**CHAPTER THREE**

**METHODOLOGY**

In this e-doctor expert system software development process, the design phase involved decomposing the system into modules and defining the relationship among the constituent modules. Top down design approach was employed in doing this. This involved dividing the system into subsystems/modules and each subsystem being further divided into even smaller subs. This process of division is repeated until each module is sufficiently small enough to be conveniently coded (implemented) as an independent entity that performs a clearly defined operation.

**3.1 System Evaluation**

The analysis and comparison of existing expert system software resulted from the review of related works. The critical analysis led to the adoption of the scheme of requirement specifications that highlighted the nature of the proposed model and system. The e-doctor expert system developed in this work focused on automation of the usual day-to-day practices that medical doctors carry out on patients when they are visited in hospitals for diagnosis of non-communicable diseases.

The Admin (the hospital records unit) manages the registration side of the software. When patients visit hospitals for the first time, they are given cards and card numbers (in the conventional manual way of operation). In this case, the hospital record logs on as admin to the system, and registers each new patient electronically as well as generates a unique patient ID for them. This ID is what the patients would use to access healthcare services such as diagnosis, prescription and other medical instructions provided by the e-doctor mobile app. This ID could either be sent to Patients phone, e-mail or printed out on a Card with other bio-data details like Patients name and Sex plus any other detail. This card would help the patient access all their records within the hospital via any unit, using the ID as the primary key. The database would also keep a log of registered users, managing all user names and passwords being used to log on to the system from time to time (time-stamps) for security reasons.

Once the user logs in, the system starts with the welcome page. The first few pages allow the user to answer some basic questions and ticks some symptoms they feel in their body; after that, he is then shown the results. Alternatively, the user can access the data collection system from here; if he chooses to do this, he will answer questions pertaining to past results collected, and see his temperature for example as it changes over time, he is then shown the results (diagnosis). Using any of the channels will both lead to finding out what the ailment is, what medication is recommended, where he can get a dosage recommendation which, given the drug chosen, returns and optimal dose.

**3.2 Data Collection and Databases**

**3.2.1 Data Collection**

For most non communicable diseases, there is very little data, either parasitological or immunological. The problem is not entirely that the data does not exist; rather, it is that the data is not being recorded in a fashion which leaves it available for further use. It is recorded on paper, or stored on computers which cannot access the Internet. Even if the computer storing the data has the ability to access the Internet, currently nobody receives this data. Asystem with a centralized database would therefore aid greatly in data collecting abilities.

For the data used in the development of the e-doctor expert system, data (major symptoms, diagnosis and prescriptions) were sourced from experts in the field of medicine, they had to help in the recording of fresh data from patients that were diagnosed. Thegoal of this system is to provide a simple, easy-to-use, integrated interface for data (symptoms) collection from users. It stores the data from the initial consultations; additionally, the system can input data on the patient's progress, such as temperature readings. The system can then be used to make all these data available for medical research purposes.

**3.2.2 Database**

The data is simply stored on the server. The system stores records of nearly all of the information it is given, and all conclusions that it derives on a permanent database on the server. Although it does not store model parameters, as the model with which it will work varies and so will the number and meaning of the parameters. Therefore, it was deemed not meaningful to store this data. The rest is stored in order to preserve as much data as possible for further research; in a database system dedicated to data collection, preservation of all data that can possibly be useful is an ideal feature. The model needs to have access to the parameters for it, as determined bythe rest of the system, at the time database is run. Rather than placing the values in a file with Java, the language in which this program was written and then having database open and read that file, the values are simply inserted into the database file itself. This saves a lot of work, and is very little additional work for the Java module, as text processing abilities are a strength of Java.

**3.3 Implementation Details**

**3.3.1 Prescription Module**

The prescription module goes through the guidelines step bystep and prescribes a drug. The actual algorithm which it uses will be attached as appendix**.** All these guidelines were derived by carrying out interviews and meetings with experts and the use of online sources*.* The drugs chosen were selected on the advice of a malariologist as the most likely to be available and representative of the different drug categories. The algorithm was written using a point-based system; each drug is assigned a certain number of points based on various factors, and the drug with the most points is the drug that is chosen. This was selected as an extremely simple way to make what could be up to a four-way comparison. Additionally, it is a good way to reflect exactly how important each factor is; if there is one reason to choose a drug, and another against choosing it, for instance, this allows easy resolution of these conflicts.

**3.3.2 Repeat Patients (On-App ‘Cache’ Memory to Remember Logged in Users)**

When the system is told that a patient has been here before, it goes through the permanent database and finds all patients bythat name. Ifno patient bythat name is in the system, it is then forced to treat this person as a new patient. If,however, this patient does exist in the database, the program then returns to the user and confirms this with them. It does this bypresenting a list of everyone with the same name as the patient in question.

**3.4 Conventional Drug Treatment**

The system has a rule-based component. This is used when initially asking the user questions; the program needs to be able to know which questions to ask (i.e. have rules such as "if the patient is male, do not bother to ask if they are pregnant or lactating"). Once the information has been gathered from the user, it will be fed to both the mathematical model described above and a point-based prescription system.

**3.5 System Design**

The system architecture is laid as depicted by the Class diagram and database schema diagrams in figure 3.1 and Figure 3.2 respectively.

**3.5.1 Database Design**

The database is a very important part in developing a mobile app. The database used in this system was developed using SQLite database, this stores the ailments’ symptoms, patients’ diagnosis, prescription, patients’ details etc. SQLite is an open source database management system. It stores data in its own format based on access jet database engine. Database design includes the creation of tables used in the system database. These tables are used to store the information about each entity as shown in figure 3.3. The data structures of the entities on the database are as shown in Table 3.1. These following entities can be identified specifically for the system being developed.

There are two types of users on this web app; the admin user with some exclusive administrative privileges and the patient. The backend is where all users’ data (and every other kind of data of course) are saved for processing and layer retrieval. At the back-end, SQLite database was employed to implement the database management. SQLite work bench was used to create the database tables; most of the tables have a one-to-one relationship.

Table 3.1 Some Database Table Elements and data-types

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data-type** | **Description** |
| Pat\_id | int(11) | Patient identification |
| Pat\_firstname | varchar(18) | Patient first name |
| Pat\_lastname | varchar(20) | Patient last name |
| Pat\_password | varchar(13) | Patient login password |
| Pat\_gender | varchar(7) | Patient gender |
| Pat\_phoneno | int(11) | Patient phone number |
| Pat\_emailadd | varchar(30) | Patient email address |
| ailment\_name | varchar(3) | Ailment name |
| Ailment\_id | varchar(3) | Ailment identification |
| Pat\_d\_o\_b | Date | Patient date of birth |
| sym\_id | int(11) | symptom identification |
| sym\_name | varchar(18) | symptom name |
| drug\_id | varchar(20) | drug identitfication |
| user\_password | varchar(13) | Patient login password |
| diag \_id | varchar(7) | Diagnosis\_id |

There are five main tables created, the first one is the table for ‘ailments’, the details of each ailment’s symptoms being created by the admin is saved on the table. Every ailment is given an id automatically once it is being added by the admin, ailment details include, ailment\_name, ailment\_symptoms, ailment\_id, drugs and dosage. The second table is the ‘symptoms’ table, each symptom is given an auto id, and the associated ailment\_name is also on the table. The next table is the ‘prescription’ table; the table stores the details of every drug added by the admin user. Every drug created is given an auto id, every drug is meant to belong (cure) to an ailment, so the ailment\_id is also included with the drug on the drug table.

Another important table is the (result) diagnosis table; a diagnosis consists of a patient’s confirmed ailment and required prescription as well as the recommended dosage after answering a series of questions taken at a particular point in time. The diagnosis details cover the user\_id, symptoms, units, drugs. The user’s table holds all bio-data and medical information of the registered patients; there two categories of users; the admin user, who can create the other entities of the mobile app. The tables designed and created can be viewed via the local host of the backend.

Figure 3.1 Class Diagram showing the design for the e-doctor expert system

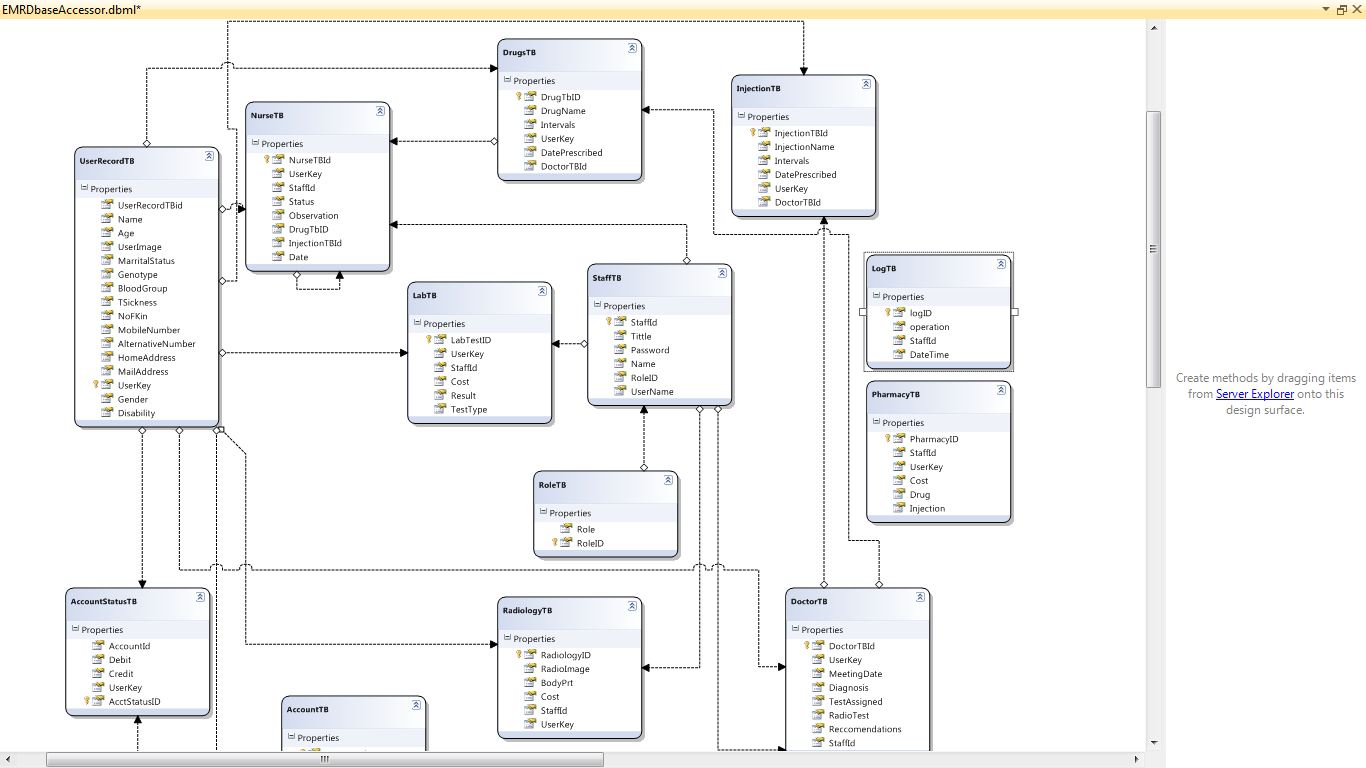


Figure 3.2 Diagram showing the database schema for the e-doctor expert system

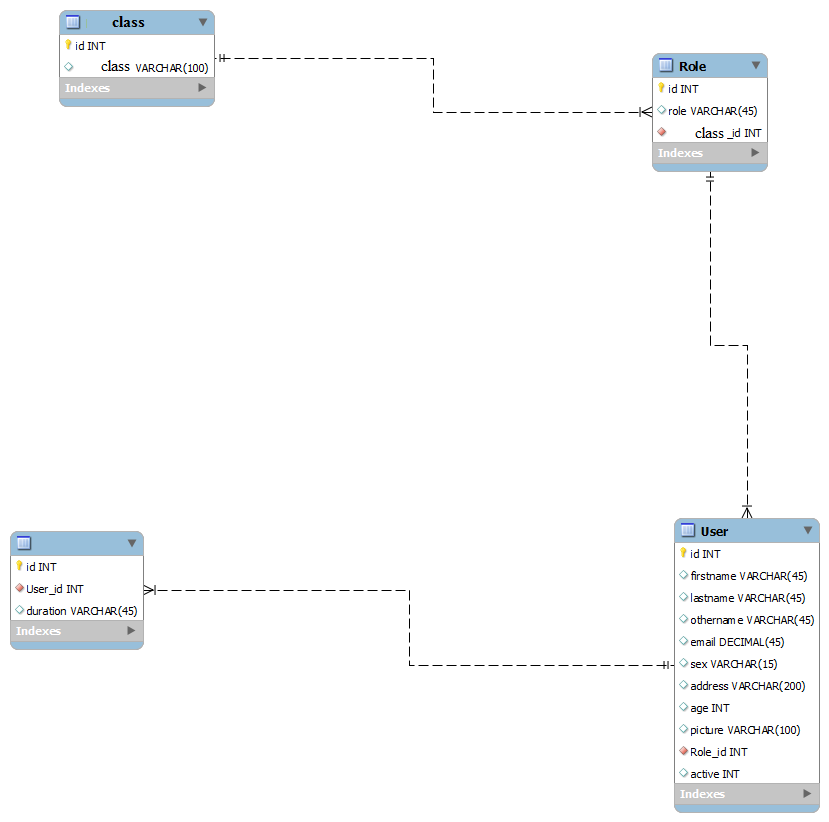


Figure 3.3 Entity Relational Diagram of the system

**3.5.2 Activity Diagram**

Activity diagrams are graphical representations of workflows depicting stepwise activities and actions with support for choice, iteration and concurrency. In the unified modeling language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

The activity diagram shows the visual representation of the process flow. In this e-doctor expert system, the diagram shows how a user will navigate from one point to another on the mobile app in order to getting the task of getting the right prescription for an accurate diagnosis of an ailment. The mobile app was developed with skilfully linked pages that followed a very orderly sequence; the landing page is the home page (index page), on the home page, the user aim is to login (after registering on the platform during the first use of the system) answer some questions which will gather the symptoms of the ailment the patient might be suffering from and get the correct diagnosis and right prescription of drugs or activities to make sure the patient gets healed of the ailment. The admin creates the ailments, symptoms, can also create users and every other entities of the mobile application software. For a user to be successfully logged in, the provided credentials must match the one stored in the database at the point of registration at the first use of the expert system. If the credentials are correctly supplied, the user would be taken to the home page, else he/she would be taken back to the ‘login’ page to supply the correct and matching username and password. This and more are more elaborated discussed in chapter four of this final year project report. Figure 3.4 on the next page clearly shows the data and process flow of activities in the e-doctor expert system.

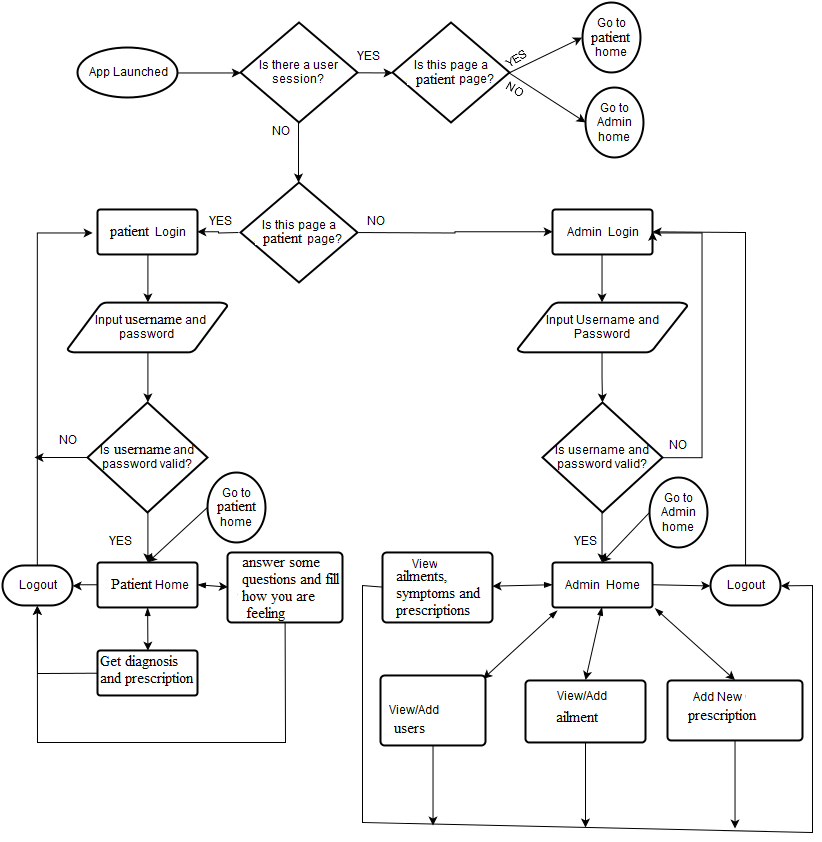


Figure 3.4 Program activity diagram

**3.5.3 Data Flow Diagram (DFD) for the System**

A data flow diagram (DFD) uses a very limited number of primitive symbols to represent the functions performed by a system and the data flow among the functions. Starting with a set of high-level functions that a system performs, a DFD model hierarchy represents various sub-functions. The data flow diagram depicted in figure 3.5 below shows the relationship between the entities in the e-doctor expert system. The entity ‘patient’ can access the diagnosis page after he or she gains access to the system. The entities ‘Admin’ can view/add ailments, their symptoms and prescriptions to be accessed by patients into the database. The admin is also saddled with the responsibility of adding new users (patients) when necessary and setting the default password for the users of the system. The entity ‘system’ is responsible for authenticating the users of the system and also provides the timeout functionality for the users’ session, the system logs off a user upon expiration of the login session.

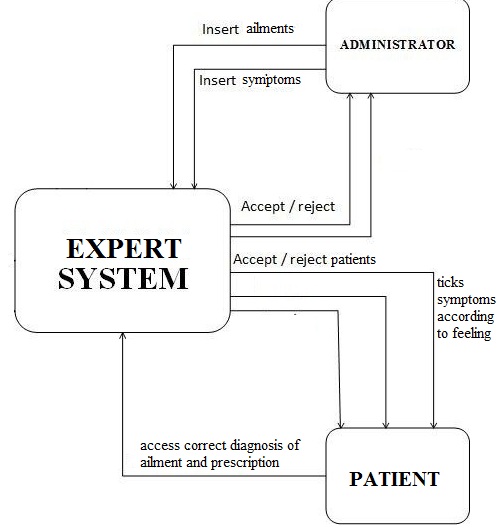


Figure 3.9 Data flow diagram

**3.6 Procedure Phases and Time Frame**

The implementation of the project was divided into the following phases:Requirement phase, Analysis Phase, Design phase, Implementation and testing phase

* Requirement phase: here, a complete description of the behavior of the system was done. The interaction users would have with the software was clearly mapped out for clear development (coding).
* Analysis Phase: Formal enquiry was carried out in order to identify a better course of action to develop the system, thoughts were shared, and online resources were perused for clear information.
* Design phase: The Graphical User Interface for the mobile application was developed. This was done using the Java programming language.
* Implementation Phase: This is the part of the process where the software engineering of the expert system was actually done. The coding from scratch using Java on android studio and SQLite. Scripts for different modules were written and fully tested okay. This phase is in progress.

The project will be completed in a six month timeline (first and second semesters). It is divided into two segments completed in each semester. The first segment is the gathering of relevant information and other resources for the successful completion of the project, this entailed the preparation of the chapter one (introduction) and two (review of literatures on past related works) of the project report. The second segment involved the actual implementation and discussion of the results attained.