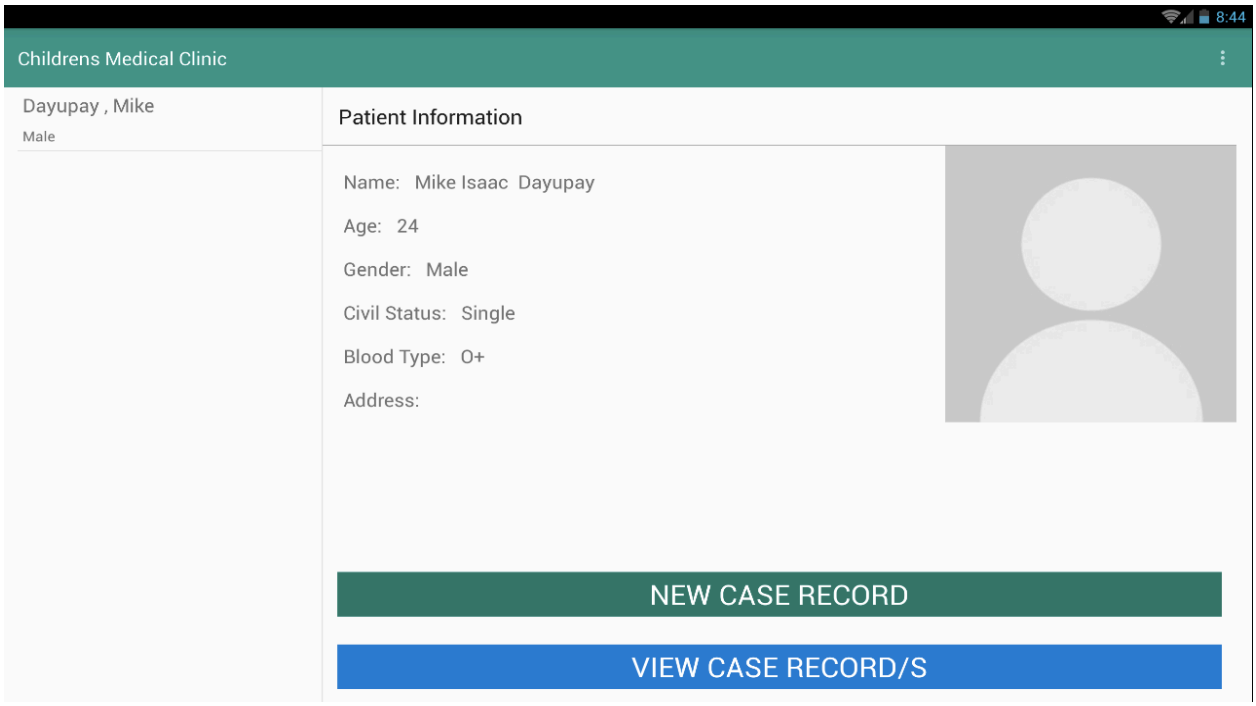

Figure 12 Select Patient Screen

4.5.3 Selecting Chief Complaint

This function is the basis of the expert system on what possible impressions that the system is going to probe. The chief complaints in this function are grouped into check buttons that the nurse/midwife selects. At least one chief complaint should be selected by the nurse/midwife. Examples of chief complaints listed in this function are “Fever”, “Pain”, “Injury/Wound”, “Skin Problem”, “General Unwellness”, etc.

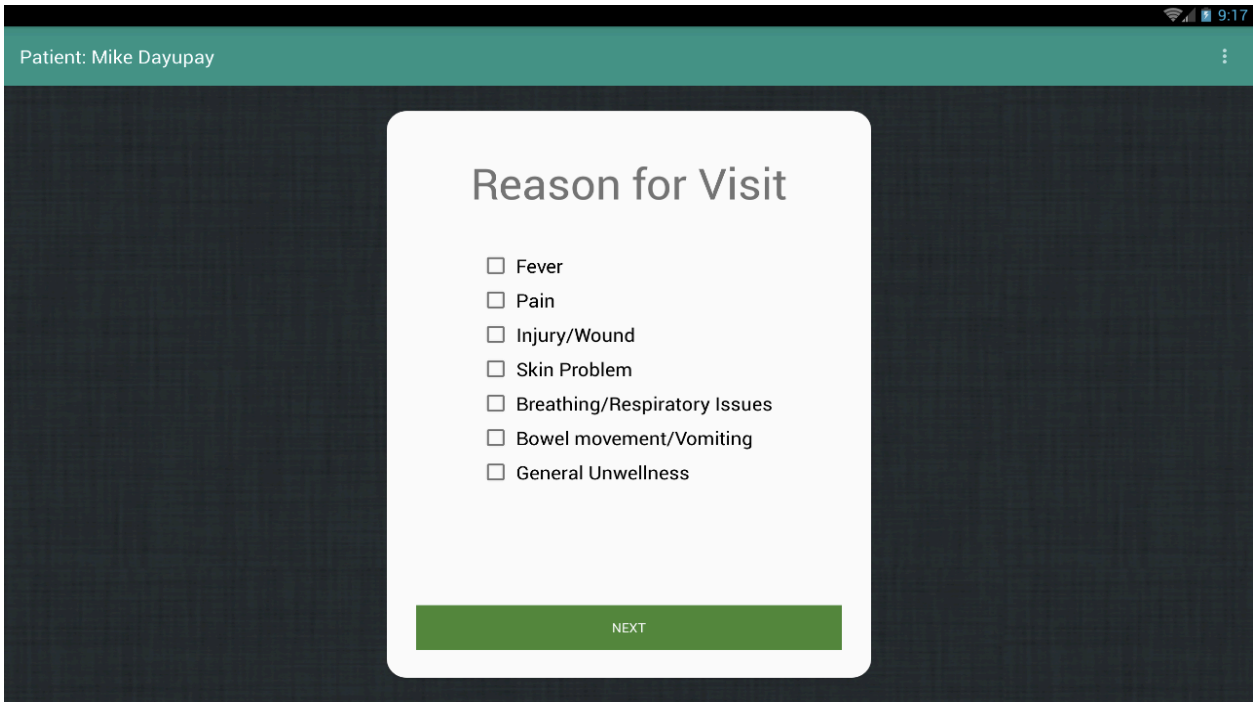

Figure 13 Chief Complaint Screen

4.5.4 Medical Diagnosis Expert System

One of the major function of the system is the medical diagnosis expert system. This function is in charge of asking the medical questions to the patient in order to arrive at plausible impressions of what kind of sickness the patient is having. Moreover, the Medical Diagnosis Expert System loads the questions and impressions from the database.

Each impression that is being probe, has a set of symptoms that the system asks. Each symptom question is displayed one by one and all the questions have two answers to choose from and only one answer can be selected. Furthermore, questions that were already asked is not asked again. If the patient wants to change his/her answer from the previous question, the system provides a button to go back to the previous question. After exhausting the set of symptom of a given impression, the system checks if the impression is plausible or ruled out then it proceeds on probing the next impression.

Along with the question and answers, the possible impressions, ruled out impressions, and positive symptoms are also displayed. After probing the last impression from the possible impressions list, the nurse/midwife is directed to the expert system summary page where the generated HPI, plausible impressions, ruled out impressions, and positive symptoms are displayed.

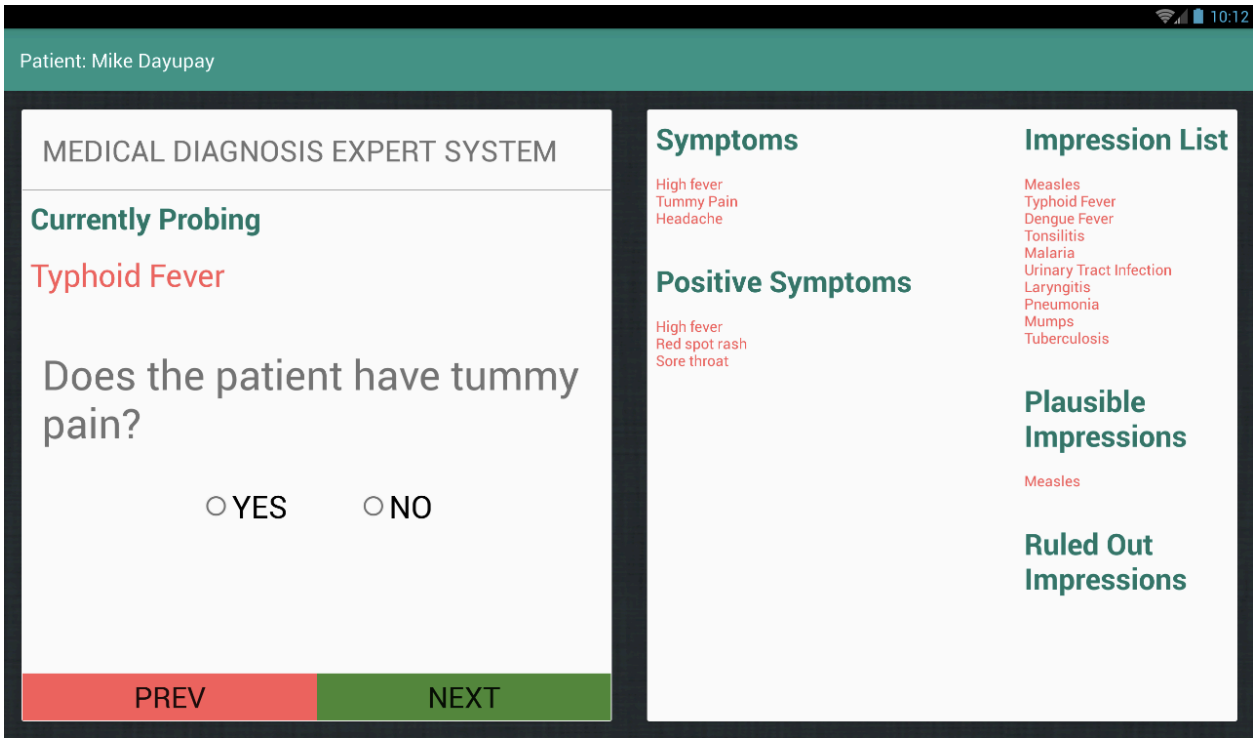

Figure 14 Medical Diagnosis Expert System

4.5.5 Generate HPI

Generating the HPI is one of the major function of the system. This function extracts all important information such as the positive symptoms, patient details, and chief complaints. After extracting important information, this function applies a template-based Natural Language Generation to generate a one paragraph History of Present Illness with a limit of 150 words. The one paragraph HPI is stored in the local database along with answers, and plausible impressions.

4.5.6 Expert System Summary Page

This function is the results page after finishing the medical diagnosis expert system. In this page, the system displays the generated History of Present Illness, plausible impressions, ruled out impressions, and positive symptoms. A button is included that redirects the nurse/midwife to the patient list after checking the summary page.

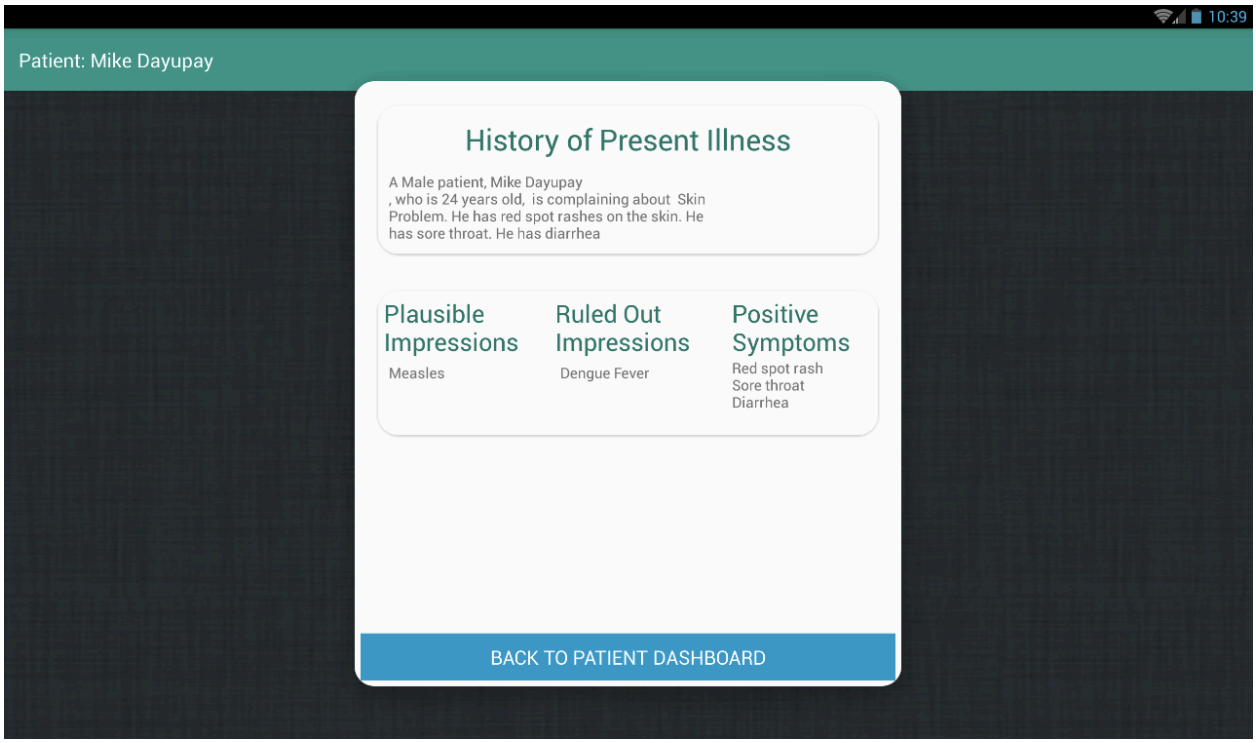

Figure 15 Expert System Summary Page

4.5.7 Admin Account Add Impression

This function lets an admin user add a new impression to the database. In adding an impression, the admin needs to enter the impression's details such as its "Medical Term", "Scientific Name", "Local Term", "Treatment Protocol", and "Additional Remarks". Moreover, the admin is also required to select at least one chief complaint that is associated with the impression being added in order for it to be included in the expert system. The admin also needs to select at least one symptom that is associated with the impression being added in order for it to be probed in the expert system.

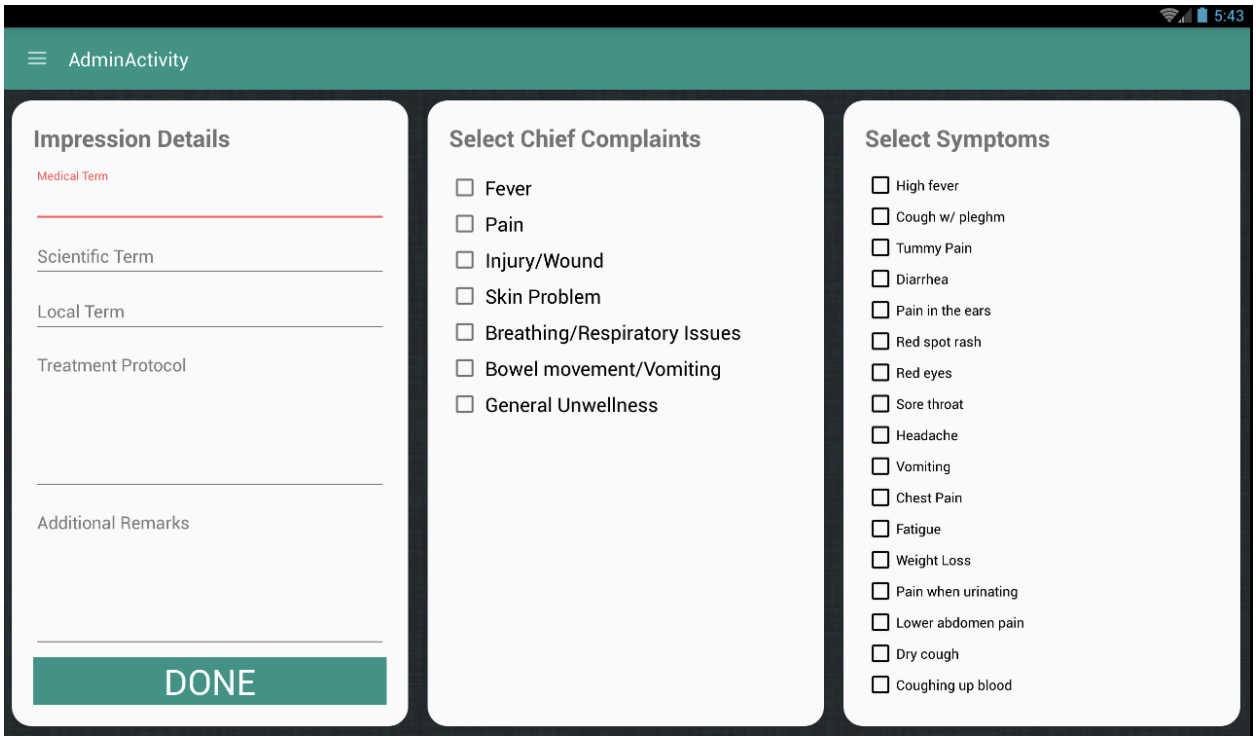

Figure 16 Admin Account Add Impression

4.5.8 Admin Account Add Symptom

This function lets an admin user add a new symptom to the database. This symptom is included in the medical diagnosis expert system. In adding a symptom, the admin needs to enter the “Symptom Name” in English and Tagalog, “Question” in English and Tagalog, “Action Needed”, “Answer Phrase”, and “Emotion Tag”.

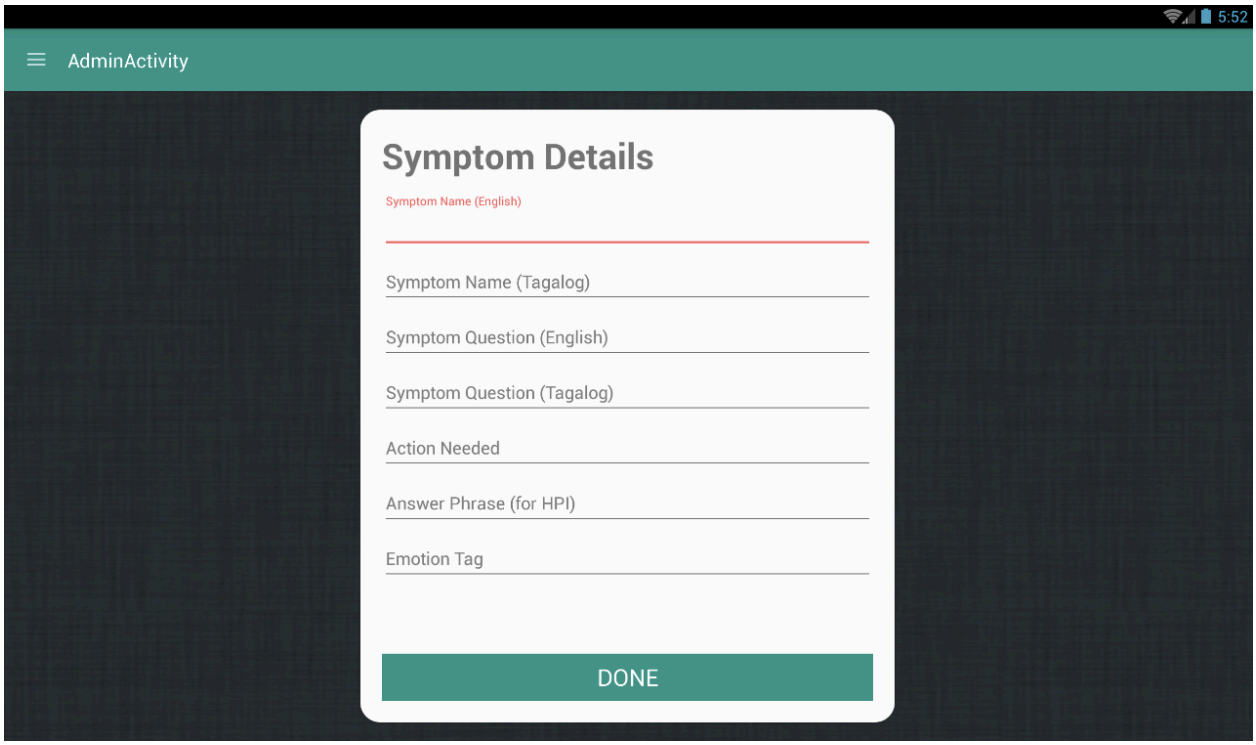

Figure 17 Admin Account Add Symptom

4.5.9 Add New Patient Record

This function lets the nurse/midwife add a new patient record in the database. The nurse/midwife enters the first name, middle name, last name, birthdate, gender, blood type, civil status, and address of the patient in creating the patient's record. Once the patient record is added to the database, it is automatically displayed on the list of patients of the health center where it was created.

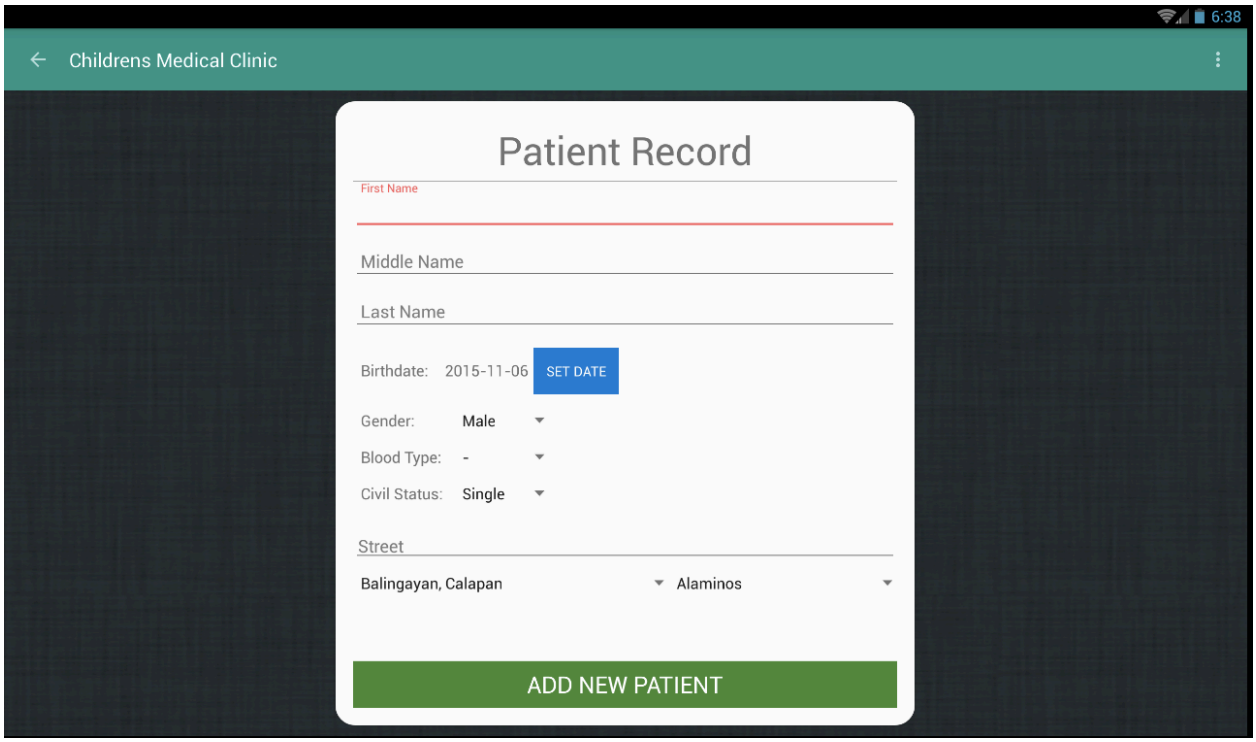

Figure 18 Add New Patient Record

4.6 Physical Environment and Resources

The system will be implemented as a separate mobile application before it will be integrated into the existing GetBetter system so the the hardware resources needed is an Android tablet with an operating system of Android version 4.0 and above. The Android tablet should be a 10-inch tablet because the user interface layout is designed for a 10-inch tablet. The system can operate offline but will need an Internet connection in order to access the database in the cloud to update the data within the mobile application.

Chapter 5 – Design and Implementation Issues

5.1 Medical Diagnosis Expert System

The expert system starts after the user tap on the “New Case Record” button on the patient’s information page. The user will be prompted to select the chief reason for visit, and this will initialize the question and answer system. The initial design for the expert system was following a patient decision flowchart, wherein the question and answer will start probing an initial symptom, answerable by a “yes” or “no”. Based on the answer, the system will proceed into probing another symptom until an impression is captured. This design would require programming a rule-based expert system using PROLOG or programming it manually using a lot of “if else” statements which would be a big scope for this research.

Redesigning the whole algorithm for the expert system was the solution that was employed. The new algorithm for the expert system includes the addition of two new database tables, tbl_symptom_family and tbl_impressions_of_complaints. Moreover, the tbl_symptom_list was modified by adding a new column, “is_answered”.

The new algorithm for the expert system starts off by selecting chief complaints. Based on the selected chief complaints, a list of impressions is queried from the database. The expert system will then probe each impression by asking a list of symptoms corresponding to the impression being probed. After responding to the symptom questions, the system will check if the impression is plausible or ruled out. The expert system will end until all the possible impressions are exhausted. In addition, the most important rule of this expert system algorithm is that symptom questions should not be repeated and this was implemented by adding the column “is_answered” in the tbl_symptom_list.

Before probing a specific symptom, the system checks if the parent symptom is already answered by querying the tbl_symptom_family table. If the parent symptom is not yet asked, the system will get the parent symptom questions and asks it first. If the answer for the parent symptom is “yes”, the system will then ask the corresponding specific symptoms. On the other hand, if the answer for the parent symptom is “no”, the system will skip asking the corresponding specific symptom. Listing 5.1 shows the pseudocode for the Medical Diagnosis Expert System algorithm.

```
GET impressions (chief complaints)
SET impression index to 0
LOOP impression index is less than impression list size
      GET symptoms (impression)
      SET symptom index to 0
      LOOP symptom index is less than symptom list size
           IF parent symptom is asked THEN
                Ask specific symptom question
           ELSE
                Ask parent symptom question
           INCREMENT symptom index
```

```
            END LOOP
      IF impression is positive THEN
            STORE impression in plausible impression list
      ELSE
            STORE impression in ruled out impression list
      INCREMENT impression index
END LOOP
```

Code Segment 4 Medical Diagnosis Expert System Algorithm Pseudocode

5.1.1 Database Background

The medical diagnosis expert system manipulates data most of the time. Every process of the expert system module, it reads data from the database and these data takes time to be acquired by the system. When the data takes time to be retrieved by the system, the app crashes. This happen when the UI Thread waits for the data retrieval operations. To avoid this from happening, performing long running operations in the UI thread is discouraged. Long running operations should be performed in a background thread.

An abstract class, AsyncTask, is used as the solution to the database issue. AsyncTask is an abstract class provided by Android which helps the use of the UI thread properly. This class lets long running operations such as the database retrieval to run in the background and show its result in the UI thread without having to manipulate threads. AsyncTask has four steps:

1. **doInBackground** – codes that perform long running operations goes in this method
2. **onPostExecute** – results from doInBackground is passed to this method
3. **onPreExecute** – this method is called before doInBackground method is called
4. **onProgressUpdate** – this method is invoked by calling publishProgress anytime from doInBackground call this method

```
private class GetDataTask extends AsyncTask<int, void, ArrayList<Symptom>> {

    @Override
    protected void onPreExecute() {
        Toast.makeText(this, “Getting Questions”, Toast.LENGTH_LONG).show();
    }
```

```
    @Override
    protected ArrayList<Symptom> doInBackground(int... params) {

        return getQuestions(params);
    }

    @Override
    protected void onPostExecute (ArrayList<Symptom> result) {

        questions.clear();
        questions.addAll(result);
    }

}
```

Code Segment 5 Getting Questions Using AsyncTask

5.2 Template-based HPI Generator

The generation of the History of Present Illness starts after finishing the medical diagnosis expert system when all the positive symptoms and plausible impressions are stored in an array list ready to be fetched in the expert system summary page. There are three templates for generating the HPI; the Introduction, the Chief Complaints, and the Positive Symptoms. The Introduction template consists of the patient's personal information specifically the name, age, and gender. The Chief Complaints template consists of the patient's chosen chief reasons for visit and the Positive Symptoms consists of the symptom questions that were answered as "yes". These three templates are combined to generate the one paragraph History of Present Illness.

The Introduction template:

"A <insert patient gender here> patient named, <insert patient name here> who is <insert patient age here> years old, is complaining of <insert chief complaints and positive symptoms template here> "

Table 2 Introduction Template

The Chief Complaints and Positive Symptoms template:

The template must start with a chief complaint followed by positive symptoms related to that chief complaint. If there are more than one chief complaint, related chief complaint and positive symptoms are group together.

“Patient has fever, with temperature of 39 degrees. The patient has been suffering from fever intermittently. He also has a skin problem that is a red spot rash. “

Table 3 Chief Complaints and Positive Symptoms Template

The sample template shown in Table 4, starts with the “Fever” chief complaint followed by related symptoms “High Fever” and “Intermittent Fever”. After the “Fever” related symptoms, the “Skin Problem” chief complaint is displayed followed by the related symptom, “Red Spot Rash”.

Chapter 6 – Results and Observations

This chapter discusses the results and observations after testing the system in different scenarios.

6.1 System Functions

The application has seven main functions and each functions manipulates data. This section discusses and illustrates the results of each function.

6.1.1 System Login and Logout

This function will let the user login to the system by entering the username and password. The user is also required to select a health center from the drop down list before logging in.

	Action Needed	Expected Output	Actual Output
1. Login Failed	Input incorrect username or password	System shows an alert saying "Username/Password is incorrect"	System showed an alert saying "Username/Password is incorrect"
2. Login Success	Input correct username and password	User logs in to the application and displays the patients list page.	The user logged in to the application and the patients list page was displayed.
3. Select health center	Tap the health center drop down list and select a health center	A list of health center will be displayed.	Health centers were listed down and the selected health center became the focus.

Table 6.1.1 System Login and Logout Test Script

6.1.2 Patient List and Patient Select

This function displays the patients belonging to the selected health center from the System Login and Logout function. Moreover, this function displays the patient's information when the patient is selected.

	Action Needed	Expected Output	Actual Output
1. Patient List	Log in the application.	Displays the lists of patients belonging to the health center selected.	After logging in the user was redirected to the patient list page and the list of patients was displayed.
2. Select Patient	Tap the patient's name on the list	Displays the patient's information.	After tapping the patient's name on the list, the system displayed the patient's information.

Table 6.1.2 Patient List and Patient Select Test Script

6.1.3 Medical Diagnosis Expert System

Medical Diagnosis Expert System is one of the major modules of the system. The first step of the expert system is the selection of chief complaints. After selecting the chief complaints, the user proceeds to the question and answer proper of the expert system. The initialized list of possible impressions will be initialized based on the selected chief complaints. The symptom questions that will be asked will be based on the possible impressions given.

	Action Needed	Expected Output	Actual Output
1. Select chief complaints	Tap the checkboxes to select at least one chief complain then tap “Next”.	Selected chief complaint will have a check on their check box.	Checkboxes are checked when selecting a chief complaint.
2. Select chief complaints	Do not select or check any of the chief complaints then tap “Next”	System shows an alert, “Please check at least one chief complaint”.	The system displayed an alert saying, “Please check at least one chief complaint.
3. Display Impression List	Select a chief complaint then tap “Next”	System displays the list of impressions based on the selected chief complaint.	Selected “Skin Problem” as the chief complaint, the expert system displayed the impression list containing “Measles”, and “Dengue”.
4. Display Symptoms of Impression	Select a chief complaint then tap “Next”	System displays the list of symptom for the impression being probed.	Symptoms are displayed corresponding to the impression that was probed. Probing “Measles”, symptoms that was displayed were “High Fever”, “Red Spot Rash”, “Sore Throat”, “Dry Cough”.
5. Display the symptom question	Select a chief complaint then tap “Next”	System displays the symptom question with radio buttons for “Yes” and “No” for the currently probing impression.	Question was displayed plus the radio buttons for the responses.
6. Answer a specific symptom question with “Yes”	Tap the radio button labeled “Yes” then tap “Next”	System adds the symptom to the positive symptoms list and is displayed under “Positive Symptoms”. Displays the next question. Update “is_answered” column to a value of 1 tbl_symptom_list.	Currently probing symptom was added to the positive symptom list and was displayed under “Positive Symptoms”. “is_answered” column value was update to 1.
7. Answer a specific symptom question with	Tap the radio button labeled “No” then tap	System adds the symptom to the ruled	Currently probing symptom was added to

“No”	“Next”	out symptom list. Displays next question.	the ruled out symptom list
8. Answer a family symptom question with “Yes”	Tap the radio button labeled “Yes” then tap “Next”	System will update answered_flag to 1 and answer_status to 1 in the tbl_symptom_family. Displays related specific symptom question.	answered_flag and answer_status values are changed to 1. The system asked the specific symptom question related to the family symptom.
9. Answer a family symptom question with “No”	Tap the radio button labeled “No” then tap “Next”	System will update answered_flag to 1 and answer_status to 0 in tbl_symptom_family. Display next symptom question	answered_flag value changed to 1, answer status value changed to 0. Skipped the specific symptom question related to the family symptom and asked the next specific symptom.
10. Rule Out an Impression	Answer “No” on at least one hard symptom of an impression.	System adds the impression in the ruled out impressions list. Displays the impression under “Ruled Out Impression”	Impression was added to the ruled out impression list and was displayed under “Ruled Out Impression”
11. Plausible Impression	Answer “Yes” on all hard symptoms of an impression.	System adds the impression in the plausible impressions list.	Impression was added to the plausible impressions list.
12. Exit expert system	When prompted to submit answers select “Yes”	System inserts answers to questions in the database. Proceeds to the expert system summary page.	All answers to the expert system was inserted in the database along with symptom_id and case_record_id.
13. Exit expert system	When prompted to submit answers select “No”	System disregards all answers and returns to patients list page.	The system went back to the patients list page.

Table 6.1.3 Medical Diagnosis Expert System

6.1.4 Generate and Display History of Present Illness

After answering the medical diagnosis expert system, the user is directed to the summary page where all the expert system's results are displayed such as the ruled out impressions, plausible impressions, positive symptoms, and the generated History of Present Illness.

	Action Needed	Expected Output	Actual Output
1. Display expert system results	Select "Yes" at the end of the expert system when prompted to submit answers	Displays the ruled out impressions, plausible impressions, positive symptoms.	The app displayed the ruled out impressions, plausible impressions, positive symptoms in the Expert System Summary Activity.
2. Display History of Present Illness	Select "Yes" at the end of the expert system when prompted to submit answers	The system displays the generated HPI.	The app displayed the History of Present Illness using the template in the Expert System Summary Activity

Table 6.1.4 Generate and Display History of Present Illness Test Script

Chapter 7 – Conclusion and Recommendations

7.1 Conclusion

This research has met all of its objectives upon its completion. An Android application for the subsystem of the existing telemedicine system GetBetter has been developed and tested, according to the set goals and objectives. The Android application has two major modules, 1) the medical impression diagnosis expert system and 2) the automatic generation of the history of present illness, that were successfully completed. The medical impression expert system is able to generate plausible impressions of what sickness the patient has by probing the different symptoms associated with the impressions. In addition, a template-based natural language generation technique is used in generating the history of present illness. The history of present illness is created by combining the details of the patient's personal details and the results gathered from the patient by the medical impression diagnosis expert system. The system has gone through many iterations of development and testing with the help of medical professionals in obtaining different medical information. In summation, the research has completed the following objectives:

1. Gather information about different common illnesses in the Philippines especially among the poor in rural areas of the country. With the help of medical professionals, online resources, and local medical books, the full set of symptoms for 20 different common illnesses were gathered. Additional details such as the local name, scientific name, and treatment protocol of the different common illnesses were also obtained.

2. Apply expert system techniques in probing for the possible illness based on the symptom manifested by the patients. A question-and-answer expert system was created by listing all the possible impressions or sickness that the patient can possibly have, based on the chief reason for visit. The different symptoms corresponding to the each of the plausible impressions are then probed. The questions are all answerable by "yes" or "no". Moreover, if the question has already been asked, it will not be asked again by the system. After probing a set of symptoms, the system will check if the impression is plausible or ruled out.

3. Implement various database techniques in extracting important information from a patient database. The system executes a lot of database queries when it is running such as getting the patient list, inserting new patients, getting the list of possible impressions, getting the list of corresponding symptoms, inserting the answers, inserting the HPI, and many more. Getting all these data from the database requires complicated database queries. In getting these data, techniques such as SQL Joins, a SELECT statement within a SELECT statement, a SELECT statement within an INSERT statement, and updating of multiple rows, are utilized.

4. Utilize various NLG techniques in creating the HPI in a one-paragraph format. Template-based natural-language generation was used in generating the history of present illness based on the results of the medical impression diagnosis expert system. The template is divided into three parts, the introductory sentence, the chief complaints, and the positive symptoms. The introductory sentence consists of the patient's name, age, and gender. The chief complaints part consists of the chosen chief reasons for visit by the patient. The positive symptoms part consists of all the symptoms that was answered by yes by the patient in the expert system. In addition, pronouns such as "he" or "she" are also generated depending on the gender of the patient.

5. Build an intuitive interface for the expert system and HPI generator. The user interface of the Android application was modeled after the existing GetBetter system. The color theme of the application,

from the background to the buttons, are similar with the GetBetter system. All buttons are properly labelled and colored to indicate what it does. The interface of the expert system displays all the important information that will train the nurses or midwives in diagnosing a patient when using the system.

6. Insert, revise, delete diagnosis questions and answers in the database. In addition to the existing database of the GetBetter system, this application has added tables for the expert system and HPI generation module of the system. Inserting, revising, and deleting impressions or sickness, and symptoms can be done to the database for future proofing and will be automatically included in the expert system without modifying the system's code. In addition, the results of the expert system and the generated HPI are inserted to the database for future reference. An admin account provides an interface for adding new impressions and symptoms to the system's database.

7.2 Recommendations

The Android application is completely working that can diagnose an impression of what sickness that patient has and generate a one paragraph history of present illness for a quick summary of the patient's case. However, there are still further works to do to improve the system.

Here are some recommendations for the system:

Questions can be answered in many different responses. The expert system module has questions only answerable by yes or no which is not closely similar to how medical professionals diagnose patients in real world scenario. Questions with many different responses will reduce the questions being asked by combining questions that are relevant to each other.

The use of standard Natural Language Generation techniques in generating the History of Present Illness. The system uses a template-based generation of the History of Present Illness which relies on phrases stored in the database and sentences concatenated with each other. The use of standard Natural Language Generation techniques will improve the generation of HPI by adding the correct tenses and verbs to the sentences, adding the right punctuations, adding modifiers and complements, adding multiple subjects, objects, complements, and many more.

A flexible user interface layout that is compatible with all mobile device sizes. The system is designed only for a 10-inch sized tablet. If the system is to be installed in a smaller or larger device, the user interface layout would be distorted. Designing the system for all mobile device sizes would let users access the system on their mobile phones or bigger tablets.

An interface for adding, revising, and deleting impressions, family symptoms, and specific symptoms. Adding, revising, and deleting of impressions, family symptoms, and specific symptoms are all done manually. An additional account for an admin will be created an it will have a different interface and its functions is to add, revise, and delete database information.

Upload and download the medical data of the patient in the cloud database of the GetBetter system. Medical data of the patients are only stored on the local database of the device. It would be better if this data will be uploaded to the GetBetter cloud database. Moreover, medical data should also be downloaded from the GetBetter cloud database and stored in the local database of the device.

Appendix A: Resource Persons

Azcarraga, Arnulfo P.
Full Professor/Adviser
Software Technology
De La Salle University
arnie.azcarraga@delasalle.ph

Gendrano, Ma. Christine A.
Assistant Professor
Software Technology
De La Salle University
ma.christine.gendrano@dlsu.edu.ph

Appendix B: Personal Vitae

Mr. Jan Michael Isaac Dayupay
12 Capetown Street BF International Las Pinas City
5191635 / 09989831052
mike_dayupay@yahoo.com

Appendix C: List of Diseases

Medical Term	Scientific Term	Local Term
Measles	Rubeola	Tigdas
Typhoid Fever	Salmonella typhi	Tipus
Dengue Fever		Dengue
Tonsillitis		
Malaria	Plasmodium	
Tuberculosis	Mycobacterium tuberculosis	TB
Urinary Tract Infection		UTI
Laryngitis		Laryngitis
Pneumonia	Streptococcus pneumoniae	Pulmonya
Mumps		Baiki
Salmonella	Salmonellosis	Salmonella
Leptospirosis		
Hepatitis A		Hepatitis A
German Measles		
Gastroenteritis		
Chicugunya		
Ameobiasis		
Conjuctivitis		Sore Eyes
Middle Ear Infection	Otitis Media	
Acute Schistosomiasis		

Appendix D: List of Symptoms

High Fever
Cough with pleghm
Tummy Pain
Diarrhea
Pain in the Ears
Red Spot Rush
Red Eyes
Sore Throat
Headache
Vomiting
Chest Pain
Fatigue
Weight Loss
Pain when Urinating
Lower Abdomen Pain
Dry Cough
Coughing Up Blood

Bibliography

Agarwal, M., & Goel, S. (2014). Expert System and its Requirement Engineering Process. *Recent Advances and Innovations in Engineering (ICRAIE), 2014* (pp. 1-4). Jaipur: IEEE.

Lewis, F. D., & Griffin, N. L. (1989). A Rule-Based Inference Engine which is Optimal and VLSI Implementable. *Tools for Artificial Intelligence, 1989. Architectures, Languages and Algorithms, IEEE International Workshop* (pp. 246-251). Fairfax: IEEE.

Reiter, E., & Dale, R. (2000). *Building Natural Language Generation Systems.* Cambridge: Cambridge University Press.

Deemter, K. v., Krahmer, E., & Theune, M. (2005). Real versus Template-Based Natural Language Generation: A False Opposition? *Computational Linguistics*, 15-24.

Gatt, A., & Reiter, E. (2009). SimpleNLG: A Realisation Engine for Practical Applications. *Computational Linguistics*, 90-93.

Narayan, K., Isbell, C., & Roberts, D. (2011). DEXTOR: Reduced Effort Authoring for Template-Based Natural Langage Generation. *Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (pp. 170-175). Association of the Advancement of Artificial Intelligence.

Channarukul, S. (1999). YAG: A Template-Based Natural Language Generator for Real-Time Systems.

Theune, M., Klabbers, E., Odijk, J., de Pijper, J. R., & Krahmer, E. (2001). From Data to Speech: A General Approach. *Natural Language Engineering*, 47-86.

J, A. (2003). Natural Language Processing. *Encyclopedia of Computer Science*, pp. 1218-1222.

Arguello, M., Des, J., Prieto, M. J., Perez, R., & Lekkas, S. (2011). An Ontology-Based Approach to Natural Language Generation from Coded Data in Electronic Health Records. *EMS*, 366-371.

Azcarraga, A. (2014). *Project: Cloud-Based Database Application for a Tele-Medicine System that is Useful for Rural Communities.*

Bose, R. (n.d.). *Natural Language Processing: Current State and Future Directions.*

Nusslin, F. (2006). Current Status of Medical Technology. 25-31.

Biermann, A. (1992). A Voice-and-Touch-Driven Natural Language Editor and its Performance. *International Journal of Man-Machine Studies Vol.37, No. 1.*

Friedman, C., & Hripcsak, G. (1999). *Natural Language Processing and Its Future in Medicine.* Retrieved November 10, 2014, from http://www.columbia.edu/itc/hs/medinfo/g6080/misc/rticles/friedman.pdf

Jaya, A., & Uma, G. V. (2008). A Novel Approach for Construction of Sentences for Automatic Story Generation Using Ontology. *Computing, Communication and Networking*, 1-4.

Todd, A. (2014, October 23). *What is Android and What is an Android Phone?* Retrieved November 2015, from recombu: https://recombu.com/mobile/article/what-is-android-and-what-is-an-android-phone_M12615.html

Abad, D., Bautista, A., Teodosio, J., & Valdez, E. (2013). *Medical Consultation Annotation and Summary (MedCAS) Generator.* Manila.

Shortliffe, E., Scott, C., Bischoff, M., Campbell, B., Van Melle, W., & Jacobs, C. (1979). ONCONCIN: An Expert System for Oncology Protocol Management.

Daella, P., Dimayacyac, K., Liao, M., Lim-Cheng, N., & Velarde, A. (2011). Question Answering System to Answer Procedural and Causal Questions in Medicine.