

Microlink Information Technology College

Department of Computer Science

E Health System

A PROJECT SUBMITTED TO MICROLINK INFORMATION TECHNOLOGY COLLEGE INPARTIAL FULFILLMENT FOR THE BACHELOR DEGREE IN COMPUTER SCIENCE

**Prepared By**

1. **Biruk Endalew**
2. **Dawit Tadesse**
3. **Eyosiyas Nigussie**
4. **Kirubel Maregn**
5. **Yohannes Melese**

**Advisor: Tensaye Desalegn**

**July 2022, Addis Ababa**



Microlink Information Technology College

Department of Computer Science

E Health System

**Prepared By**

1. **Biruk Endalew**
2. **Dawit Tadesse**
3. **Eyosiyas Nigussie**
4. **Kirubel Maregn**
5. **Yoannes Melese**

**Advisor: Tensaye Desalegn**

**APPROVED BY**

**EXAMINING BOARD:**

1. **Tensaye Desalegn, Advisor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
2. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
3. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**July 2022, Addis Ababa**

ACKNOWLEDGMENTS

First and foremost, we would want to express our thankfulness to the Almighty God for providing us with the courage, knowledge, and skills necessary to begin and complete this project.

We would like to express our gratitude to all of the Microlink Information College's teaching and non-teaching staff members as well as to our devoted families for their effort, support, and amazing teamwork in making this project a reality.

It brings us great pleasure to express our sincere appreciation to Mr. Tensaye Desalegn, the project's lead adviser and a department of computer science instructor at Microlink Information College, for his unwavering support, recommendations, and encouragement as we built and developed our project.

TABLE OF CONTENTS

**Contents**

[ACKNOWLEDGMENTS i](#_Toc108115979)

[TABLE OF CONTENTS ii](#_Toc108115980)

[LIST OF FIGURES iv](#_Toc108115981)

[LIST OF TABLES v](#_Toc108115982)

[ABBREVIATIONS AND ACRONYMS vi](#_Toc108115983)

[ABSTRACT vii](#_Toc108115984)

[CHAPTER 1 1](#_Toc108115985)

[1.SOFTWARE PROJECT PROPOSAL 1](#_Toc108115986)

[**1.1Background** 1](#_Toc108115987)

[**1.1 The Existing System** 2](#_Toc108115988)

[**1.1.1 Problems of the Existing System** 2](#_Toc108115989)

[**1.2 The Proposed System** 3](#_Toc108115990)

[**1.2.1 General Objective** 3](#_Toc108115991)

[**1.2.2 Specific Objective** 3](#_Toc108115992)

[**1.2.3 Significance of the Project** 3](#_Toc108115993)

[**1.2.4 Scope of the Project** 4](#_Toc108115994)

[**1.3 Methodologies and Techniques** 4](#_Toc108115995)

[**1.3.1 Data Collection Methods and Techniques** 5](#_Toc108115996)

[**1.4 System Analysis and Design and Development Tools** 5](#_Toc108115997)

[CHAPTER 2 6](#_Toc108115998)

[2. Software Requirements Specification 6](#_Toc108115999)

[**2.1 Introduction** 6](#_Toc108116000)

[**2.1.1 Purpose** 6](#_Toc108116001)

[**2.1.2 Scope** 7](#_Toc108116002)

[**2.1.2 Overview** 7](#_Toc108116003)

[**2.2 Current System** 7](#_Toc108116004)

[**2.3 Proposed System** 8](#_Toc108116005)

[**2.3.1 Overview** 8](#_Toc108116006)

[**2.3.2 Functional Requirements** 8](#_Toc108116007)

[**2.3.3** **Nonfunctional requirements** 9](#_Toc108116008)

[**2.3.4** **Constraints (“Pseudo Requirements”)** 13](#_Toc108116009)

[**2.4 System models** 14](#_Toc108116010)

[**2.4.1 Scenarios** 14](#_Toc108116011)

[**2.4.2** **Use Case Model** 17](#_Toc108116012)

[**2.4.3** **Object Model** 23](#_Toc108116013)

[**2.4.4** **Data Dictionary** 23](#_Toc108116014)

[**2.4.5** **Class Diagram** 25](#_Toc108116015)

[**2.4.6** **Dynamic Models** 26](#_Toc108116016)

[**2.4.6.1 State Diagram** 26](#_Toc108116017)

[CHAPTER 3 32](#_Toc108116018)

[3. SYSTEM DESIGN DOCUMENT 32](#_Toc108116019)

[**3.1 Introduction** 32](#_Toc108116020)

[3.1.1 Purpose 33](#_Toc108116021)

[3.1.2 Scope 33](#_Toc108116022)

[3.2 Goals and Trade-Offs 33](#_Toc108116023)

[**3.2.1 Goals** 33](#_Toc108116024)

[3.2.2 Trade-offs 34](#_Toc108116025)

[3.3 System Decomposition 34](#_Toc108116026)

[3.3.1 Layers and Partitions 35](#_Toc108116027)

[3.3.2 System Topology 35](#_Toc108116028)

[3.4 Concurrency Identification 36](#_Toc108116029)

[3.5 Hardware/Software Allocation 36](#_Toc108116030)

[3.5.1 System Performance 37](#_Toc108116031)

[**3.5.1.1 General system performance** 37](#_Toc108116032)

[**3.5.1.2 Input/Output Performance** 37](#_Toc108116033)

[**3.5.1.3 Processor allocation** 37](#_Toc108116034)

[**3.5.1.3 Memory allocation** 38](#_Toc108116035)

[3.5.2 Connectivity 38](#_Toc108116036)

[3.5.3 Network architecture 38](#_Toc108116037)

[3.6 Data Management 38](#_Toc108116038)

[3.7 Global Resource Handling 38](#_Toc108116039)

[3.8 Software Control Implementation 39](#_Toc108116040)

[3.8.1 External control flow (between subsystems) 39](#_Toc108116041)

[3.8.2 Concurrent control 39](#_Toc108116042)

[3.8.3 Internal control (within a single process) 39](#_Toc108116043)

[3.8.4 User Interface 39](#_Toc108116044)

[3.9 Boundary Conditions 40](#_Toc108116045)

[3.9.1 Initialization 40](#_Toc108116046)

[3.9.2 Termination 40](#_Toc108116047)

[3.9.3 Failure 41](#_Toc108116048)

[3.10 Design Rationale 41](#_Toc108116049)

[CHAPTER 4 41](#_Toc108116050)

[4 IMPLEMENTATION 41](#_Toc108116051)

[4.1 Introduction/Overview 41](#_Toc108116052)

[4.2 Tools and Technology Utilized for Implementation 42](#_Toc108116053)

[4.3 System Implementation 42](#_Toc108116054)

[**4.3.1 System Interfaces** 42](#_Toc108116055)

[**4.3.3 Source Code** 42](#_Toc108116056)

[CHAPTER 5 48](#_Toc108116057)

[5 CONCLUSION AND FUTURE WORK 48](#_Toc108116058)

[5.1 Conclusion 48](#_Toc108116059)

[5.2 Future work 48](#_Toc108116060)

[Glossary](#_Toc108116061) [48](#_Toc108116062)

LIST OF FIGURES

Figure 1- Use Case Diagram

Figure 2: Class Diagram

Figure 3: State Diagram

Figure 4: Sequence diagram for Login

Figure 5: Sequence diagram for Appointment

Figure 6: Sequence diagram for Patient Follow Up

Figure 7: Sequence diagram for admin

Figure 8: Activity diagram for patient

Figure 9: Activity diagram for doctor

Figure 10: Activity diagram for admin

Figure 11: Component Diagram

Figure 12: Deployment Diagram

Figure 13: Connectivity Diagram

Figure 14: Home Page

Figure 15: Patient Home Page

Figure 16: Doctor Home Page

LIST OF TABLES

Table 1: Project Hardware Consideration

Table 2: Login/Logout Scenario

Table 3: Make Appointment

Table 4: Follow up with doctor

Table 5: Doctor Appointment Response

Table 6: Follow up Patient

Table 7: View Data

Table 8: Add Doctor

Table 9: Delete Doctor

Table 10: Use case description for user registration

Table 11: Use case Description for Login

Table 12: Use case Set Appointment

Table 13: Use case Follow up Doctor

Table 14: Use case View Data

Table 15: Use case Response Appointment

Table 16: Use case Follow Up Patient

Table 17: Use case Add Doctor

Table 18: Use case Delete Doctor

Table 19: Patient Register

Table 20: Doctor

Table 21: Appointment

Table 22: Follow up

Table 23: Admin

ABBREVIATIONS AND ACRONYMS

|  |  |
| --- | --- |
| **IT** | Information Technology |
| **CSS** | Cascading Style Sheet is used to format the layout of web page. |
| **DB** | Data Base Is an organized collection of data |
| **JS** | **Javascript** Is a programming language that is used to create application for android |
| **MySQL** | Is a relational database system |
| **PHP** | Is a scripting language and used to develop dynamic web application together with MySQL, HTML, CSS and Java Script. |
| **Xampp** | Server |
| **FR** | Functional Requirements |
| **NFR** | Non- Functional Requirements |
| **RAD** | Requirement Analysis Document |
| **GUI** | Graphical User Interface |

ABSTRACT

The Project “E Health System” provides online appointment and follow up with doctor. A user has own personal email and password to access the system. The system helps to facilitate online appointment and follow up for patient and doctor.

We discuss organization, outline the problem statement, and then present the newly suggested system in the software project proposal. We discuss how the system is set up and built for a while in this project document. Next in the System Requirement Specification, we explain about the requirements elicitation and analysis activities (UML Scenarios, Use Case, Activity diagrams).

We discuss the project's design objectives, subsystem decomposition (using UML Class diagrams), hardware/software mapping (using UML Deployment Diagrams), data management, access control, control flow methods, and boundary conditions in the System Design section.

The tools and technologies used to construct the project are described and defined in the implementation, along with the design and implementation of the system.

The document outlines the system's conclusions and future work with regard to system maintenance and improvement in the section titled "Conclusion and Future Works." Lastly, References and Glossary will hire and will contain additional information.

CHAPTER 1

1.SOFTWARE PROJECT PROPOSAL

**1.1****Background**

E-Health is a wide word that refers to the technology and infrastructure that are used to capture, analyze, and communicate patient health data. The goal of health is to offer better treatment for patients and to contribute to health fairness. It enhances healthcare delivery quality, boosts patient safety, reduces medical mistakes, and strengthens interactions between patients and healthcare professionals.

The "online Doctor Appointment system" was created to overcome the challenges that existed in the conventional manual approach. This program is supported in order to remove and, in certain situations, decrease the difficulties encountered by the current system. Furthermore, this system is built to meet the specific needs of the firm in order to carry out activities in a smooth and effective manner.

If a person becomes ill and want to see a doctor for a checkup, he or she must go to the hospital and wait until the doctor becomes available. While waiting for an appointment, the patient simultaneously waits in line. If the doctor cancels the appointment due to an emergency, the patient will not be aware of the cancellation until and until he or she attends the hospital. Because mobile communication technology is continually evolving, web apps may be used to alleviate such challenges and inconveniences for patients.

Patient Doctor Follow Up is application that allows patient to communicate with their doctors in an efficient and easy manner. The application gathers all of the patient’s information, including the kind and stage of the condition. The patient ask the doctor the specific question at any moment and application sends all of the patient’s questions to the doctor and the doctor can send response to all patient’s questions. The patient can view all the details of the response given by the doctor

**1.1 The Existing System**

The existing system of Addis Hiwot General Hospital

**Manual Appointment System**

Medical appointments have typically been made via telephone or in-person with schedulers. These methods are based on real-time verbal communication and allow for maximum flexibility in difficult situations. Because these old methods rely on schedulers, the ability to get an appointment on time is limited not only by the availability of appointment times but also by the availability of schedulers and phone lines. The ability to book at the proper time with the right health service providers has an impact on patient satisfaction with appointment scheduling.

**The Traditional way of Patient Doctor Follow Up**

Patients are unable to contact directly with doctors under the current system. It takes a long time to respond to the patient. There is very little dialogue between the patient and the doctor. Patients must physically travel to the doctor, which requires a significant amount of time and effort.

**1.1.1 Problems of the Existing System**

- Data entry inconsistency, the potential for errors, mis keying information

- Extensive continuous training costs for employees.

- The system is predicated on excellent people.

- Reduction in information sharing and customer service.

- Reports are time-consuming and pensive to produce.

- difficult to change the appointment date

**1.2 The Proposed System**

We have proposed a system that will solve the above problems mentioned.

**Online appointment is the proposed system**

The proposed system will allow people to make appointments online to Doctor or Physician at any time and from any location with an internet connection.

**Patient Doctor Follow up**

The suggested approach ensures effective communication between the patient and the doctor. The patient can save a significant amount of time and physical effort in obtaining a remedy from the doctor.

With the assistance of a doctor, the program collects all of the patient's data and determines the best treatment for his disease. The suggested approach allows patients and physicians to receive responses quickly and without wasting time or effort.

**1.2.1 General Objective**

This project has the general objectives of developing a web-based E-health System for Addis Hiwot General Hospital. The project is primarily concerned with the patient. As a result, we aimed to help patients make online appointments with doctors and patient doctor follow up.

**1.2.2 Specific Objective**

* To develop an online Doctor Appointment system
* To eliminate time-consuming
* To make patient change schedule (date) any time
* To make patient ask question doctor or physician
* To make doctor follow up patient

**1.2.3 Significance of the Project**

* it allows to book whenever you like
* it saves time for patients and doctors
* reduces burden of some administrative tasks for staff.
* follow-up can help reduce hospital re admissions
* Daily, weekly and monthly patients scheduled are easily viewed.
* it helps patients after their visit to talk about their status or ask question to doctors

**1.2.4 Scope of the Project**

Build for General Hospital. The proposed system replaces traditional booking chaos with online facility. System allow patient the power to book their own appointment with respective doctor online that benefits organizations to manage a time. Patient can give his/her status to doctor after they meet and ask different question doctor, he/she choose.

The Limitation of this project when a patient attempts to schedule an appointment, the system fails in areas where there is no Network. This system does not handle day-to-day hospital management concerns such as payment and billing, card creation, personnel management, and so on.

**1.3 Methodologies and Techniques**

* Waterfall Model

1. Requirement Analysis
2. System Design
3. Implementation
4. Testing
5. Deployment
6. Maintenance

Waterfall model is most appropriate

* Requirement are very well documented, clear and fixed
* Product definition is stable
* Simple and easy to understand and use
* Easy to manage due to the rigidity of the model

1. **Requirement Analysis and Gathering**

During this phase, all potential system needs are identified and recorded in a requirement specification document.

1. **System design**:

This phase studies the need specifications from the previous phase and prepares the system design. System design aids in the specification of hardware and system requirements, as well as the definition of overall system architecture.

1. **Implementation:**

The system is first built-in tiny programs called units, with input from system design, and then combined in the following step. Unit Testing is the process through which each unit is designed and tested for functioning.

1. **Testing:**

After each unit has been tested, all of the units built during the implementation phase are merged into a system. Following integration, the complete system is tested for flaws and failures.

1. **Deployment of system**:

Following the completion of functional and nonfunctional testing, the product is deployed in the client environment or launched to the market.

1. **Maintenance**:

There are a few difficulties that arise in the client environment. Patches have been provided to address these problems. In order to improve the product, newer versions are published. Maintenance is performed in order to implement these modifications in the client environment.

**1.3.1 Data Collection Methods and Techniques**

Because the project will focus on healthcare, hospitals, clinics, and other health care institutions will be the primary locations where we will gather the majority of the project's data. The process of requirement analysis and elicitation will be carried out through observation and interviews.

**1.4 System Analysis and Design and Development Tools**

We plan to use different tools for the system design and analysis of the project. It will be used for:

* LibreOffice - to prepare requirement analysis document.
* Draw.io - as a UML tool for system analysis and design phase since this
* Tool will be fully object-oriented feature that helps for the preparation of system design document (SDD).
* Gantt - to produce the Gantt chart for the schedule of the project.
* Uxpin – to develop the designed user interface.

**Development Tools**

* For the front end, we plan to use HTML, CSS, BOOTSTRAP, JAVASCRIPT**.**
* On the back-end side, we plan to use PHP**.**
* On the database side we will use relational database which is MySQL.
* We will use Flutter to develop applications for android. is an open-source mobile application development SDK.

**CHAPTER 2**

**2. Software Requirements Specification**

**2.1 Introduction**

This section gives an overview of the contents of the full document. the results of the

requirements elicitation effort reported in this Requirement Analysis Document (RAD). RAD uses a variety of textual, symbolic, and diagrammatic representations based on the subject's wishes. The primary functional activities, states, and item interactions within the design are shown using the system model. It acts as a contractual basis between the client and the developers, and it seeks to explain the system in terms of functional and non-functional requirements. This document quickly summarizes the system's functional and non-functional needs, as well as its security, hardware and software requirements, risk, cost, how the user interacts with the system, the type of training required, error handling, backup strategy, and power stability.

**2.1.1 Purpose**

The results of the requirements elicitation and the analysis activities are documented in the Requirements Analysis Document (RAD) For E Health System. This document completely describes the system in terms of functional and nonfunctional requirements and serves as a contractual basis between the customer and the developer. The RAD must be written in the language of the customer's domain of business/expertise. Under no circumstances should any "computerize" terminology creep into this document.

**2.1.2 Scope**

The proposed software product is the E Health System. In this project we are going to design and build a functional web-based E Health System.

The aim of this project is to create an E Health System that Patient can able to book doctor’s appointment and follow up with doctors (communicate after appointment). Doctors have access to view appointment and set available time for appointment thereby making it more convenient for them and follow up patient. The admin also has access to database to add or delete doctor on system.

The system will be web-based online application that is available on all major browsers. should be user friendly in all usual devise (PC, mobile, tablet).

**2.1.2 Overview**

The proposed system E Health System is organized to clearly define the proposed system using object-oriented software engineering components and techniques that briefly explain the proposed system definition, functional requirements, non-functional requirements, the software interfaces, what the software will do, the constraints under which it must operate, and how the software will react to external stimuli. This paper is designed for both end users and software developers.

**2.2 Current System**

The existing system of Addis Hiwot General Hospital

**Manual Appointment System**

Medical appointments have typically been made via telephone or in-person with schedulers. These methods are based on real-time verbal communication and allow for maximum flexibility in difficult situations. Because these old methods rely on schedulers, the ability to get an appointment on time is limited not only by the availability of appointment times but also by the availability of schedulers and phone lines. The ability to book at the proper time with the right health service providers has an impact on patient satisfaction with appointment scheduling.

**The Pervious way of Patient Doctor Follow Up**

Patients are unable to contact directly with doctors under the current system. It takes a long time to respond to the patient. There is very little dialogue between the patient and the doctor. Patients must physically travel to the doctor, which requires a significant amount of time and effort.

**2.3 Proposed System**

**2.3.1 Overview**

The proposed system is to make web application based online Doctor or Physician appointments at any time and from any location with an internet connection and to ensures effective communication between the patient and the doctor.

In this subsection of the RAD, we will describe the proposed system Functional Requirements, Non-Functional Requirements and Constraints

**2.3.2 Functional Requirements**

In this sub-section we will explain about the functional requirements of the newly proposed system that define, the fundamental actions and description of activities that the system must perform and services a system must provide and accomplish. In other words, it describes the service provided for the users. The functional requirements categorized as the following:

* + - 1. **Patients**
* The patient can access the E Health System.
* The patient inserts his/her email and password at log in form in the patient login page.
* If the patient inserts a valid email and password, he or she will see the main menu that contains the following items:
* Access His/her Profile.
* Personal information: his/her id given by system, name and email.
* See his/her previous appointment or follow up question and answers he/she made with doctor.

**Functional Requirements:**

* Patient can choose doctors.
* Patient can make appointment by choosing comfortable day and time.
* Patient can make contact to doctor to ask question and to get information or to give his/her status to doctor after they meet with doctor after appointment by Follow Up option.
* Patient can change his/her email, password, name or update his/her profile.
  + - 1. **Doctor**
* The doctor can access the E Health System.
* The doctor insert/s his/her email and password at log in form in the doctor login page.
* If the doctor inserts a valid email and password, he or she will access the operation on:
* Access his/her Profile.
* Personal information: his/her id given by system, name and email.

**Functional Requirement**

* Doctor can response to the appointment made by patient if comfortable to him/her he/she will approve or decline.
* Doctor can response to the his/her patient question or will give information to patient.
* Doctor can update his/her profile like his/her profile image, name, email, password and contact.
  + - 1. **Admin**
* The admin can access the E Health System.
* The admin insert/s his/her email and password at login form in the admin login page.
* After the admin insert valid email and password, he will access the operation on:
* Access all appointment made by patient weather approved or decline.
* Access all follow up or exchange of information by patient and doctor.

**Functional Requirement**

* Admin can register new doctor to the system
* Admin can delete doctor from the system if it is necessary
  + 1. **Nonfunctional requirements**

Throughout this article, we will discuss the non-functional objectives of the newly suggested E Health System, as well as a description of features, constraints, and other qualities that constitute an acceptable system.

* + - 1. **User Interfaces**

The system will have three types of users that interact with the system. These users are Patients, Doctors and Admin.

The GUI will be intended to make a good first impression and appear user-friendly, so that people can easily utilize it. Error handling is done using massages which are popped up or alert to the user

**2.3.3.2 Documentation**

In the E Health System there will be four kinds of documentation

1. **Functional Description**: describes the general overview of the system.
2. **Install Manual**: explain how to set up the machine and setup the program.
3. **Introductory Manual**: describe how to get started with the program.
4. **Reference Manual**: describe all the program functions available for the user.
   * + 1. **Hardware considerations**

The system will be executed and run in a computer and a smart phone with the following hardware considerations.

The hardware considerations listed in the below table were considered the system to provide the best of its performance.

**Table 1.0 Project Hardware Consideration**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Hardware Type | Minimum Requirement  Specification | Recommended Requirement  Specification |
| 1 | Desktop Computer | **Processor:** Dual Core, 2.10GHz  **RAM:** 2GB, DDR3  **HDD:** 500 GB, 12,000rpm, WD or Seagate | **Processor:**5 Core, 2.80GHz or above speed  **RAM:** 8GB, DDR3 or above capacity  **HDD:** 1TB, 12,000rpm, WD or Seagate |
| 2 | Server Computer | **Processor:**5 Core, 2.10GHz, 20MB Cache Memory per processor  **RAM:** 8GB, DDR4 SDRAM,1866MHz  **HDD:** 1TB, SATA 6GB/s, 12,000rpm, SCSI | **Processor:** 8 Core, 3.20GHz and above speed with 20MB Cache Memory per processor  **RAM:** 12GB and expandable to 16GB, DDR4 SDRAM, 1866MHz or above capacity  **HDD:** 2TB and expandable, SATA 6GB/s, 12,000rpm, SCSI |
| 3 | Smart phone | Android OS based smart Phone having version API 21 (Lollipop)  2 Gigabytes of RAM  Storage 8GB | Android OS based smart Phone having version API 27 (Oreo)  4 Gigabytes of RAM  Storage 32 Gb |
| 4 | Printer | HP LaserJet 2010 | HP LaserJet 2010 |
| 5 | Switch | 2 CISCO Small Business 200 Series, 10/100/1000 Base-TX | 5 CISCO Small Business 200 Series, 10/100/1000 Base-TX |
| 6 | Network Cable | CAT-5 | CAT-6 |
| 7 | Internet Connection | 500kbps bandwidth | 1mbps bandwidth |

**2.3.3.4 Performance characteristics**

The system should respond to a user’s request for information in less than 0.1 sec. during peak time and 0.01 sec during normal time. The E Health System will allow users to access the system every 24/7. There are no size constraints on the data to be processed by the system.

**2.3.3.5 Error handling and extreme conditions**

* System handles input errors and exceptions by pop-up or alert message to notifying the users.
* Admin should continuously maintain it.
* Power failures handled by Stabilizers.
* Hardware failure switch, Router, network cable, printer, Storage, Database problem, Network problems informing the user about the problem

**2.3.3.6 System interfacing**

* + The system interface has user-friendly features.
  + There will be a GUI that will be used to interact with the web application and a smart phone.
  + Focus on delivering messages. A void busy UI.
  + The website design should clearly explain the purpose of the service.

**2.3.3.7 Quality issues**

The main things for the system to be reliable to run it have backup storage.

The system will trap all possible faults runtime and user invalid input error with its error handling method pops-up a message box or alert to the user what is happened, why is happened and what should be done and take care of about the fault.

The acceptable system downtime per 24-hour period will be the total of 45 minutes

* + The downtime when the system is unavailable, the time that a system fails the system will restart in less than 20 minutes all services must be activated.

**2.3.3.8 System modifications**

* + As the number of users increases from time to time, technology changes and improvements will have an effect on the different hardware’s and software’s of the system, some features might require to be changed. So that the system will face performance problems, and the user also needs a user-friendly interface for the system to used, and then a change has to be made on some of the hardware, forms, user interface and additional features.

**2.3.3.9 Physical environment**

The target equipment operates in a personal computer and server. On the first deployment of the system is in the organization single place and the system accessed from different sides of its user. The server will be located in a safe, appropriate room in which it may not be affected by dusts, unconditional room temperature, any person unless it is required maintenance.

**2.3.3.10 Security Issues**

Access to the system will be protected by user login screen that requires a user name and password.

Unverified users cannot access the system.

The system’s back-end server shall only be accessible to authenticated person that is the system admin.

The system shall automatically log out all customers after a period of inactivity.

The system will be secure and maintain physically.

**2.3.3.11 Resources and Management Issues**

The system is backed by the sensitivity of the number of users that are using the system and information that is stored on the database initially for the first round the database is backed by monthly base later it can be modified.

System Administrator is responsible to:

System installation: managing, troubleshooting, licensing, and updating hardware and software assets.

Backup: implement backup database weekly, with at least daily transaction log backups.

Maintenance: Required to troubleshoot and fix issues that compromise system performance or access. By using this all method to manage the system.

**2.3.3.12 Reliability**

* The software will not be able to connect to the centralized database in the event that the internet network fails or in the event of the server being down due to a hardware or software failure
* Components of the project code shall be tested alongside the implementation phase to ensure that they are functional.
* Final, integrated project Code shall be tested with any testing tool to ensure that greater than or equal to 80% of the integrated code is covered at run-time, and is functioning properly. The remaining 20% will be inspected through manual testing to ensure the highest chance of being quality code.
  + 1. **Constraints (“Pseudo Requirements”)**

The pseudo requirement of the proposed system of E health System is defined and listed below:

* The front-end should be implement by using JavaScript for Web application
* The backend system should be implement using PHP for Web application
* The database should be implemented using MySQL database and also web application must be use MySQL, and PHP dynamic page for server-side scripting language

**2.4 System models**

**2.4.1 Scenarios**

**Table 2 Login/Logout Scenario**

|  |  |
| --- | --- |
| Scenario Name | User Log In/ Logout |
| Actors | Admin, Patients and Doctors  1. The user initiates the application.  2. The system redirects the user to the log in page.  3. The users inter the email and password.  4. The system checks the log in user and redirect to the home page.  5. The user will Logout after use |

**Table 3 Make Appointment**

|  |  |
| --- | --- |
| Scenario Name | Make Appointment |
| Actors  Flow of events | Patients   1. The user initiates the application. 2. The system redirects the user to the log in page. 3. The users inter the email and password. 4. The system checks the log in user and redirect to the home page. 5. The patient chooses doctors 6. Then Make appointment by choosing is comfortable day and time. 7. The System accepts the data 8. The user will Logout after use |

**Table 4 Follow up with doctor**

|  |  |
| --- | --- |
| Scenario Name | Follow up with doctor |
| Actors  Flow of events | * Patient  1. The user initiates the application. 2. The system redirects the user to the log in page. 3. The users inter the email and password. 4. The system checks the log in user and redirect to the home page. 5. Patient choose his/her doctor send questions or his/her status. 6. The System accepts the data. 7. The user will Logout after use |

**Table 5 Doctor Appointment Response**

|  |  |
| --- | --- |
| Scenario Name | Doctor Appointment Response |
| Actors  Flow of events | Doctors   1. The user initiates the application. 2. The system redirects the user to the log in page. 3. The users inter the email and password. 4. The system checks the log in user and redirect to the home page. 5. The doctor will approve or decline the appointment 6. The System accept the data. 7. The user will Logout after use |

**Table 6 Follow up Patient**

|  |  |
| --- | --- |
| Scenario Name | Follow up Patient |
| Actors  Flow of events | Doctors   1. The user initiates the application. 2. The system redirects the user to the log in page. 3. The users inter the email and password. 4. The system checks the log in user and redirect to the home page. 5. The doctor will answer the patient’s question or give information, 6. The System accept the data. 7. The user will Logout after use |

**Table 7 View Data**

|  |  |
| --- | --- |
| Scenario Name | View Data |
| Actors | Patient, Doctor, Admin |
| Flow of events | 1. The user initiates the application. 2. The system redirects the user to the log in page. 3. The users inter the name and password. 4. The system checks the log in user and redirect to the home page. 5. The user can view permissible data. 6. The user will Logout after use. |

**Table 8 Add Doctor**

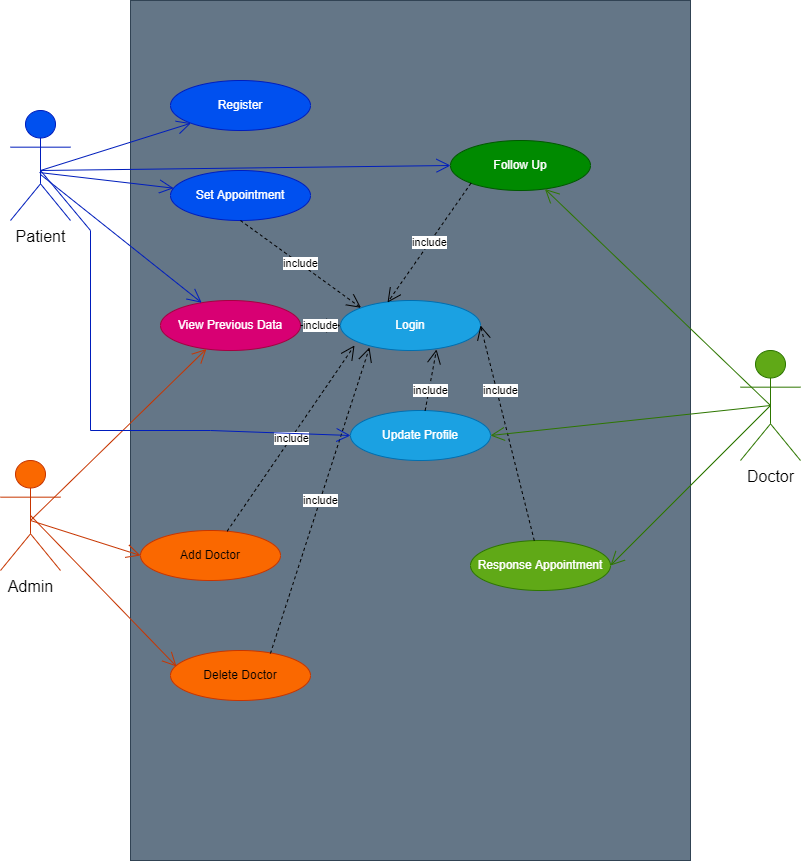
|  |  |
| --- | --- |
| Scenario Name | Add Doctor |
| Actors | Admin |
| Flow of events | 1. The user initiates the application. 2. The system redirects the user to the log in page. 3. The users inter the name and password. 4. The system checks the log in user and redirect to the home page. 5. The user will add new doctor to the system. 6. The System will save data 7. The user will Logout after use. |

**Table 9 Delete Doctor**

|  |  |
| --- | --- |
| Scenario Name | Delete Doctor |
| Actors | Admin |
| Flow of events | 1. The user initiates the application. 2. The system redirects the user to the log in page. 3. The users inter the name and password. 4. The system checks the log in user and redirect to the home page. 5. The user will delete doctor from the system. 6. The System will save data 7. The user will Logout after use. |

* + 1. **Use Case Model**

**Figure 1: use case model**



**Table 10 Use case description for user registration**

|  |  |
| --- | --- |
| **Name** | User Registration |
| Actor | Patient |
| Entry Conditions | The user registration page should be loaded |
| Flow of Events | 1. 1. The user initiates the system. 2. 2. The system redirects to the Registration page. 3. 3. User click on signup button 4. 4. User to registration form page which include Full name,   Email, Emergence Name, Emergence Contact, Age, Contact, Password   1. 5. The user fills the registration form page and submits (Register). 2. 6. The systems prompt the home page. |
| Exit Conditions | User is registered with detail information |

**Table 11 Use case Description for Login**

|  |  |
| --- | --- |
| Name | User Log In |
| Actors | Patient, Doctor, Admin |
| Entry Conditions | The Log in Page should be Loaded. |
| Flow of events | 1. The user initiates the application. 2. The system redirects the user to the log in page. 3. The users enter the email and password. 4. The system checks the log in user and redirect to the home page. |
| Exit Condition | email and Password are validated |

**Table 12 Use case Set Appointment**

|  |  |
| --- | --- |
| Name | Set Appointment |
| Actors | Patient |
| Entry Conditions | The Log in Page should be Loaded. |
| Flow of events | 1.The user initiates the application.  2.The system redirects the user to the log in page.  3.The users enter the name and password.  4.The system checks the log in user and redirect to the home page.  5. Choose Doctor  6. Pick comfortable date and time set Appointment |
| Exit Condition | Set the appointment to doctor |

**Table 13 Use case Follow up Doctor**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | Follow up Doctor | |
| Actors | | Patient | |
| Entry Conditions | | The Log in Page should be Loaded. | |
| Flow of events | | 1.The user initiates the application.  2.The system redirects the user to the log in page.  3.The users enter the name and password.  4.The system checks the log in user and redirect to the home page.  5. Choose Doctor  6. Send question or his/her status to doctor. | |
| Exit Condition | | * Send to doctor | |

**Table 14 Use case View Data**

|  |  |
| --- | --- |
| Name | View Data |
| Actors | Patient |
| Entry Conditions | The Log in Page should be Loaded. |
| Flow of events | 1. The user initiates the application.  2. The system redirects the user to the log in page.  3. The users enter the name and password.  4. The system checks the log in user and redirect to the home page.  5. Choose Previous appointment or Previous follow up option |
| Exit Condition | * View selected data |

**Table 15 Use case Response Appointment**

|  |  |
| --- | --- |
| Name | Response Appointment |
| Actors | Doctor |
| Entry Conditions | The Log in Page should be Loaded. |
| Flow of events | 1. The user initiates the application.  2. The system redirects the user to the log in page.  3. The users enter the name and password.  4. The system checks the log in user and redirect to the home page.  5. Approve or Decline the Appointment that made by patient |
| Exit Condition | * Response to the Appointment. |

**Table 16 Use case Follow Up Patient**

|  |  |
| --- | --- |
| Name | Follow Up Patient |
| Actors | Doctor |
| Entry Conditions | The Log in Page should be Loaded. |
| Flow of events | 1. The user initiates the application.  2. The system redirects the user to the log in page.  3. The users enter the name and password.  4. The system checks the log in user and redirect to the home page.  5. Response the Patient’s Question |
| Exit Condition | * Response to the patient. |

**Table 17 Use case Add Doctor**

|  |  |
| --- | --- |
| Name | Add Doctor |
| Actors | Admin |
| Entry Conditions | The Log in Page should be Loaded. |
| Flow of events | 1. The user initiates the application.  2. The system redirects the user to the log in page.  3. The users enter the name and password.  4. The system checks the log in user and redirect to the home page.  5. choose Add Doctor option.  6. Fill all required Field and submit. |
| Exit Condition | * Register new doctor |

**Table 18 Use case Delete Doctor**

|  |  |
| --- | --- |
| Name | Delete Doctor |
| Actors | Admin |
| Entry Conditions | The Log in Page should be Loaded. |
| Flow of events | 1. The user initiates the application.  2. The system redirects the user to the log in page.  3. The users enter the name and password.  4. The system checks the log in user and redirect to the home page.  5. choose Edit Doctor option.  6. the System will list Register doctor then press delete option. |
| Exit Condition | * Delete doctor |

* + 1. **Object Model**

List of the fundamental objects:

* **Entity Object**: Patient, Doctor, Admin
* **Boundary Object**: registration form, login form, appointment form, follow up form,
* **Control Object**: register User, User login, set appointment, follow up, add doctor, edit doctor.
  + 1. **Data Dictionary**

**Table 19 Patient Register**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Patient Register** | | | | |
| Attributes | Data Type | Index | Null Value | Description |
| P\_id | Int(11) | Primary key | Not |  |
| Fname | varchar(100) |  | Not | First name |
| Lname | varchar(100) |  | Not | Last name |
| Email | varchar(100) |  | Not |  |
| Password | varchar(100) |  | Not |  |
| Contact | varchar(100) |  | Not |  |
| Age | Int(3) |  | Not |  |
| Emerg\_name | varchar(100) |  | Not | Emergence  name |
| Emerg\_contact | varchar(100) |  | Not | Emergence contact |

**Table 20 Doctor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Doctor** | | | | |
| Attributes | Data Type | Index | Null Value | Description |
| D\_id | Int(11) | Primary key | Not |  |
| Fname | varchar(100) |  | Not | First name |
| Lname | varchar(100) |  | Not | Last name |
| Email | varchar(100) |  | Not |  |
| Password | varchar(100) |  | Not |  |
| Contact | varchar(100) |  | Not |  |
| Department | varchar(100) |  | Not |  |
| img | blob() |  | Not | Doctor profile image |

**Table 21 Appointment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Appointment** | | | | |
| Attributes | Data Type | Index | Null Value | Description |
| app\_id | Int(11) | Primary key | Not |  |
| P\_id | int(100) | Foreign key | Not | Patient’s id |
| D\_id | int(100) | Foreign key | Not | Doctor’s id |
| pn | varchar(100) |  | Not | Patient’s name |
| Dn | varchar(100) |  | Not | Doctor’s name |
| Date | datetime(2) |  | Not | Schedule |
| permission | varchar(100) |  | Not | Status(approved or decline) |

**Table 22 Follow up**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Follow up** | | | | |
| Attributes | Data Type | Index | Null Value | Description |
| f\_id | Int(11) | Primary key | Not |  |
| P\_id | int(100) | Foreign key | Not | Patient’s id |
| D\_id | int(100) | Foreign key | Not | Doctor’s id |
| P\_case | varchar(100) |  | Not | Patient’s question |
| D\_ans | varchar(100) |  | Not | Doctor’s answer |

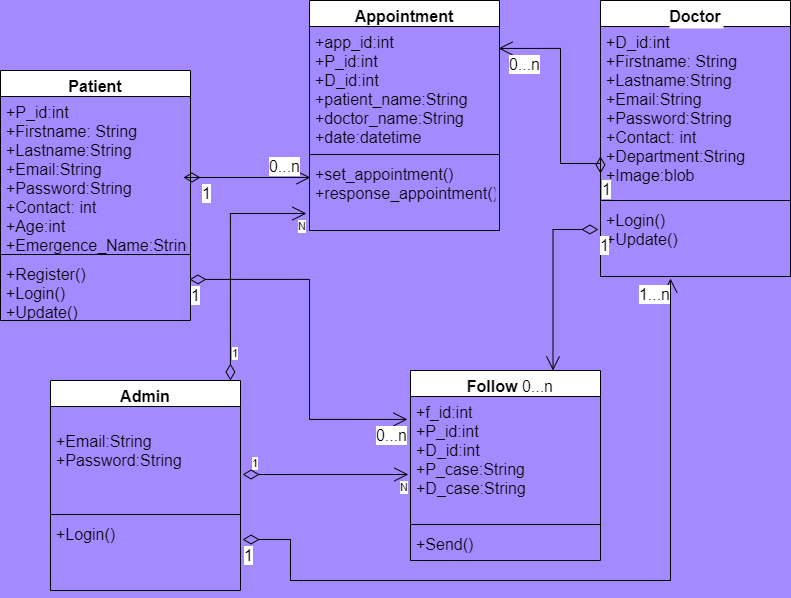
**Table 23 Admin**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Admin** | | | | |
| Attributes | Data Type | Index | Null Value | Description |
| id | Int(11) | Primary key | Not |  |
| Fname | Varchar(100) |  | Not |  |
| email | Varchar(100) |  | Not |  |
| password | Varchar(100) |  | Not |  |

* + 1. **Class Diagram**

Classes, associations, attributes and operations

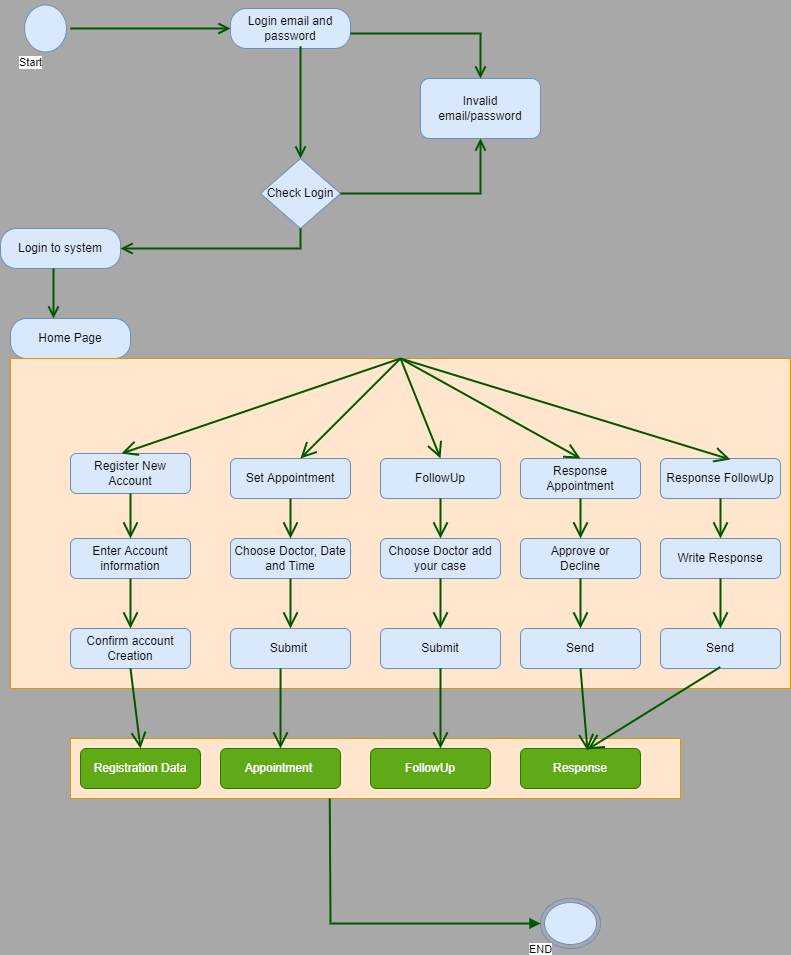
**Figure 2 class diagram**



* + 1. **Dynamic Models**

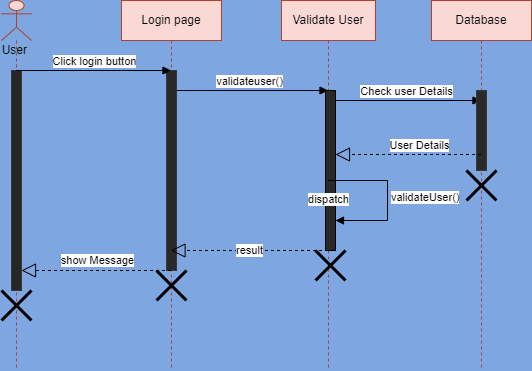
### **2.4.6.1 State Diagram**

**Figure 3: State diagram**

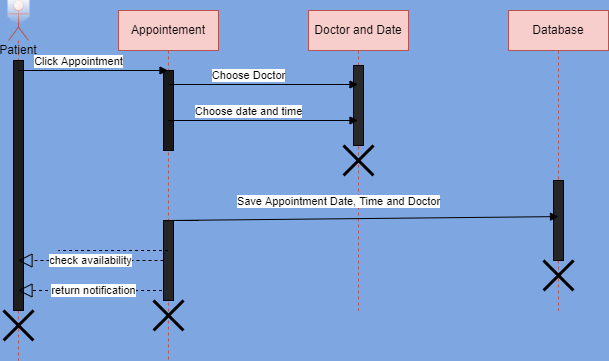
****

**2.4.6.2 Sequence Diagram**

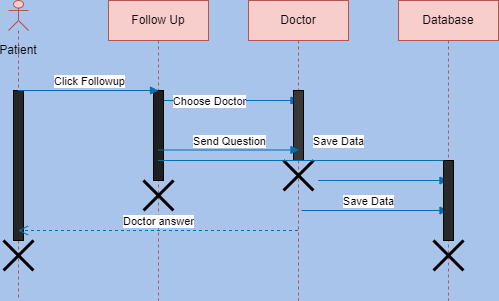
**Figure 4: Sequence diagram for Login**

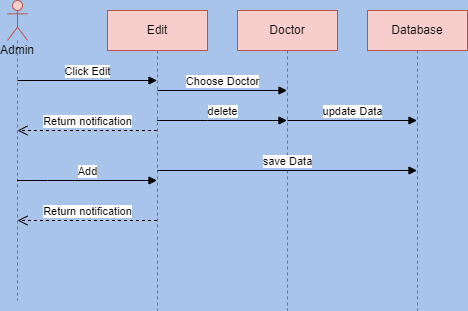


**Figure 5: Sequence diagram for Appointment**



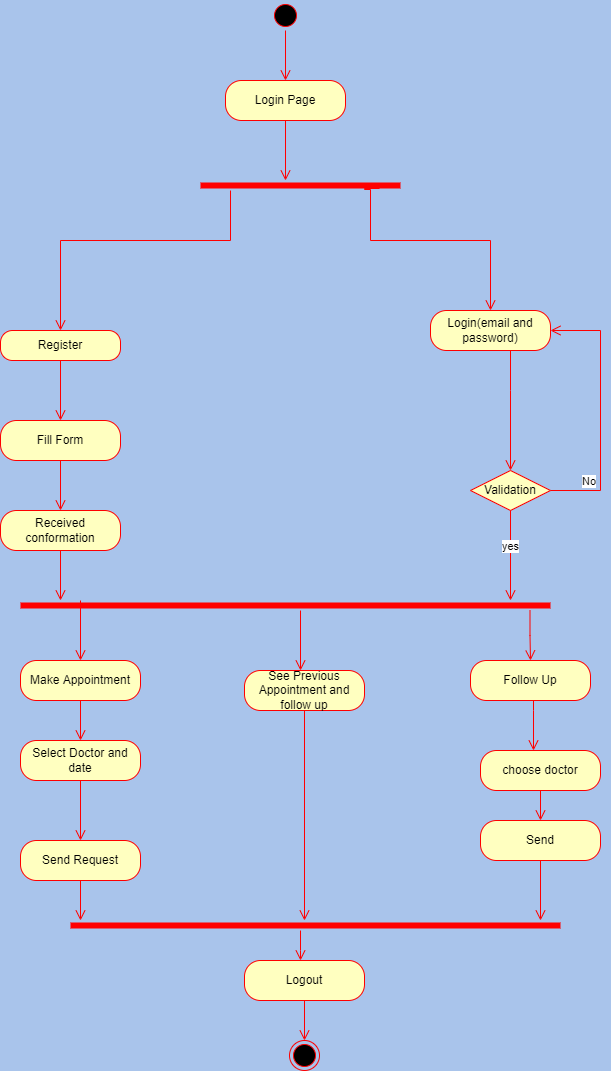
**Figure 6: Sequence diagram for Patient Follow Up**

**Figure 7: Sequence diagram for admin**

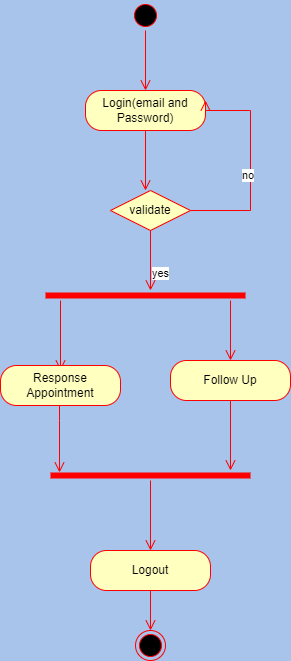


#### **2.4.6.3 Activity diagram**

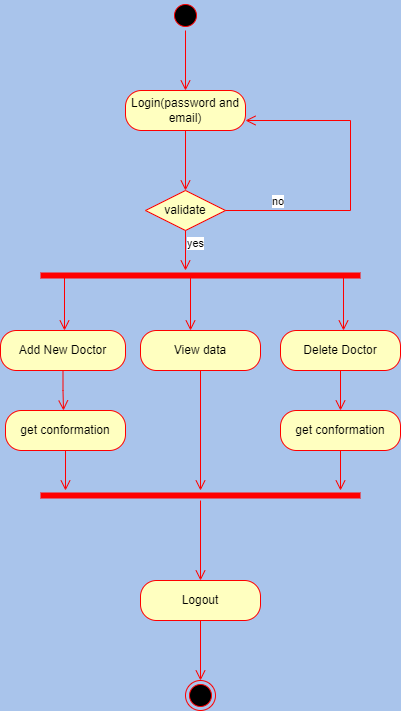
**Figure 8: Activity diagram for patient**

****

**Figure 9: Activity diagram for doctor**

****

**Figure 10: Activity diagram for admin**

****

**CHAPTER 3**

**3. SYSTEM DESIGN DOCUMENT**

**3.1 Introduction**

This system design document describes the system requirement, architecture, and database design of an E-health system. This system will provide patient registration, appointment making, and other such related services on an easy-to-use platform.

Patients will be able to login into the system and choose which healthcare provider they would like to consult and set a date of consultation, then the healthcare provider will be notified of the request and upon their acceptance of the appointment will be sent the patients history and any other information that they may need to properly care for that patient.

## 3.1.1 Purpose

The purpose of this project is to create a simple to use and access online system that provides both patients and doctors with easier contact procedures that will save them both a lot of time and effort.

## 3.1.2 Scope

The proposed software product is the E-Health System. In this project, we are going to design and build a functional web-based E-Health System.

This project aims to create an E-Health System that Patients can able to book doctor’s appointments and follow up with doctors (communicate after appointment). Doctors have access to view appointments and set available times for appointments thereby making it more convenient for them and follow-up patients. The admin also has access to a database to add or delete doctors on the system.

The system will be a web-based online application that is available on all major browsers. should be user-friendly on all usual devices (PC, mobile, tablet).

## 3.2 Goals and Trade-Offs

### **3.2.1 Goals**

1. **Functionality**: The basic function of the system is to create a platform that is used to reduce the burden of all the tiring and time-wasting that goes into traditional appointment systems.
2. **Usability:** The system provided will be an effective as well as an efficient system in which it will help provide services that help the user to achieve their goal easily, and also users will spend minimum effort and resources to achieve their goal.
3. **User-friendly interface:** This means that the system is very easy to operate as well as learn and understand by anyone who is either a person familiar with technology or one who is not by giving a well-made and interactive interface that can allow making it very easy to use.
4. **Security:** The system will be fully secured along with any unauthorized access to the system should be restricted. Through our login feature only authorized users can access the system moreover some of the users will have different capabilities they can do in the system and the login information will be stored on the database.
5. **Fault Tolerance:** The system should be able to give a response error message when the user enters incorrect input. This recommends the user enter the correct input.

## 3.2.2 Trade-offs

Our design priority for this project was to make a reliable and simple understanding and easy-to-use app and website.

**Rapid prototyping vs. completeness of functionality**

This system was conceptualized to be used both on the web and on phones wherever internet data could be found but due to time shortage, we are forced to use the prototype on one computer.

**Usability vs. functionality**

Since making, many functions in one interface is an annoyance for the user that makes interfaces more difficult to learn and use. Therefore, our system alleviates this problem by allocating specific and similar tasks on one interface.

**Efficiency vs. portability:**

Our application will run in any android operating system and can be transferred from one android mobile environment to another with a good use of processor and memory.

**Cost vs. Reliability:**

For the reliability of the software tests, will write for the main function to make it error free. In addition, all these possible and predetermined errors from the user side will be handled without affecting the cost of the system.

**Reusability vs. Cost**

To make the development cost our classes and functions are designed to be reused in the code, we will not write the classes with some functions two times; rather, we follow the design pattern that forces us to use the re-usability of the code. This saves the development time, which leads to decreasing the development cost.

# **3.3 System Decomposition**

In total three sub-systems build up this project. A brief description of these subsystems is:

1. The patient sub-system is responsible for choosing doctors, making appointments, and follow-ups.
2. The doctor sub-system is responsible for accepting or declining a patient's request for an appointment or follow-up.
3. The admin sub-system is responsible for adding or removing doctors from the system.

The sub-systems do interact with each other depending on the circumstance for example for a patient to choose a doctor the system has to retrieve data from a list of doctors from the doctor's database.

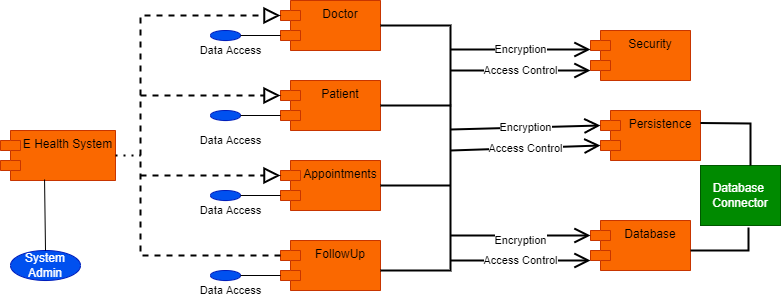
## 3.3.1 Layers and Partitions

The dependency of the systems on each other is not hierarchy one because the sub-systems are equally dependent on each other.

## 3.3.2 System Topology

* + 1. Component Diagram Component diagrams show certain aspects of relationships that are not apparent in other class diagrams. The relationship between provided and required interfaces helps the reader understand the design better. The ultimate purpose is to help the reader of the system design document to understand the design and the spectrum of choices available for design and implementation.

**Figure 11 Component Diagram**



* + 1. Deployment Diagram

The deployment diagram represents the hardware topology used and the runtime system assigned. The hardware encompasses processing units in the form of nodes as well as communication relationships between the nodes. A runtime system contains artifacts that are deployed to the nodes.

**Figure 12 Deployment Diagram**

# **3.4 Concurrency Identification**

The system can provide access to multiple users from different machines, and since the database server is multi-threaded it can handle several users at a time.

In Dart, we can accomplish concurrency by utilizing the Isolates. The isolates are autonomous

workers that don’t share memory however rather interconnect by ignoring message channels.

.

# **3.5 Hardware/Software Allocation**

* **Hardware requirement for server side:**
  + Processor: 5 Core, 2.5GHz
  + Secondary Storage: 1TB HDD
  + ROM: 52X CD ROM Drive
  + Memory: 86GB RAM
  + Network Adapter: Ethernet Adapter
  + Modem: 1mbps
  + Others: 17” Color Monitor, Printer, Keyboard, Mouse.
  + The system needs to be operating 24 hours; hence, power supply should be 100% assured.
* **Software Interface for the server side:**
  + Platform: Windows, Latest versions are recommended
  + Operating System: Windows 8 and above
  + Back-end tool: JavaScript
  + The system has PHP v8.0 that is used to load and compile server side scripts
* **Software Interface for the client side**

- Mobile android operating system above API 21

- The system browser should be JavaScript enabled so that it uses JavaScript for client side scripting and compilation of Web Pages.

- The system browser should be JavaScript enabled so that it uses JavaScript for client side scripting and compilation of Webpages.

## 3.5.1 System Performance

### **3.5.1.1 General system performance**

The system is expected to respond to requests within a few seconds not exceeding 4-8 seconds. For tasks that may take longer it is desired to respond in the 1-minute else request the user to reload.

### **3.5.1.2 Input/Output Performance**

There is no need for extra hardware to handle the data generation rate of the system. The existing communication bandwidth is sufficient to support all the communication between subsystems.

### **3.5.1.3 Processor allocation**

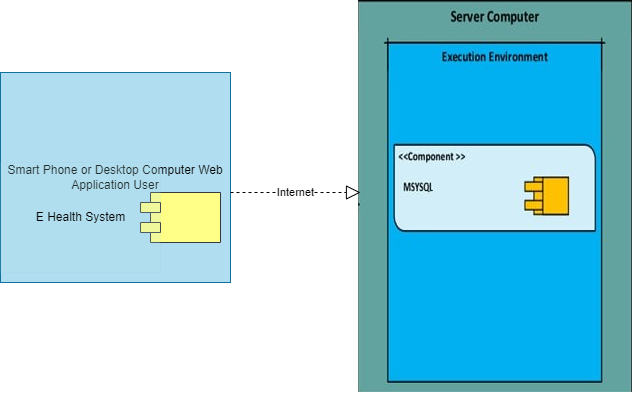
The computation rate is very small compared to other applications that the deployment system’s process can handle normally

### **3.5.1.3 Memory allocation**

There is enough memory to handle requests. The traffic generated by the application is medium.

## Connectivity

**Figure 13 Connectivity Diagram**



## 3.5.3 Network architecture

The general interaction mechanism and protocol for the channels and the communication are synchronous communication so that the server should be available for the client, otherwise the communication will be disconnected. The expected bandwidth for the networking is 25MB/S.

# **3.6 Data Management**

The data generated by the system is saved on a single server’s database using MySQL. and the data is not distributed as a backup on every weak end and the data is accessed very often while the system is being used.

# **3.7 Global Resource Handling**

There is a multitude of resources that are available for different kinds of users but, there needs to be a differentiation between the Patients, the doctors, and Admin. We accomplished this by having different sets of user authentication for these different kinds of users. The authentication scheme used in this program is password-based authentication. The patients for example will log in with their patient's id or email address and password and can access the only needed parts to make appointments and follow-ups with a doctor whereas Doctors will have the choice to respond to patients that request appointments or follow-ups with them and nothing more. The resources are partitioned into different categories depending on who is permitted to access them, and the type of authentication used to log in will determine which sets of resources will be made available to that user.

# **3.8 Software Control Implementation**

## 3.8.1 External control flow (between subsystems)

First of all, we have an email or id and password authentication model to authenticate users and detect and differentiate whether the user is a patient, doctor, or admin and a MySQL-based database in which all data management occurs as well as all the different privileges are stored, the admin is capable of adding or removing doctors and accessing appointment history of patients. But a standard user is only capable of accessing only portions of the subsystems. All of the tools provided by the system can be accessed via login with different permissions

## 3.8.2 Concurrent control

In this system, there are no subsystems that run concurrently

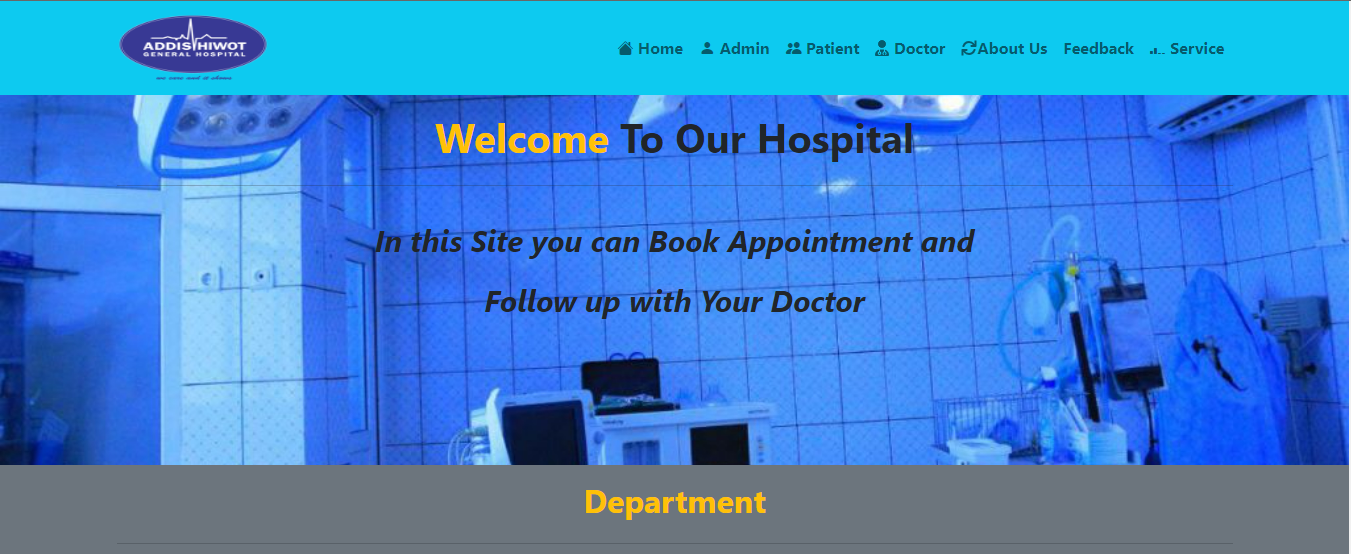
## 3.8.3 Internal control (within a single process)

All of the subsystems found in our system will be modeled as to the specifications of the user, for making the whole system more useable and efficient. Also, the way the subsystems can communicate with each other is through complete access to the database. And the whole system and subsystems were designed to have better usability and functionality by allowing page-to-page transference when an object is clicked.

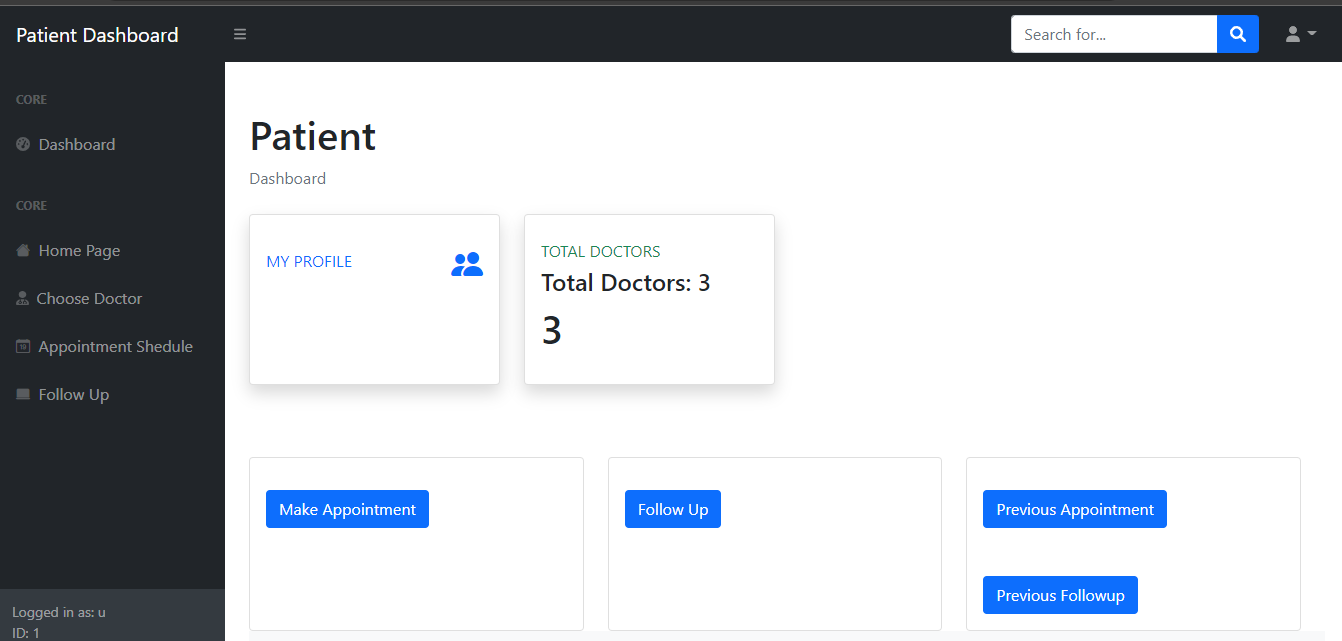
## 3.8.4 User Interface

Some of the user interfaces are shown below:

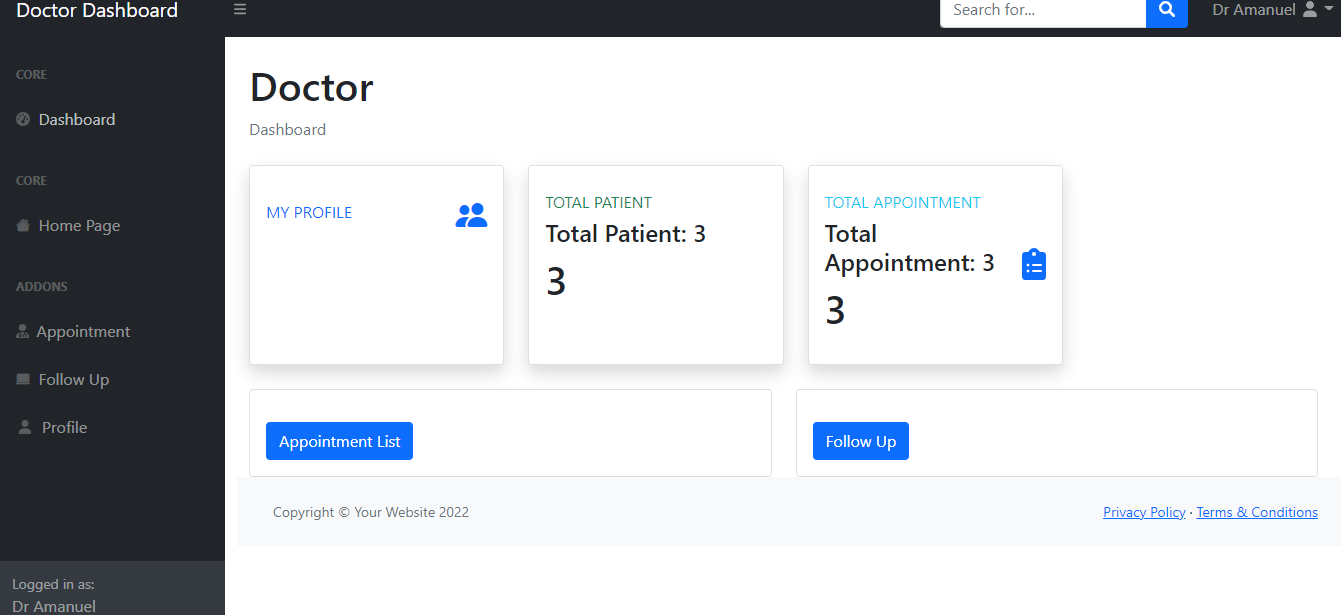
**Figure 14 Home Page**



**Figure 15 Patient Home Screen**



**Figure 16 Doctor Home Screen**



# **3.9 Boundary Conditions**

## 3.9.1 Initialization

Before opening the system the user first needs to turn on xampp server so the system can load needed data on start-up after that the system can be opened from an application or website which will first show the front page users can choose not to log in to if they want but they will not be able to access any of the system's functions.

## 3.9.2 Termination

The Termination conditions for the system are as soon as the users sign out or destroy the life Cycle of the application. At the time of termination, subsystems will be notified of any action they need to be perform. Database commits will be done by the subsystems when the system terminates. And the database needs to be closed personally by the user.

## 3.9.3 Failure

Any incomplete transaction will be rolled back to the point where it was before the system process started.

# 3.10 Design Rationale

* **Maintainability**
  + The system will be maintained periodically through effective monitoring and evaluation. This will go a long way to help identify and debug emergency production problems and address them accordingly. A considerable amount of time would be spent to effect changes in data, files, reports, hardware, and software.
  + That means we can remain agile and keep moving out to new releases quickly.
* **Testability**
* **Functionality Testing**

All functions in the application, database connection, and forms used to enter data for submission, editing, and getting or deleting information from users were tested. Developers performed the test of the website. Some functionality requirements were tested during the test security and Database of the system test.

* **Extensibility**
  + we will have a better chance of making any of those parts more reusable.

# **CHAPTER 4**

# **4 IMPLEMENTATION**

## 4.1 Introduction/Overview

in order to develop this system from web perspective we use different types of software and hardware consideration for instance we use PHPv8 the web server side scripting and for user interface we used HTML, CSS and Java script for validation and MySQL were used for Database design and storage.

The E Health System well elaborated in the previous section of the document will have the following implementation details.

## 4.2 Tools and Technology Utilized for Implementation

* PHP.
* MySQL.
* Xampp
* Web Browser (Chrome, Edge).

## 4.3 System Implementation

### **4.3.1 System Interfaces**

**4.3.2 System Algorithm**

### **4.3.3 Source Code**

//patient login, register, update, set appointment and follow up

**<?php**

**session\_start();**

**require\_once 'db.php';**

**// create account for Patient**

**$email=mysqli\_real\_escape\_string($connection,$\_POST['email']);**

**$password=mysqli\_real\_escape\_string($connection,$\_POST['password']);**

**if(isset($\_POST['Createpatient'])){**

**$Fname =mysqli\_real\_escape\_string($connection, $\_POST['First\_Name']);**

**$Lname =mysqli\_real\_escape\_string($connection, $\_POST['Last\_Name']);**

**$email =mysqli\_real\_escape\_string($connection, $\_POST['Email']);**

**$password =mysqli\_real\_escape\_string($connection, $\_POST['password']);**

**$contact = mysqli\_real\_escape\_string($connection,$\_POST['Contact']);**

**$age = mysqli\_real\_escape\_string($connection,$\_POST['Age']);**

**$Emerg\_name = mysqli\_real\_escape\_string($connection,$\_POST['Emerg\_name']);**

**$Emerg\_contact = mysqli\_real\_escape\_string($connection,$\_POST['Emerg\_contact']);**

**$insert\_query = "INSERT INTO patient (Fname, Lname, Email, password, Contact, Age, Emerg\_name, Emerg\_contact ) VALUES('$Fname','$Lname','$email','$password','$contact','$age','$Emerg\_name','$Emerg\_contact')";**

**$query\_run = mysqli\_query($connection, $insert\_query);**

**if($query\_run){**

**$\_SESSION['status'] = "Your Account is Created";**

**header("Location: Login.php")**

**}**

**else{**

**$\_SESSION['status'] = "Your Account is Not Created Try Again";**

**header("Location: CreateAccount.php")**

**die;}**

**// Patient Login**

**require\_once('db.php');**

**if(isset($\_POST['PatientLogin'])){**

**if(empty($\_POST['email']) || empty($\_POST['password'])){**

**$\_SESSION['status'] = "Please fill it";**

**header("location:Login.php");**

**}**

**else{**

**$query="SELECT \* from patient where email='$email' and password= '$password'";**

**$result=mysqli\_query($connection,$query);**

**if(mysqli\_num\_rows($result)>0){**

**$row=mysqli\_fetch\_assoc($result);**

**$\_SESSION['ID']=$row['P\_id'];**

**header("location:patients.php");**

**}**

**else{**

**$\_SESSION['status'] = "Email / Password is Invalid";**

**header("location:Login.php");**

**}}}**

**//UPdate profile**

**if(isset($\_POST['update'])){**

**$id= $\_SESSION['ID'];**

**$Fname = mysqli\_real\_escape\_string($connection,$\_POST['Fname']);**

**$Lname = mysqli\_real\_escape\_string($connection, $\_POST['Lname']);**

**$email = mysqli\_real\_escape\_string($connection, $\_POST['email']);**

**$password = mysqli\_real\_escape\_string($connection, $\_POST['password']);**

**$contact = mysqli\_real\_escape\_string($connection,$\_POST['contact']);**

**$age = mysqli\_real\_escape\_string($connection, $\_POST['age']);**

**$En = mysqli\_real\_escape\_string($connection, $\_POST['Emerg\_name']);**

**$Ec = mysqli\_real\_escape\_string($connection,$\_POST['Emerg\_contact']);**

**$insert\_query = "UPDATE patient SET Fname='$Fname',Lname='$Lname', email='$email', password='$password', contact='$contact',**

**age='$age',Emerg\_name='$En',Emerg\_contact='$Ec' WHERE P\_id=$id ";**

**$q=mysqli\_query($connection,$insert\_query);**

**if($q){**

**$\_SESSION['status'] = "Your Account is Created";**

**header("Location: Login.php");**

**}**

**else{**

**$\_SESSION['status'] = "Your Account is Not Created Try Again";**

**header("Location: CreateAccount.php"); }die;}**

**// Patient Login by id**

**require\_once('db.php');**

**if(isset($\_POST['logid'])){**

**if(empty($\_POST['email']) || empty($\_POST['id'])){**

**$\_SESSION['status'] = "please fill it";**

**header("location:password.php");**

**}**

**else{**

**$query="SELECT \* from patient where email='".$\_POST['email']."' and p\_id ='".$\_POST['id']."'";**

**$result=mysqli\_query($connection,$query);**

**if(mysqli\_num\_rows($result)>0){**

**$row=mysqli\_fetch\_assoc($result);**

**$\_SESSION['ID']=$row['P\_id'];**

**header("location:patients.php");**

**}**

**else{**

**$\_SESSION['status'] = "Email / ID is Invalid";**

**header("location:password.php");**

**}}}**

**// Ended Patient Login**

**// Appointment**

**if(isset($\_SESSION['ID'])){**

**if(isset($\_GET['ap'])){**

**$id=$\_GET['ap'];**

**if(isset($\_POST['send'])){**

**$dat=$\_POST['d'];**

**$DI=$id;**

**$p\_id=$\_SESSION['ID'];**

**//extract patient name**

**$ss="SELECT Fname FROM patient WHERE P\_id=$p\_id";**

**$qq=mysqli\_query($connection,$ss);**

**$rr=mysqli\_fetch\_assoc($qq);**

**$pn=$rr['Fname'];**

**//extract doctor name**

**$sss="SELECT Fname From doctor WHERE D\_id=$DI";**

**$qqq=mysqli\_query($connection,$sss);**

**$rrr=mysqli\_fetch\_assoc($qqq);**

**$dn=$rrr['Fname'];**

**//set the value into table appointment (app)**

**$s="INSERT INTO app (`app\_id`,`P\_id`,`D\_id`,`pn`,`dn`,`date`) VALUES(DEFAULT,'$p\_id','$DI','$pn','$dn','$dat')";**

**$q=mysqli\_query($connection,$s);**

**if($q){**

**header('Location:patients.php');**

**} else{**

**echo "error";**

**} }**

**include('included/header.php');**

**?> <body style="background-color:grey">**

**<div class="container bg-secondary w-100 p-3">**

**<form method='POST' action=""<center><label>Date <input type="datetime-local" name="d"></label><br><br>**

**<button name='send' class="btn btn-primary">set</button></center> </form></div> </body>**

**<?php**

**}}**

**else{**

**header("Location:Login.php");}**

**//followup**

**if(isset($\_SESSION['ID'])){**

**if(isset($\_GET['fdi'])){**

**$id=$\_GET['fdi'];**

**if(isset($\_POST['setf'])){**

**echo "good";**

**$P\_case=$\_POST['P\_C'];**

**$DI=$id;**

**$p\_id=$\_SESSION['ID'];**

**//extract patient name**

**//set the value into table appointment (app**

**$s="INSERT INTO followup (`F\_id`,`P\_id`,`D\_id`,`P\_case`,`D\_ans`)VALUES(DEFAULT,'$p\_id','$DI','$P\_case','')";**

**$q=mysqli\_query($connection,$s);**

**if($q){**

**header('Location:patients.php');**

**} else{ echo "<script>alert('Error');</script>"; }**

**}include('included/header.php');?>**

**<!DOCTYPE html>**

**<body style="background-color:grey"> <div class="container bg-secondary w-100 p-3">**

**<form method='POST' action=''>**

**<center><label>Patient Case <textarea name="P\_C" id="" cols="30" rows="5"></textarea></label><br><br>**

**<button name='setf' class="btn btn-primary" >Send</button></center> </form></div></body></html>**

**<?php }}else{**

**header("Location:Login.php");}**

**?>**

# **CHAPTER 5**

# **5 CONCLUSION AND FUTURE WORK**

## 5.1 Conclusion

For the current manual method, the web-based E Health System is a more automated computerized substitute. This system was primarily created so that patients could schedule appointments online and follow up with their doctors. It provides efficient solutions, improving efficiency.

## 5.2 Future work

We have further functions that can be added to the present ones or even premium additions planned for the future as people become accustomed to the system. these are

* Online Payment
* Live talk patient and doctor.

# **Glossary**

|  |  |
| --- | --- |
| **System** | A set of inter-connected computer equipment and/or programs used together for a particular purpose and operating together. |
| **Action** | A fundamental unit of behavior. Actions can take a set of inputs, produce a set of outputs, and change the state of the system. Actions are fundamental in the sense that they are not decomposed further. |
| **Actor** | An entity that needs to interact with the system for exchange information. An external entity that needs to exchange information with the system. An actor can represent either a user role or another system |
| **Scenario** | A short text description on the system function. |
| **Use case** | An action implemented by the computer system. |
| **Diagram** | A simple plan which represents a machine, system or idea, etc., often drawn to explain how it works. |
| **Class diagram** | Is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects |
| **Sequence Diagram** | A diagram which represents sequence of action that human or system represents. |
| **State diagram** | Is a type of diagram used to describe the behavior of system |
| **Server** | A computer system that provides services to other computing systems over a network. |
| **Users** | Student, Student Dean, Department Head, Academic Dean and Instructors. |
| **IT** | Information Technology |
| **CSS** | Cascading Style Sheet is used to format the layout of web page. |
| **DB** | Data Base Is an organized collection of data |
| **JS** | **Javascript** Is a programming language that is used to create application for android |
| **MySQL** | Is a relational database system |
| **PHP** | Is a scripting language and used to develop dynamic web application together with MySQL, HTML, CSS and Java Script. |
| **Xampp** | Server |
| **FR** | Functional Requirements |
| **NFR** | Non- Functional Requirements |
| **RAD** | Requirement Analysis Document |
| **GUI** | Graphical User Interface |