

**DoctorDESK**  
**A PROJECT REPORT**

*Submitted by*

<b>Aastha Patel</b>	<b>170770116052</b>
<b>Vidhi Patel</b>	<b>170770116088</b>
<b>Manushi Patel</b>	<b>170770116072</b>

*In fulfilment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

*in*

Information Technology



Silver Oak College of Engineering & Technology  
Opp. Bhagwat Vidhyapith, Near Gota Cross Road, Ahmedabad-382481

**Gujarat Technological University, Ahmedabad**

Academic Year 2020-21

## Candidate's Declaration

I/We hereby declare that project report titled “**DoctorDESK**” submitted towards the completion of project in 8<sup>th</sup> semester of Bachelor of Information Technology in Silver Oak College Of Engineering & Technology, Ahmedabad is an authenticate record of our work carried out.

I/We further declare to the best of our knowledge the report of I.T. 8<sup>th</sup> Semester.

Candidate's Signature	:
Candidate's Name	: <b>Aastha Patel</b>
Branch	: <b>IT</b>
Enrollment Number	: <b>170770116052</b>

Candidate's Signature	:
Candidate's Name	: <b>Vidhi Patel</b>
Branch	: <b>IT</b>
Enrollment Number	: <b>170770116088</b>

Candidate's Signature	:
Candidate's Name	: <b>Manushi Patel</b>
Branch	: <b>IT</b>
Enrollment Number	: <b>170770116072</b>

**GTU PMMS PORTAL  
GENERATED COMPLETION  
CERTIFICATE  
(Each Student Individual Copy)**

**Silver Oak College of Engineering & Technology**  
Department of Information Technology  
2020-21

**CERTIFICATE**

**Date:**

This is to certify that the Project Work entitled “DoctorDESK” has been carried out by AASTHA PATEL (170770116052), VIDHI PATEL (170770116088) and MANUSHI PATEL (170770116072) under my guidance in fulfilment of the degree of Bachelor of Engineering in Information Technology (8<sup>th</sup> Semester) of Gujarat Technological University, Ahmedabad during the academic year 2020-21.



**Prof. Bhavin Trivedi**

Internal Guide

SOCET

**Prof. Jaimin Dave**

Head - Information Technology

SOCET

## ACKNOWLEDGEMENT

We would like to extend our heartily thanks with a deep sense of gratitude and respect to all those who has provided us immense help and guidance during our project. We would like to express our sincere thanks to our faculty guide **Prof. Bhavin Trivedi** for providing a vision about the system and for giving us an opportunity to undertake such a great challenging and innovative work. We are grateful for the guidance, encouragement, understanding and insightful support given in the development process.

We would like to extend my gratitude to **Prof. Jaimin Dave** head of Information Technology Engineering Department, Silver Oak college of Engineering and Technology, Ahmedabad, for his continuous encouragement and motivation.

Last but not the least we would like to mention here that we are greatly indebted to each and everybody who has been associated with our project at any stage but whose name does not find a place in this acknowledgement.

Yours Sincerely,

Aastha Patel (170770116052)  
Vidhi Patel (170770116088)  
Manushi Patel (170770116072)

## ABSTRACT

**‘DoctorDESK’** is a software that will revolutionize the habits to contact the local clinics, which involves 3-layer system. This software will manage overall interaction between the doctor and patients and many more functionalities such as right from doctor’s appointment, to the patient history.

This software provides a way to effectively control record & track patient’s traffic.

Basically, this software will be used by 3 types of user:

- Admin
- Patient
- Doctor

**Certificate** obtained from the Plagiarism checking software.  
**Undertaking About Originality of Work**

## LIST OF FIGURES

Figure No.	Figure Name	Page No.
2.1	Incremental model	10
2.2	Pie chart	11
2.3	Roles & Responsibilities	12
2.4	Gantt Chart	13
4.1	Use case diagram – Patient	23
4.2	Use case diagram - Doctor	24
4.3	Use case diagram - Admin	25
4.4	Activity diagram – Patient	26
4.5	Activity diagram - Admin	27
4.6	Activity diagram - Doctor	28
4.7	Sequence diagram – User	30
4.8	Sequence diagram – Admin	30
4.9	Sequence diagram - Doctor	31
4.10	E-R diagram	32
4.11	Class diagram	34
5.1	Flow chart	40
5.4	State chart diagram	43
6.1	Coding screenshot	47
6.2	Coding screenshot	47
6.3	Coding screenshot	48
7.1	Testing strategy	51
8.1	User login screenshot	55
8.2	User dashboard screenshot	55
8.3	User book appointment screenshot	56
8.4	User appointment history screenshot	56
8.5	Doctor dashboard screenshot	57
8.6	Doctor manage patient screenshot	57
8.7	Doctor add medical history screenshot	58
8.8	Admin dashboard screenshot	58
8.9	Admin add doctor screenshot	59



## LIST OF TABLES

<b>Table No.</b>	<b>Table Name</b>	<b>Page No.</b>
2.1	Project plan	10
2.2	Milestone and deliverable	12
5.1	Admin table	36
5.2	Appointment table	36
5.3	Doctor table	37
5.4	Doctor login table	37
5.5	Doctor specialization table	37
5.6	Contact us table	38
5.7	Medical history table	38
5.8	Patient table	38
5.9	Patient login table	39
5.10	Patient registration table	39

## **TABLE OF CONTENT**

<b>Chapter No.</b>	<b>Title</b>	<b>Page No.</b>
	<b>ACKNOWLEDGEMENT</b>	<b>iii</b>
	<b>ABSTRACT</b>	<b>iv</b>
	<b>LIST OF FIGURES</b>	<b>v</b>
	<b>LIST OF TABLES</b>	<b>vi</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	<b>1.1 Project Summary</b>	<b>2</b>
	<b>1.2 Purpose</b>	<b>2</b>
	<b>1.3 Scope</b>	<b>3</b>
	<b>1.4 Technical and Literature Review</b>	<b>3</b>
<b>2</b>	<b>SOFTWARE PROJECT MANAGEMENT</b>	<b>7</b>
	<b>2.1 Project Planning and Scheduling</b>	<b>8</b>
	2.1.1 Project Plan	<b>8</b>
	2.1.2 Scheduling	<b>8</b>
	<b>2.2 Project Development Approach</b>	<b>8</b>
	<b>2.3 Project Plan</b>	<b>10</b>
	2.3.1 Milestone and Deliverables	<b>11</b>
	2.3.2 Roles and Responsibilities	<b>12</b>
	2.3.3 Gantt Chart	<b>13</b>
	2.3.4 Cost Estimation	<b>13</b>
<b>3</b>	<b>SYSTEM REQUIREMENT STUDY</b>	<b>14</b>
	<b>3.1 User Characteristics</b>	<b>15</b>
	3.1.1 Patient module	<b>15</b>
	3.1.2 Doctor module	<b>15</b>
	3.1.3 Administration module	<b>15</b>
	<b>3.2 Hardware and Software Characteristics</b>	<b>16</b>

	3.2.1 Hardware Requirements	16
	3.2.2 Software Requirements	16
	<b>3.3 Specific Characteristic</b>	<b>16</b>
	3.3.1 Functionality	16
	3.3.2 Usability	16
	3.3.3 Reliability	16
	3.3.4 Performance	16
	3.3.5 Supportability	17
	<b>3.4 Design Constrain</b>	<b>17</b>
	<b>3.5 Assumption and Dependencies</b>	<b>17</b>
<b>4</b>	<b>SYSTEM ANALYSIS</b>	<b>18</b>
	<b>4.1 Study of Current System</b>	<b>19</b>
	<b>4.2 Requirement of New System</b>	<b>19</b>
	4.2.1 Functional Requirement	19
	4.2.2 Non-functional Requirement	20
	<b>4.3 Feasibility Study</b>	<b>20</b>
	<b>4.4 Requirement Validation</b>	<b>22</b>
	<b>4.5 Function of the System</b>	<b>22</b>
	4.5.1 Use-Case	22
	4.5.2 Activity Diagram	26
	4.5.3 Sequence Diagram	29
	<b>4.6 Data Modelling</b>	<b>31</b>
	4.6.1 E-R Diagram	31
	4.6.2 Class Diagram	33
<b>5</b>	<b>SYSYEM DESIGN</b>	<b>35</b>
	<b>5.1 Database Design</b>	<b>36</b>
	<b>5.2 Data Dictionary</b>	<b>36</b>
	<b>5.3 Flow Chart</b>	<b>40</b>

	5.3.1 User	40
	5.3.2 Admin	41
	5.3.3 Doctor	42
	<b>5.4 Input / Output and Interface design</b>	<b>43</b>
	5.4.1 State chart diagram	43
<b>6</b>	<b>SYIMPLEMENTATION AND PLANNING DETAILS</b>	<b>44</b>
	<b>6.1 Implementation and planning details</b>	<b>45</b>
	<b>6.2 Security features</b>	<b>45</b>
	<b>6.3 Coding standards</b>	<b>46</b>
	<b>6.4 Sample Coding</b>	<b>47</b>
<b>7</b>	<b>TESTING</b>	<b>49</b>
	<b>7.1 Testing Plan</b>	<b>50</b>
	<b>7.2 Testing strategy</b>	<b>50</b>
	<b>7.3 Testing methods</b>	<b>51</b>
	<b>7.4 Test Cases</b>	<b>52</b>
<b>8</b>	<b>SCREENSHOT AND USER MANUAL</b>	<b>54</b>
<b>9</b>	<b>LIMITATION AND ENHANCEMENT</b>	<b>60</b>
	<b>9.1 Limitation</b>	<b>61</b>
	<b>9.1 Future Enhancement</b>	<b>61</b>
	<b>CONCLUSION</b>	<b>62</b>
	<b>REFERENCES</b>	<b>64</b>

# **CHAPTER: 1**

## **INTRODUCTION**

## CHAPTER :1

### INTRODUCTION

#### 1.1 Project Summary

- In this project, we are going to develop the php based software, which involves the multi clinics management.
- This model is useful from both the sides patient as well as the doctor.
- Seeing the Doctor's side, it helps them by interacting with more patients, viewing patient's medical history and managing appointment virtually.
- Seeing the Patient's side, it helps them know the local clinic doctors more efficiently and also book their appointment with the doctors according to their specialization.
- Taking Admin side into account, it can manage all the record of doctors as well as the patient's.
- Doctor's account is being created by admin and the patients can create their account by themselves as well as doctor can also create any patient's account if required.

#### 1.2 Purpose:

- Basically, 'DoctorDESK' software is built for the managing the multiple local clinics.
- And patient can operate it virtually for booking their appointments according to doctor's clinic time.
- The main reason behind making this project is to taking local clinic into market.

#### 1.3 Scope:

**The scopes are mention below:**

- DoctorDESK software which will provide platform to a lot of Patients for finding perfect doctor according to their specialization.
- There are different sectors like Registration, Book Appointment, Search, Medical History, Prescription History, Track History etc. Patient can directly search Doctor according to their required specialization.
- In the package module one can easily know about every doctor consultancy fees.

**1.4 Technology and Literature Review:**

- The front end used in our project is HTML and CSS, whereas the back end used is PHP.
- We will follow the Iterative model for developing this Project and whole Project will be developed using the PHP.

**❖ PHP**

- PHP is a popular general- purpose scripting language that is especially suited to web development.
- Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world.

**❖ SQL:**

- SQL (Structured Query Language) is a domain-specific language used in programming and designed for managing data held in a relational database

management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables.

❖ **Advantages:**

- Faster Query Processing – Large amount of data is retrieved quickly and efficiently.
- No Coding Skills – For data retrieval, large number of lines of code is not required.
- Standardized Language.
- Portable.
- Interactive Language.
- Multiple data views.

❖ **HTML:**

- The HyperText Markup Language, or HTML(HyperText Markup Language) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
- Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
- HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and



<input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

❖ **CSS:**

- Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
- CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.
- Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

❖ **Bootstrap:**

- Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.
- Bootstrap is among the most starred projects on GitHub, with more than 142,000 stars, behind free Code Camp (almost 312,000 stars) and marginally behind Vue.js framework

**❖ JavaScript:**

- JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.
- Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.
- As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).
- The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

# **CHAPTER: 2 SOFTWARE PROJECT MANAGEMENT**

## CHAPTER: 2

### SOFTWARE PROJECT MANAGMENT

#### 2.1 Project planning and scheduling

##### 2.1.1 Project Planning

- Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment.
- Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure.
- Project planning is often used to organize different areas of a project, including project plans, workloads and the management of teams and individuals.

##### 2.1.1 Project Scheduling

- Project Scheduling is the culmination of a planning activity that is primary component of software project management.
- When combined with estimation methods and risk analysis, scheduling, establishes a road map for the project management.
- Scheduling begins with the process composition. The characteristics of the project are used to adapt an appropriate task set for the work to be done.
- The task network is used to compute the critical project path, a time line chart and a variety of project information.

#### 2.2 Project Development Approach

The activities we followed for this project is listed below:

- Planning the work objectives

- Analysis & Design of objectives
- Assessing and controlling risk
- Allocation of resources
- Organizing the work

The Process Paradigm we used for our project is Incremental Model.

### **The Incremental Software Process Model**

- The Incremental Model combines elements of the linear sequential model with the iterative philosophy of prototyping. The incremental model applies linear sequences in a staged fashion as calendar time progresses.
- Each linear sequence produces a deliverable “increment” of the software. For example, word processing software developed using the incremental paradigm might deliver basic file management, editing and document production functions in the first increment; more sophisticated editing and document production capabilities in the second increment; spelling and grammar checking in the third increment; and advanced page layout capability in the fourth increment.
- It should be noted that the process flow for any increment can incorporate the prototyping paradigm.
- When an incremental model is used, the first increment is often a core product. That is, basic requirements are addressed, but many supplementary features remain undelivered.
- The core product is used by the user. As a result of use and/or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the user and the delivery of additional features and functionality.
- This process is repeated following the delivery of each increment, until the complete product is produced.
- The Incremental process model, like prototyping and other evolution approaches, is iterative in nature.
- But unlike prototyping, the incremental model focuses on the delivery of an operational product with each increment.

- Early increments are stripped down versions of the final product, but they do provide capability that serves the user and also provide a platform for evaluation by the user.

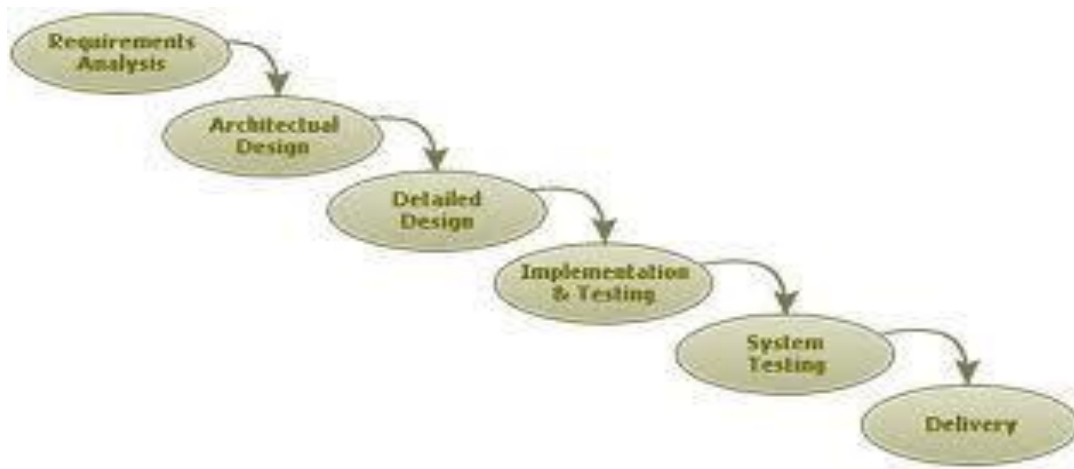
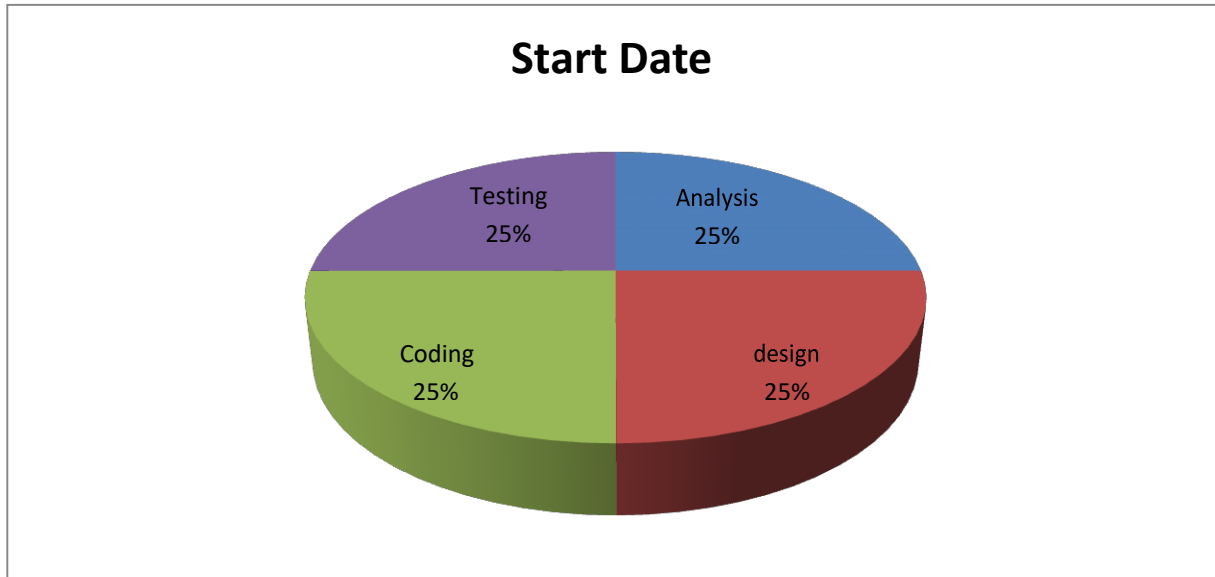


Figure 2.1 Incremental Model

## 2.3 Project Plan

	From Date	To Date
<b>1.Preliminary Investigation</b>	15/02/2021	26/02/2021
<b>2.Requirment Analysis</b>	27/02/2021	28/02/2021
<b>3.Designing</b>	5/03/2021	25/03/2021
<b>4.Implementation</b>	26/03/2021	
<b>5.Testing</b>		

[Table 2. 1 Project Plan]



[ Figure 2.2 Pie Chart]

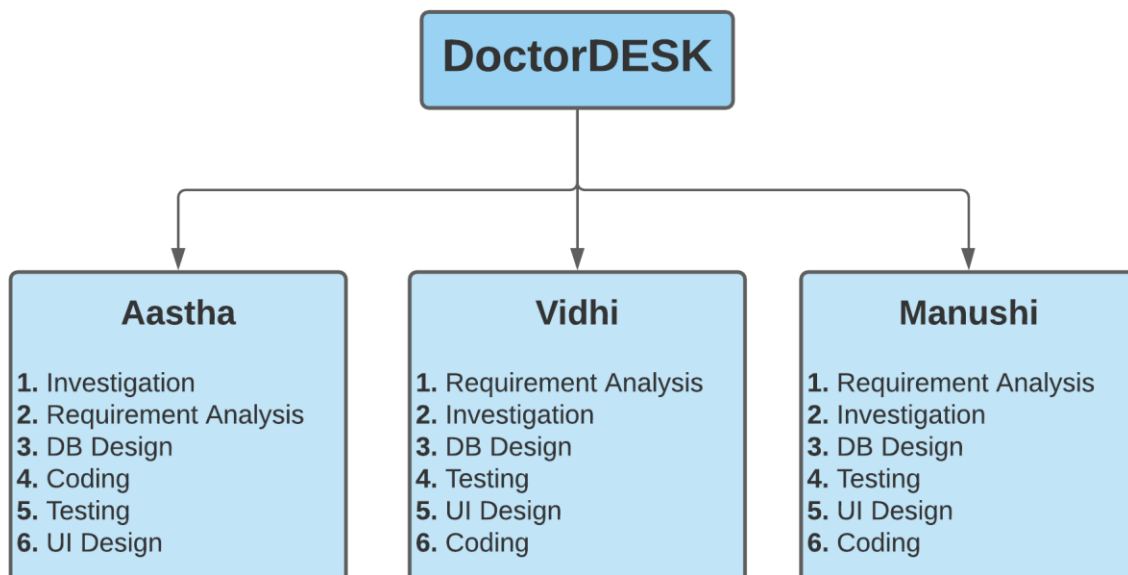
### 2.3.1 Milestone and Deliverables

- In this project, we got to know that the basic part is database designing so we started learning deeply about it.
- In this project, we went through Module Wise Completion. First, we did analysis of first module; we went through all the requirements for first module.
- By this analysis, we decided to gather the necessary features for this software according every user.

Software Process Activity	Milestones
Project Plan	Project Schedule
Requirement Collection	User requirements
Data Flow Analysis	System Flow
Design <ol style="list-style-type: none"> <li>1. User Interface Design</li> <li>2. System Design</li> <li>3. Database Design</li> </ol>	System Design Document
Implementation <ol style="list-style-type: none"> <li>1. Code For giving security</li> <li>2. Code for reports</li> </ol>	Access Reports
Testing	Setting validation and error message

[Table 2. 2 Milestones and Deliverables]

### 2.3.2 Roles and Responsibilities



[Figure 2. 3 Roles and Responsibilities]



### 2.3.3 Gantt Chart

ID	Task Name	Start	Finish	Duration	2020						2021			
					Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
1	Preliminary Investigation	15/02/2021	26/02/2021	1.4w										
2	Requirement Analysis	27/02/2021	28/02/2021	1 d										
3	Design	05/03/2021	25/03/2021	2.5 w										
4	Implementation	25/03/2021												
5	Testing													

[Table 2. 3 Gantt Chart]

### 2.3.4 Cost Estimation:

- Generally, the cost of making a web app ranges extremely: from total zero to unbelievably expensive price that could reach millions. Although, frankly there is no simple answer to this inquiry due to multiple factors at play. Different developer rates, project complexity and time it takes to build an app impact the cost of making a web application. The price to make an web app depends on the following aspects:

- type (game, business, social networking, lifestyle. etc.)
- design (basic, individual, custom)
- number of pages
- features & infrastructure

# **CHAPTER:3**

## **SYSTEM REQUIREMENT**

### **STUDY**

## CHAPTER: 3

### SYSTEM REQUIREMENT STUDY

#### 3.1 User Characteristics:

##### 3.1.1 Patient module:

In the Patient module, user will be able to update their Profile.

- User can search Doctor.
- User can view profile.
- User can book appointment.
- User can view consultancy fees.
- User can cancel appointment.
- User can have their prescription history.

##### 3.1.2 Doctor module:

In the Doctor module, user will be able to update their Profile.

- Doctor can view their Appointment according to time.
- Doctor can add Prescription.
- Doctor can have Medical History of Patient.
- Doctor can update their consultancy fees.
- Doctor can cancel any appointment if required.
- Doctor can add Patient if required.

##### 3.1.3 Administration module:

Following are the sub module in the Administration module.

- Admin can Register Doctor.
- Manage Doctor's Data.
- Manage Patient's Data

## **3.2 Hardware and Software requirement:**

### **3.2.1 Hardware Requirement:**

- User can use this website on Laptop, Mobile phone and Tablet.
- Any operating system.
- In laptop, minimum 4Gb RAM, in Mobile or Tablet minimum 2Gb RAM is preferred for good experience.
- Efficient Processor.
- Minimum 32Gb internal storage in all devices

### **3.2.2 Software Requirement:**

- Web Browser is required for use this website.

## **3.3 Specific Characteristic:**

### **3.3.1 Functionality:**

- Slot for appointment.

### **3.3.2 Usability:**

- Patients, Doctors, Admin.
- Doctor can know the medical history of any Patient.
- Any Patient would know about the local clinic Doctors easily through this software.

### **3.3.3 Reliability:**

- It can handle multiple entries at a time.
- Can be access from anywhere.
- Good validations of user inputs will be done to avoid incorrect storage of records.

### **3.3.4 Performance:**

- The website performance good into network.
- The capability of the computer depends on the performance of the software. The software can take any number of inputs provided the database size is larger

enough. This would depend on the available memory space

### **3.3.5 Supportability:**

- It supports all operating system; all we need are Web Browser and Internet Connection.

### **3.4 Design Constrains:**

- The constraints at the designing time are that the needs of the end users may keep on changing so the designers must keep this in view and design the product in the way that it is easily updatable.
- The operating system may not support older hardware like keyboard and mouse's so the designers must keep update with the entire hardware requirement so that it can work on the update of software

### **3.5 Assumption and Dependencies:**

- All the data entered will be correct and Up To Date.
- The software depends on Front-end and Back-end.
- The product must have interface which is simple to understand.

# **CHAPTER: 4**

## **SYSTEM ANALYSIS**

## **CHAPTER: 4**

### **SYSTEM ANALYSIS**

#### **4.1 Study of Current System**

##### **4.1.1 Existing System:**

There are already existing same applications like this but there are some problems in them. Some of existing systems are as follow:

- No existing system for interacting local clinic to the Patients.
- Lack of services.
- Lack of Security.

#### **4.2 Requirement of this System**

- In this project, we are going to develop the php based software, which involves the multi clinics management.
- Currently, this type of concept is not available in market. We are developing our software on this creative concept.

##### **4.2.1 Functional Requirement:**

###### **User Requirement:**

- From the user's perspective, our system fully operational feasible as it just requires some knowledge of computer.

###### **Identification of functional requirement:**

- The high level functional requirement often needs to be identified from an informal problem description document or from a conceptual understanding of the problem.
- Each high-level requirement characterizes away of system usage by some users to perform some meaningful piece of work.

###### **Documentation of functional requirement:**

- For documenting the functional requirement, we need to specify the set of functionalities supported by the system.

**4.2.2 Non-functional requirement:****Usability:**

- The interface should use terms and concepts, which are drawn from the experience of people who will make most of the system.

**Efficiency:**

- The software must provide easy and fast access without consuming more time for booking appointment and finding right Doctor.

**Readability:**

- User should never be surprised by the behaviour of the system and it should also provide meaningful feedback when error occurs so that user can recover from the error.

**Accuracy:**

- The user should require that data are obtained from database and stored in database must be accurate.

**Security:**

- The user wants the data stored in database must be secured and cannot be accessed by unauthorized user.

**Maintainability:**

- User wants that the system should be maintained easily means that if there are some changes required in the system that can be done easily.

**4.3 Feasibility Study**

- Feasibility is the measure of how beneficial the development of information system will be to an organization.
- The feasibility analysis is categorized under four different types.
  1. Operational Feasibility
  2. Technical Feasibility
  3. Schedule Feasibility
  4. Economic Feasibility



**1. Operational Feasibility:**

- The System is to be developed for any user who wants to use it. We want our system user friendly and easy to use.
- The user also may be non-technical, so the user interface will be designed in such a way that it gets comfortable for non-technical person to operate easily.

**2. Technical Feasibility:**

- It is a partially measurement of specific technical solution and the availability of technical resorts and expertise.
- The analyst must find out whether the current technical resources, which are available in the system is capable of handling the job.
- If not, then the analyst with the help of developer should confirm whether the technology is available and capable or not.

**Better Considering:**

- Here we have to consider those tools which are required for developing the project.
- As far as basic knowledge concerned we have studied basic of PHP and MySQL.

**3. Schedule Feasibility:**

- Schedule feasibility corresponds to whether sufficient time is available to complete the project.

**Factor considered:**

- Schedule of the project
- Time by which project has to be completed
- Reporting period

**4. Economic Feasibility:**

- Economic feasibility is a measure of cost effectiveness of a project or solution.
- For declaring that the system is economically feasible, the benefits from the project should exceed or at least to the equal to the cost of development.

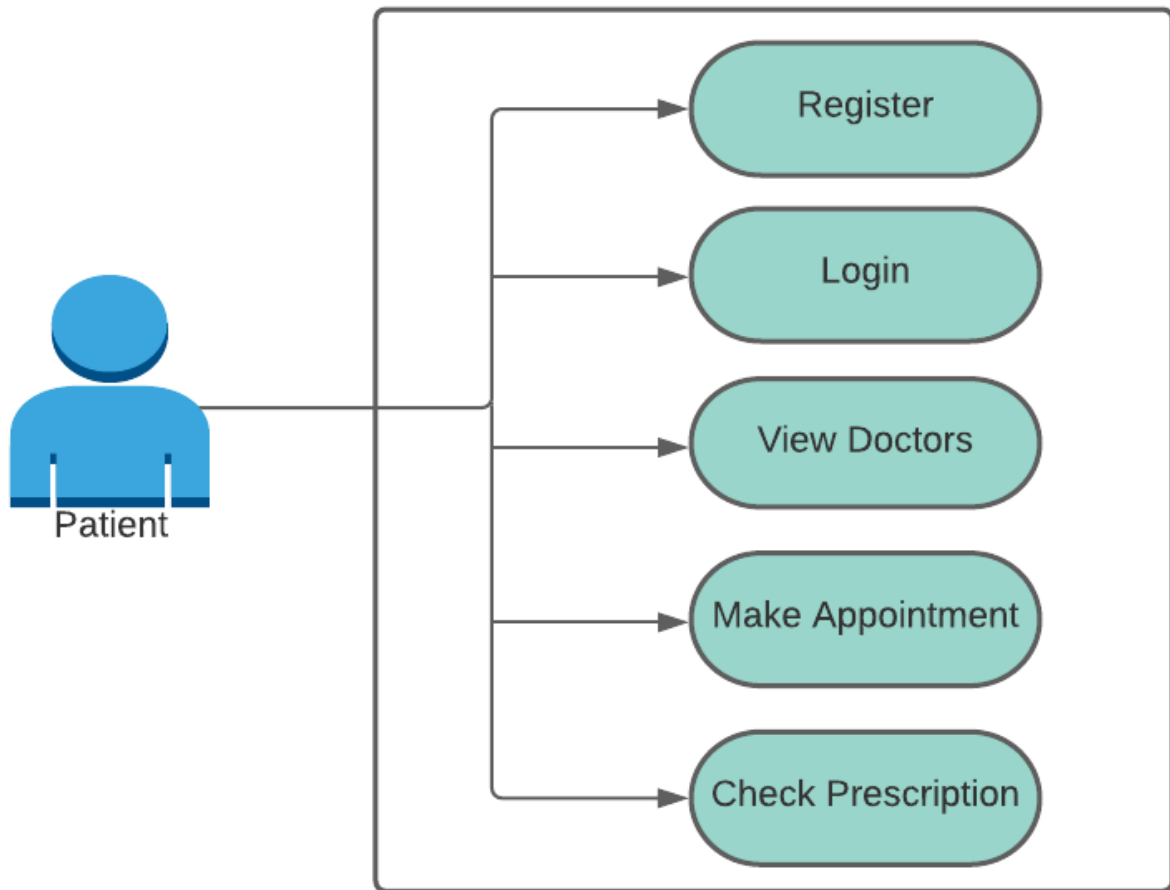
## 4.4 Requirement Validation:

- Requirement validation examines this specification to ensure that all the system requirements have been stated unambiguously.
- These inconsistent, error have been detected and corrected and the work products confirmed to the standard.
- Source of the requirement are identified; final Statement of requirement has been examined by original source.
- Requirements related to main requirements are founds.
- Requirements are clarifying stated and are not misinterpreted.
- All sources of requirements are covered to get a maximum requirement.
- All method of finding requirements is applied.

## 4.5 Function of the System:

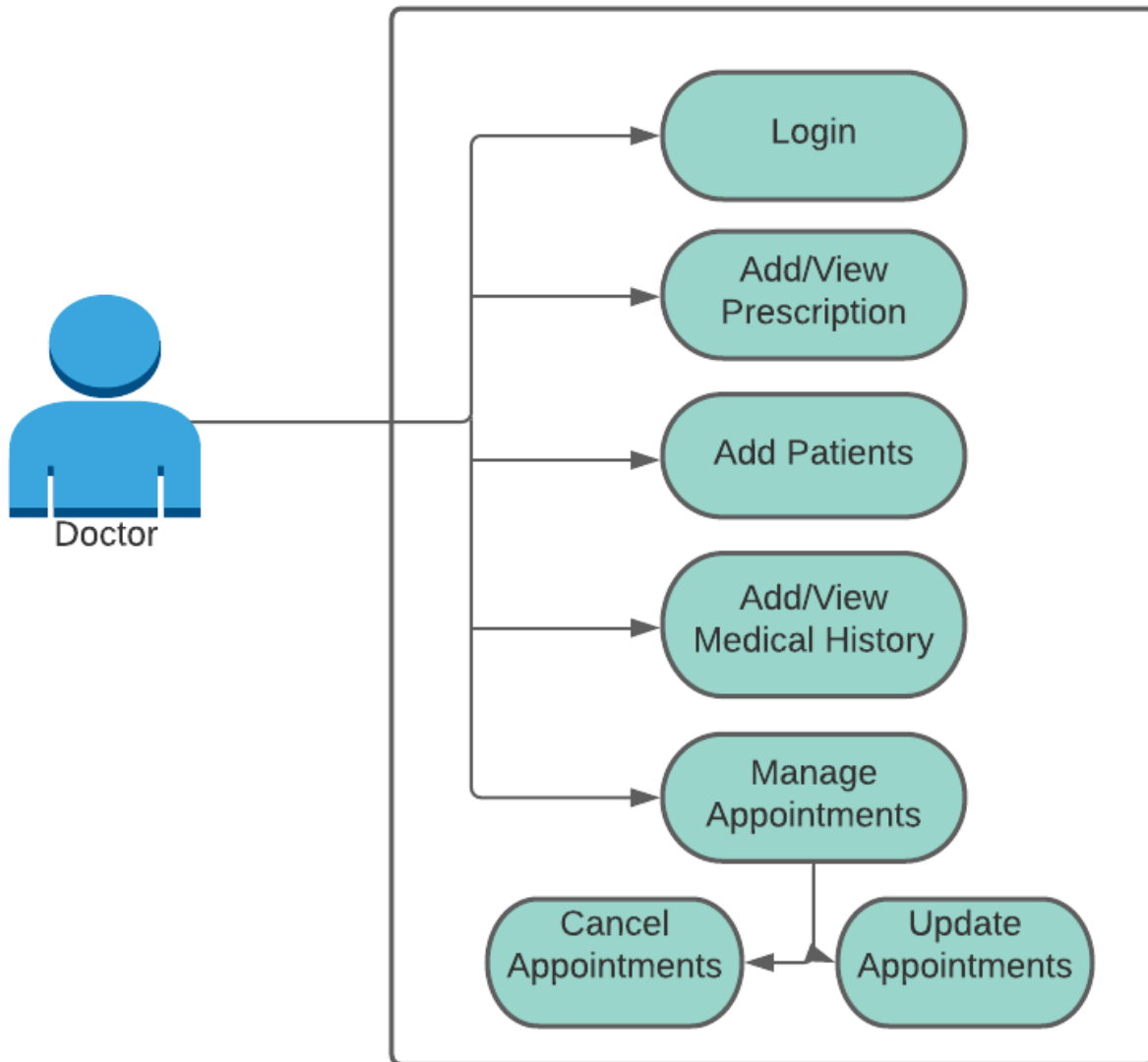
### 4.5.1 Use-Case Diagram:

- In software and systems engineering, a **use case** is a list of steps, typically defining interactions between actor and a system, to achieve a goal.
- The actor can be a human, an external system, or time.
- In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals.
- The detailed requirements may then be captured in Systems Modelling Language or as contractual statements.
- As an important requirement technique, use cases have been widely used in modern software engineering over the last two decades.
- Use case driven development is a key characteristic of process models and frameworks.
- With its iterative and evolutionary nature, use case is also a good fit for agile development.

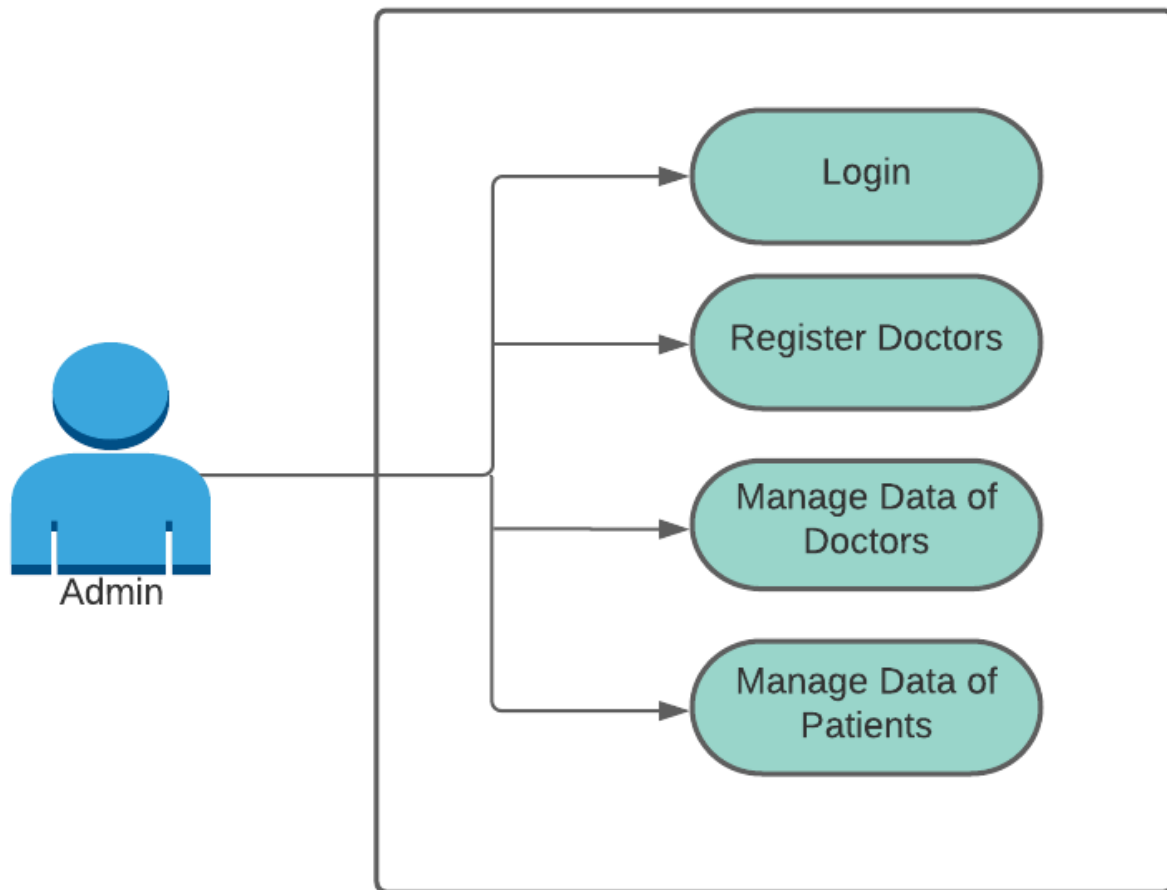
**4.5.1.1. Use-Case Diagram - Patient:**

[Figure 4. 1 Use-case Diagram - Patient]

## 4.5.1.2. Use-Case Diagram - Doctor:



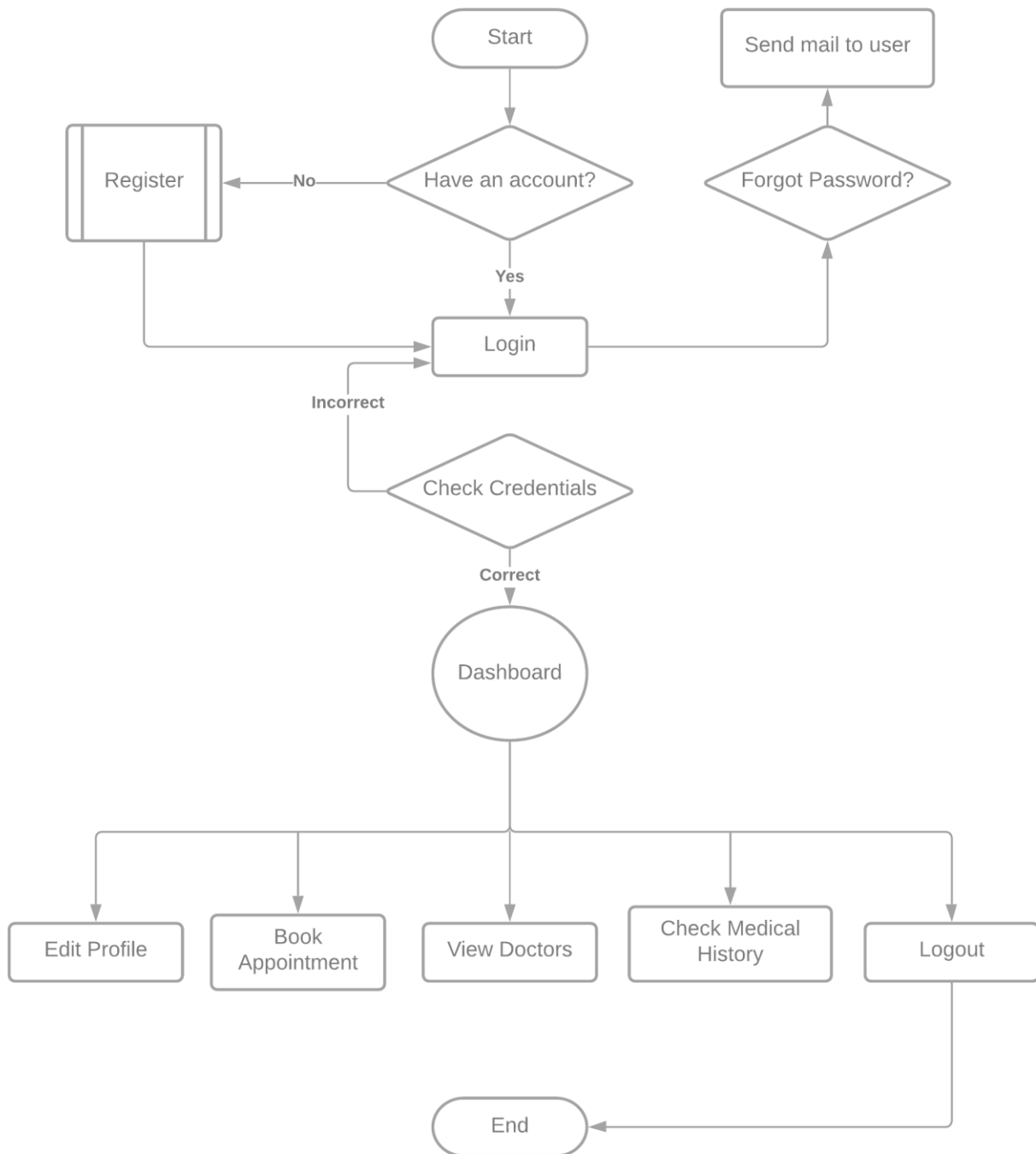
[Figure 4. 2 Use-case Diagram - Doctor]

**4.5.1.3. Use-Case Diagram - Admin:**

[Figure 4. 3 Use-case Diagram - Admin]

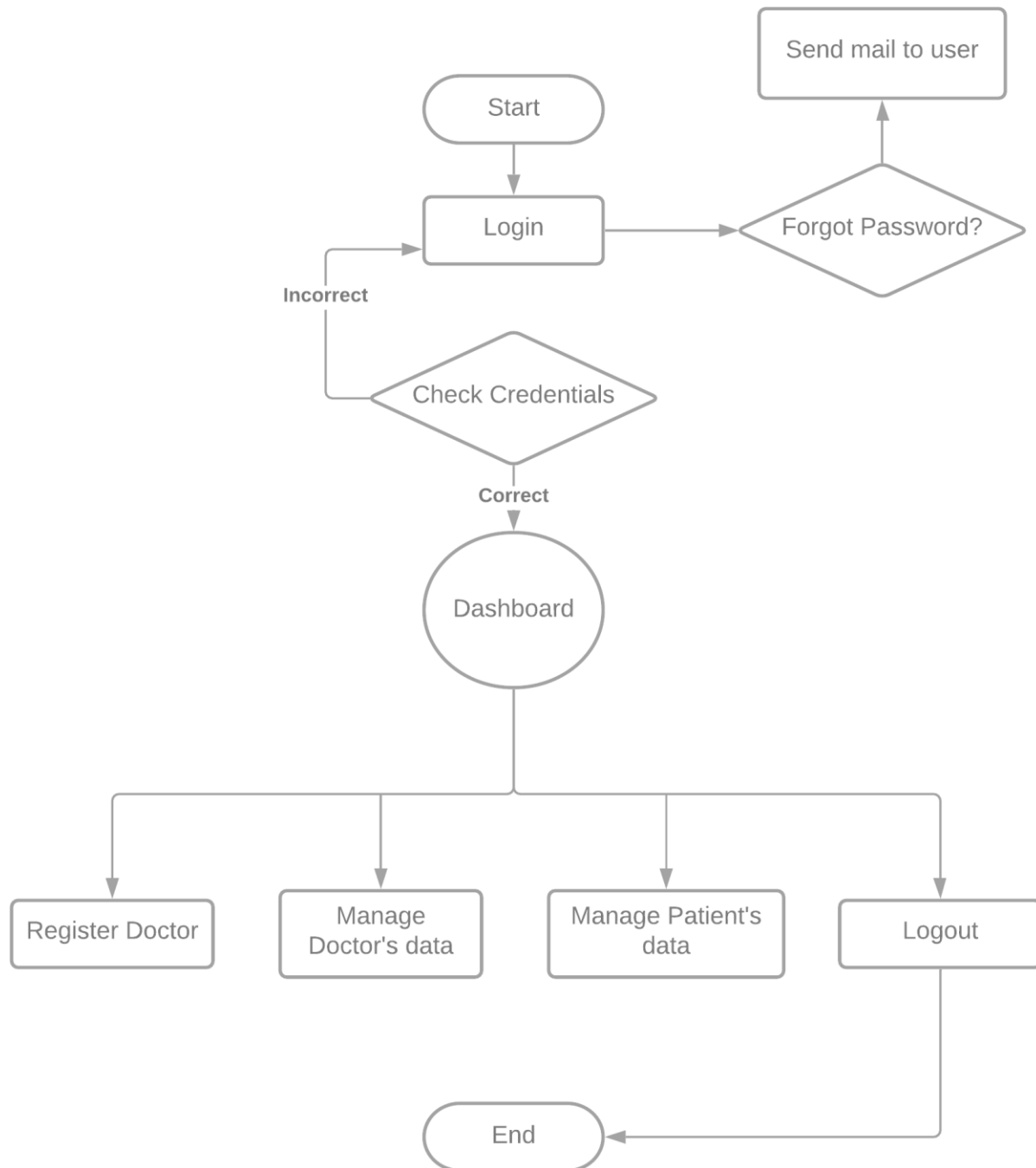
## 4.5.2 Activity Diagram:

### 4.5.2.1 Activity Diagram - Patient:



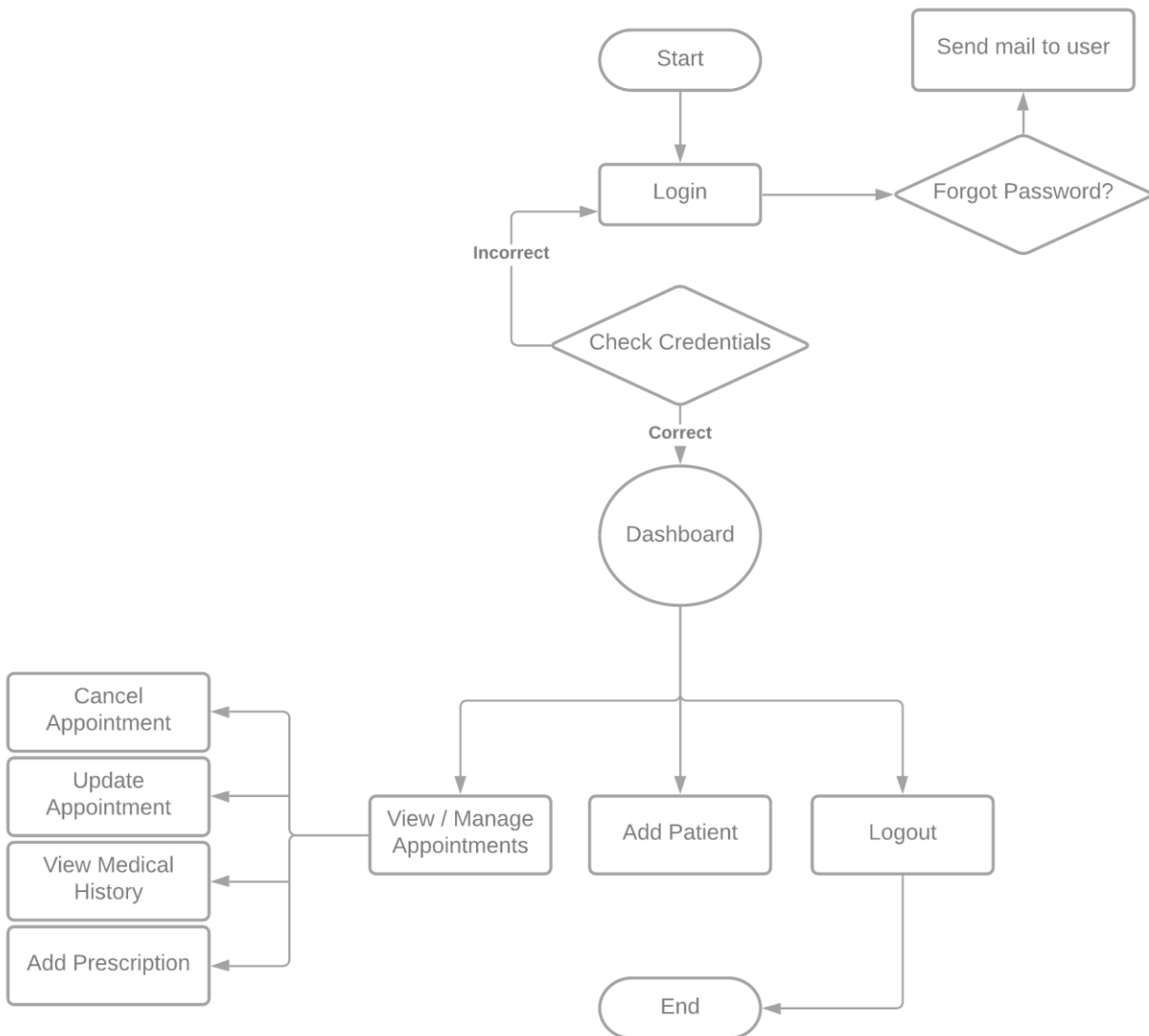
[Figure 4. 4 Activity Diagram-Patient]

## 4.5.2.2 Activity Diagram - Admin:



[Figure 4. 5 Activity Diagram – Admin]

## 4.5.2.2 Activity Diagram - Doctor:



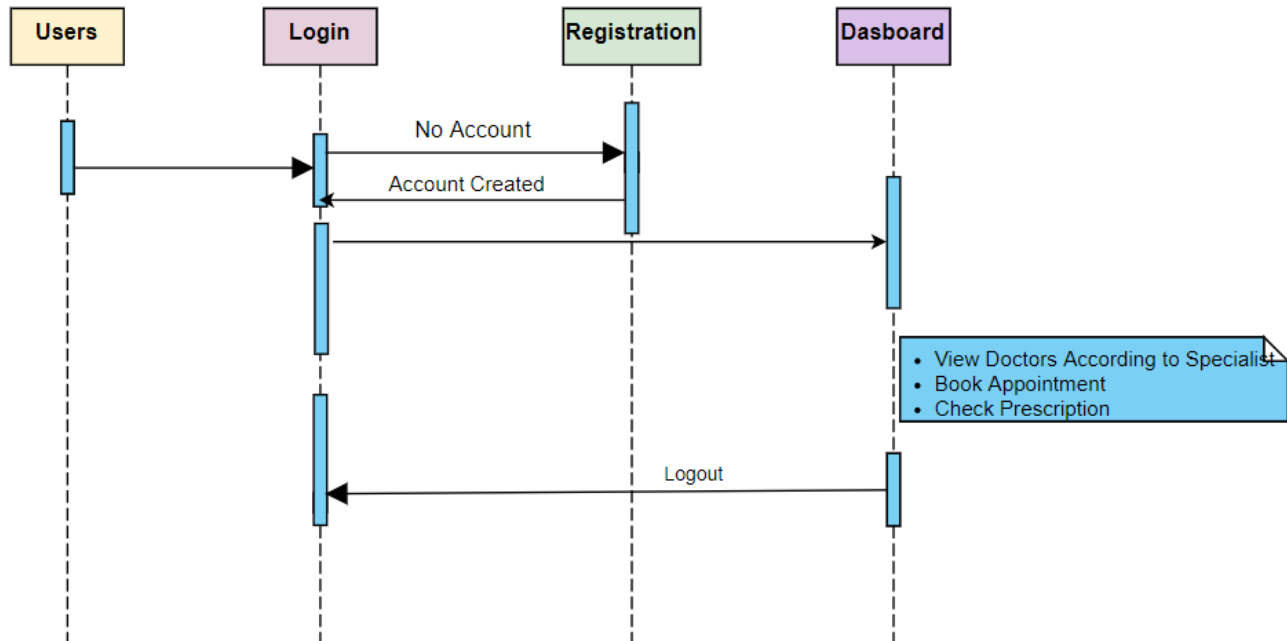
[Figure 4. 6 Activity Diagram - Doctor]



### 4.5.3 Sequence Diagram:

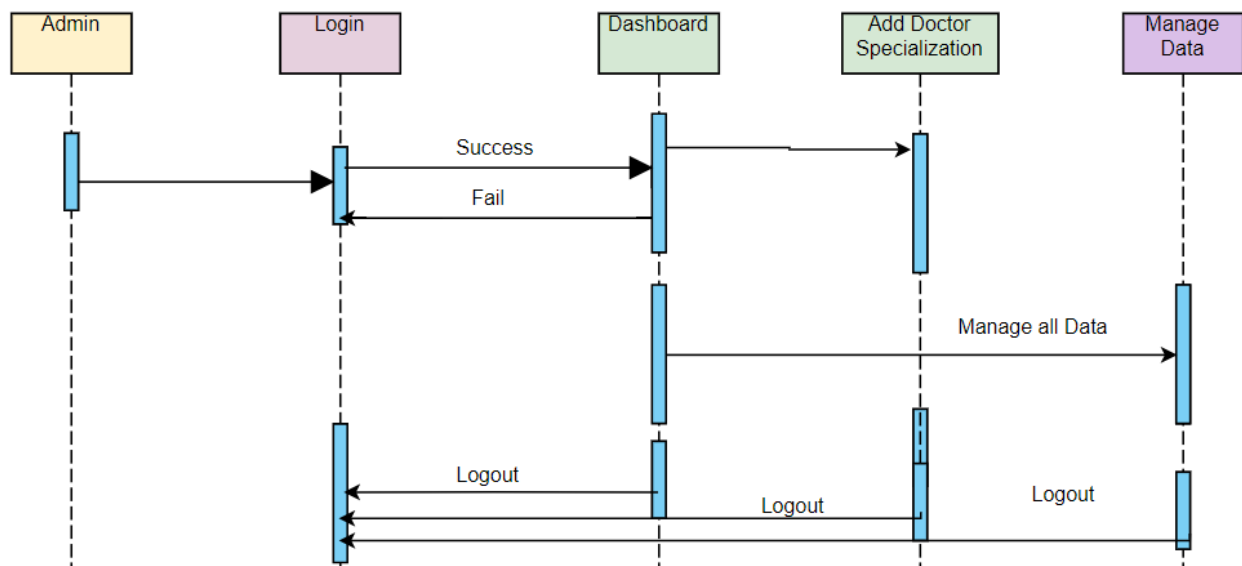
- The well-known Message Sequence Chart technique has been incorporated into the Unified Modelling Language (UML) diagram under the name of **Sequence Diagram**.
- A sequence diagram shows, as parallel vertical lines, different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur.
- This allows the specification of simple runtime scenarios in a graphical manner.
- The well-known Message Sequence Chart technique has been incorporated into the Unified Modelling Language (UML) diagram under the name of **Sequence Diagram**. A sequence diagram shows, as parallel vertical lines, different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

## 4.5.3.1 Sequence Diagram - User:



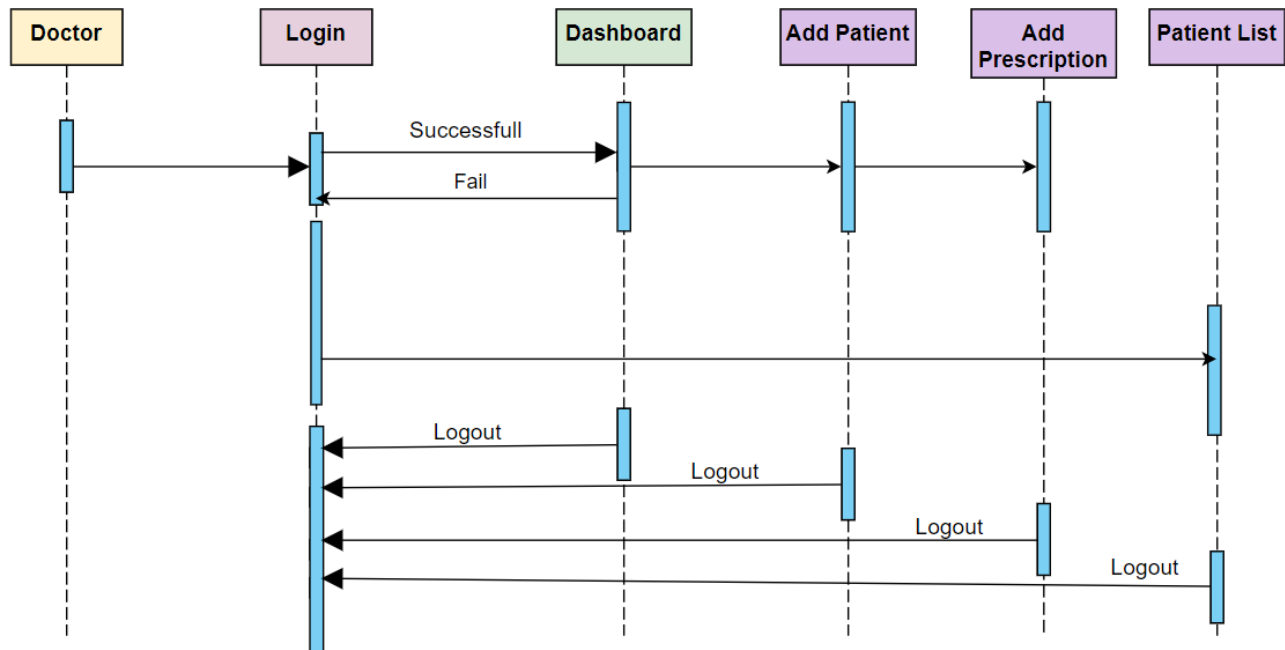
[Figure 4. 7 Sequence Diagram - User]

## 4.5.3.2 Sequence Diagram - Admin:



[Figure 4. 8 Sequence Diagram - Admin]

### 4.5.3.3 Sequence Diagram - Doctor:



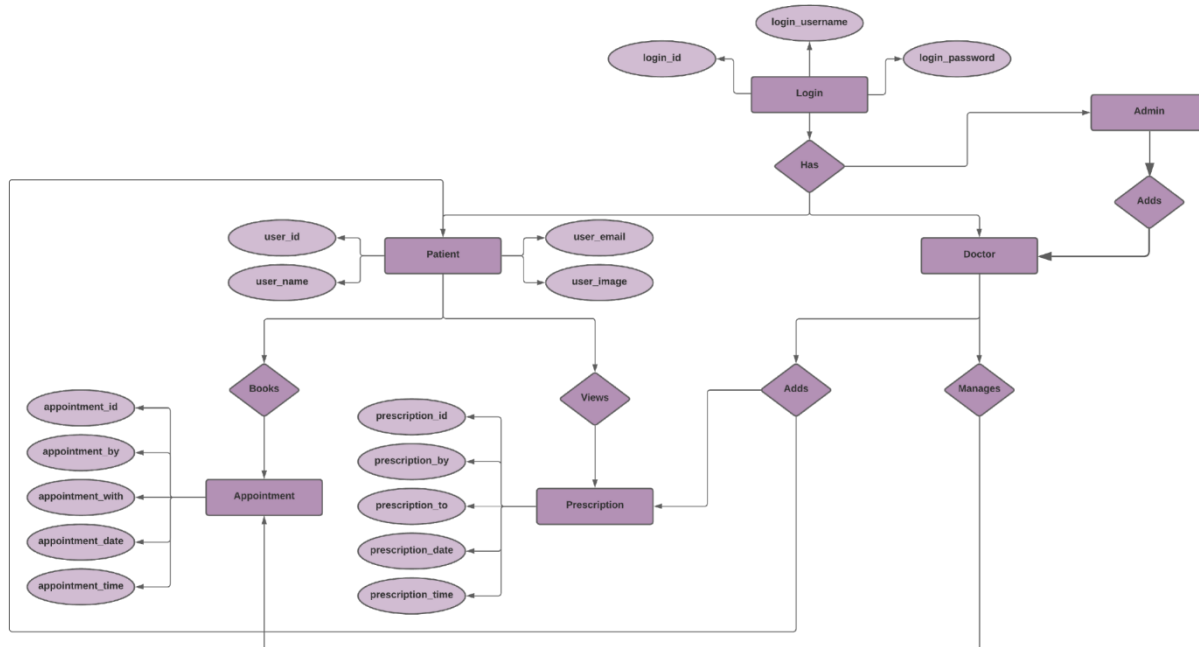
[Figure 4. 9 Sequence Diagram - Doctor]

## 4.6 Data Modelling:

### 4.6.1 E-R Diagram:

- In software engineering, an **entity–relationship model (ER model)** is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database.
- The main components of ER models are entities and the relationships that can exist among them, and databases.
- An entity-relationship model is a systematic way of describing and defining a business process.
- The process is modelled as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterize them.

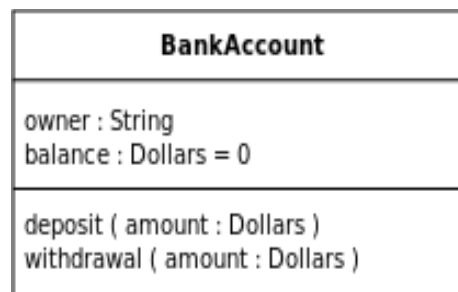
- Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.



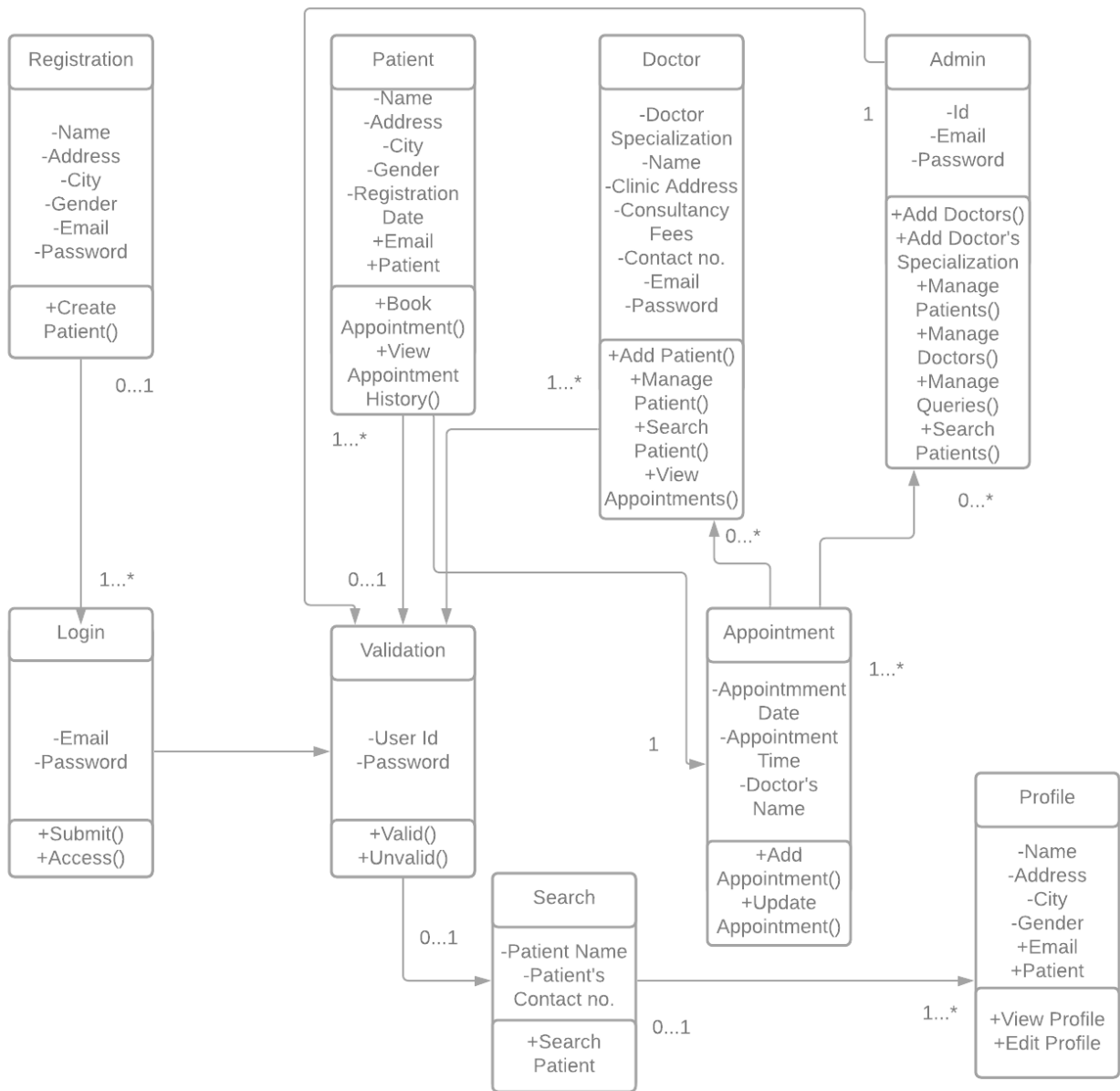
[Figure 4.10 E-R Diagram]

### 4.6.2 Class Diagram:

- In software engineering, a **class diagram** in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.
- The class diagram is the main building block of object oriented modelling.
- It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code.
- Class diagrams can also be used for data modelling.
- The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.



- In the diagram, classes are represented with boxes which contain three parts:
- The top part contains the name of the class. It is printed in Bold, centred and the first letter capitalized.
- The middle part contains the attributes of the class. They are left aligned and the first letter is lower case.
- The bottom part gives the methods or operations the class can take or undertake. They are also left aligned and the first letter is lower case.



[Figure 4.11 Class Diagram]

# **CHAPTER: 5**

## **SYSTEM DESIGN**

---

## CHAPTER: 5

### SYSTEM DESIGN

#### 5.1 Database Design:

We are going to use server-side database PHP-MySQL which is commonly used in Web application.

What is MySQL?

- MySQL is a database system used on the web.
- MySQL is a database system that runs on a server.
- MySQL is ideal for both small and large applications.
- MySQL is developed, distributed, and supported by Oracle Corporation.
- With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.
- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform).

#### 5.2 Data Dictionary:

##### 5.2.1 Admin Table:

Field Name	Data Type(With size)	Constraint	Description
id	Int(11)	Primary Key	AUTO_INCREMENT
username	Varchar(255)	Not Null	
Password	Varchar(255)	Not Null	
updateDate	Varchar(255)	Not Null	

[Table 5.1- Admin Table]

##### 5.2.2 Appointment Table:

Field Name	Data Type(With size)	Constraint	Description
Id	int(11)	Primary Key	AUTO_INCREMENT
docterSpecialization	Varchar(255)	Not Null	
docterId	int(11)	Not Null	
userId	int(11)	Not Null	
consultancyFees	int(11)	Not Null	



appointmentDate	Varchar(255)	Not Null	
appointmentTime	Varchar(255)	Not Null	
postDate	timestamp	Not Null	
userStatus	int(11)	Not Null	
docterStatus	int(11)	Not Null	
updationDate	timestamp	Not Null	ON UPDATE CURRENT_TIMESTAMP()

[Table 5.2- Appointment Table]

### 5.2.3 Doctors Table:

Field Name	Data Type(With size)	Constraint	Description
Id	int(11)	Primary Key	AUTO_INCREMENT
specialization	Varchar(255)	Not Null	
docterName	Varchar(255)	Not Null	
address	longtext	Not Null	
docFees	Varchar(255)	Not Null	
contactno	bigint(11)	Not Null	
docEmail	Varchar(255)	Not Null	
password	Varchar(255)	Not Null	
creationDate	timestamp	Not Null	
updationDate	timestamp	Not Null	

[Table 5.3- Doctor Table]

### 5.2.4 Doctors login Table:

Field Name	Data Type(With size)	Constraint	Description
Id	int(11)	Primary Key	AUTO_INCREMENT
uid	int(11)	Not Null	
username	Varchar(255)	Not Null	
userip	binary(16)	Not Null	
login Time	timestamp	Not Null	
logout	Varchar(255)	Not Null	
status	int(11)	Not Null	

[Table 5.4- Doctor Login Table]

### 5.2.5 Doctors Specialization Table:

Field Name	Data Type(With size)	Constraint	Description
Id	int(11)	Primary Key	AUTO_INCREMENT

specialization	Varchar(255)	Not Null	
creationDate	timestamp	Not Null	
updatetime	timestamp	Not Null	ON UPDATE CURRENT_TIMESTAMP()

[Table 5.5- Doctor Specialization Table]

**5.2.6 Contact Us Table:**

Field Name	Data Type(With size)	Constraint	Description
Id	int(11)	Primary Key	AUTO_INCREMENT
fullname	Varchar(255)	Not Null	
email	Varchar(255)	Not Null	
contactno	bigint(12)	Not Null	
message	mediumtext	Not Null	
PostingDate	timestamp	Not Null	
AdminRemark	mediumtext	Not Null	
LastupdateDate	Timestamp	Not Null	
IsRead	int(11)	Not Null	

[Table 5.6- Contact Us Table]

**5.2.7 Medical History Table:**

Field Name	Data Type(With size)	Constraint	Description
ID	int(11)	Primary Key	AUTO_INCREMENT
PatientID	int(11)	Not Null	
BloodPressure	Varchar(255)	Not Null	
BloodSugar	Varchar(255)	Not Null	
Weight	Varchar(255)	Not Null	
Temperature	Varchar(255)	Not Null	
MedicalPres	mediumtext	Not Null	
CreationDate	timestamp	Not Null	ON UPDATE CURRENT_TIMESTAMP()

[Table 5.7- Medical History Table]

**5.2.8 Patient Table:**

Field Name	Data Type(With size)	Constraint	Description
ID	int(10)	Primary Key	AUTO_INCREMENT

Docid	int(10)	Not Null	
PatientName	Varchar(200)	Not Null	
PatientContno	bigint(10)	Not Null	
PatientEmail	Varchar(200)	Not Null	
PatientGender	Varchar(50)	Not Null	
PatientAdd	mediumtext	Not Null	
PatientAge	int(10)	Not Null	
PatientMedhis	mediumtext	Not Null	
CreationDate	timestamp	Not Null	
UpdationDate	timestamp	Not Null	ON UPDATE CURRENT_TIMESTAMP()

[Table 5.8- Patient Table]

### 5.2.9 Patient Login Table:

Field Name	Data Type(With size)	Constraint	Description
id	int(11)	Primary Key	AUTO_INCREMENT
uid	int(11)	Not Null	
username	Varchar(255)	Not Null	
userip	binary(16)	Not Null	
login Time	timestamp	Not Null	
logout	Varchar(255)	Not Null	
status	int(11)	Not Null	

[Table 5.9- Patient Login Table]

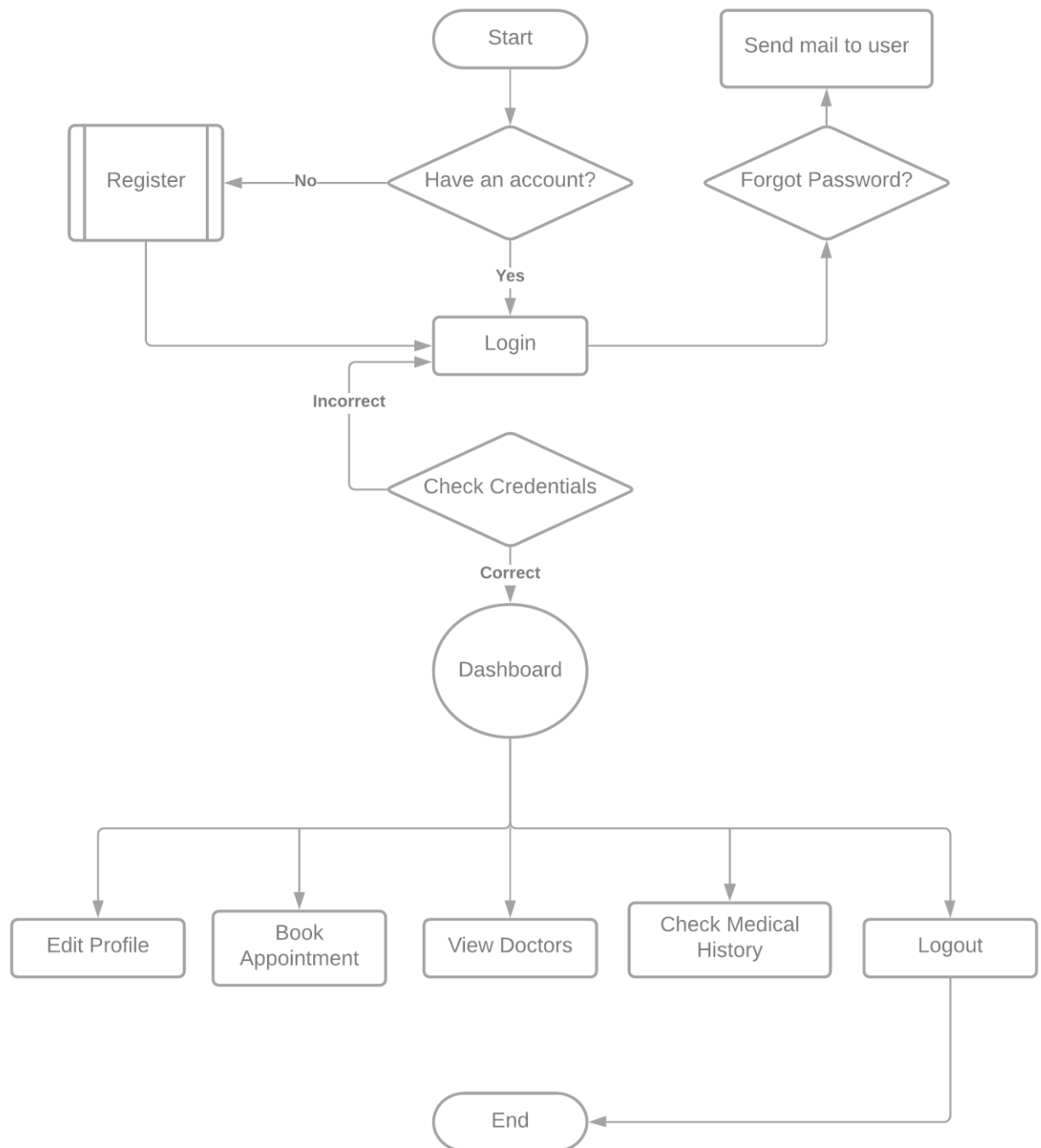
### 5.2.10 Patient Registration Table:

Field Name	Data Type(With size)	Constraint	Description
id	int(11)	Primary Key	AUTO_INCREMENT
fullName	Varchar(255)	Not Null	
address	longtext	Not Null	
city	Varchar(255)	Not Null	
gender	Varchar(255)	Not Null	
email	Varchar(255)	Null	
password	Varchar(255)	Not Null	
regDate	timestamp	Not Null	
updationDate	timestamp	Not Null	ON UPDATE CURRENT_TIMESTAMP()

[Table 5.10- Patient Registration Table]

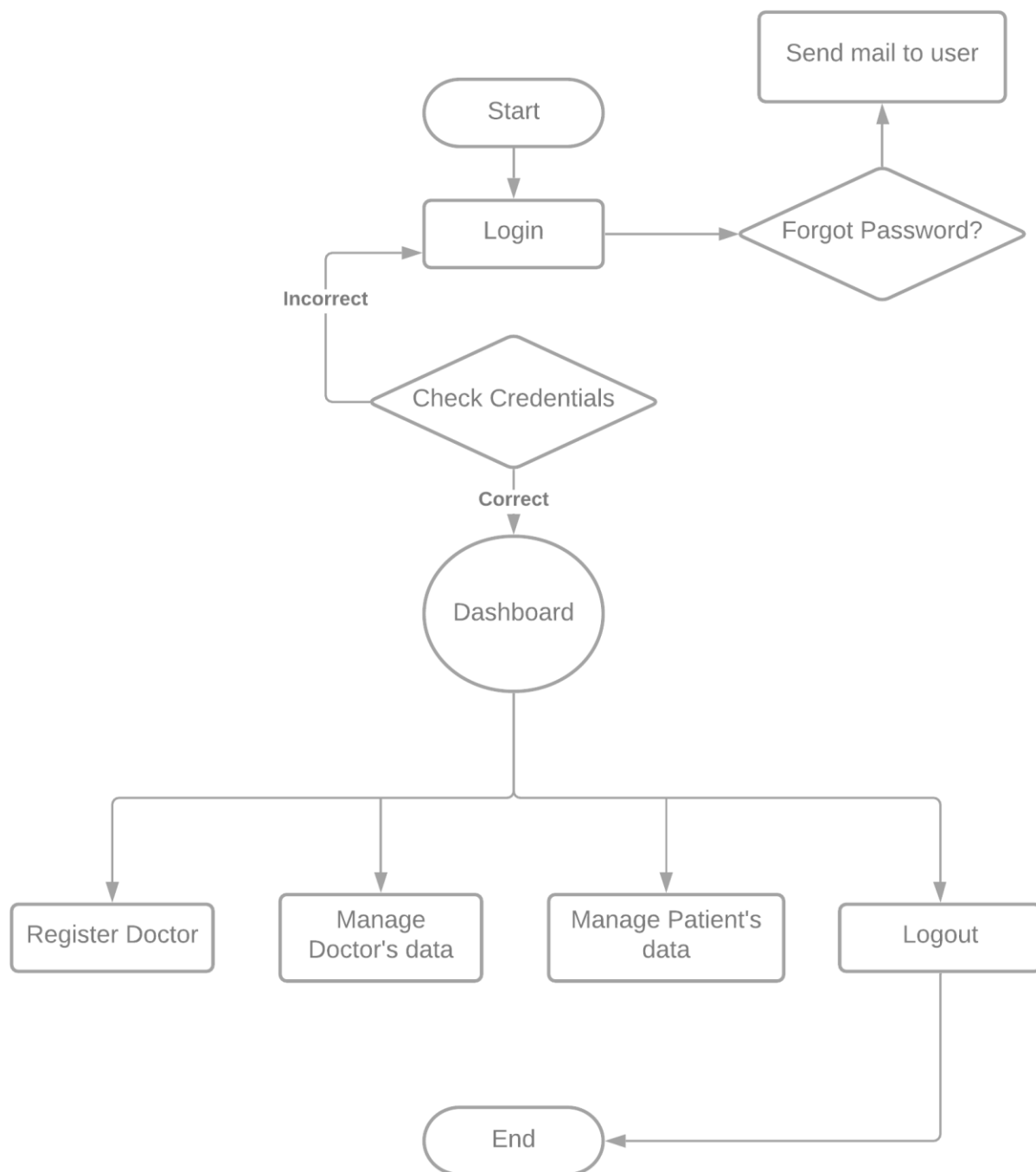
### 5.3 Flow Chart

#### 5.3.1 Patient:



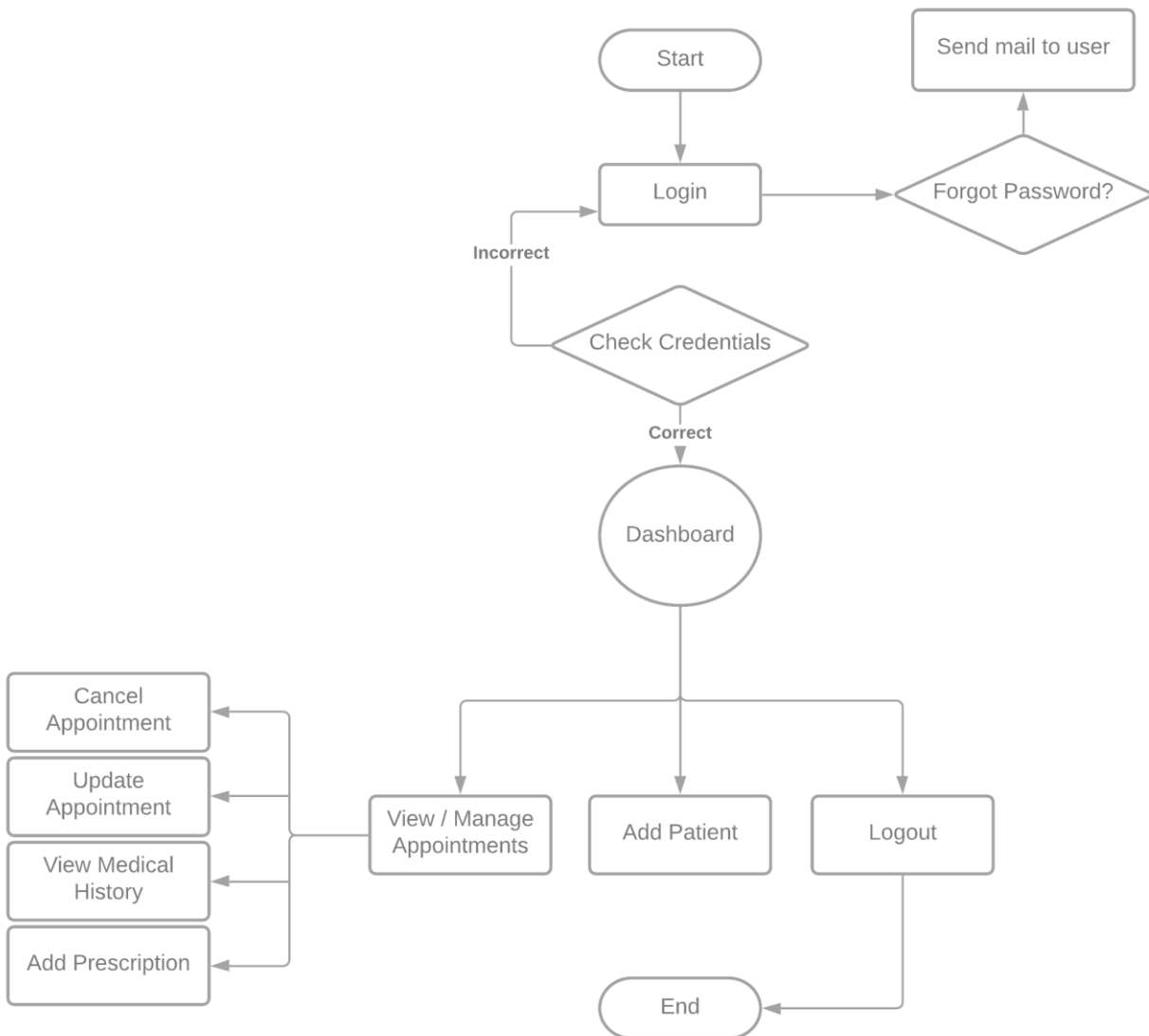
[Figure 5.1 Flow Chart - Patient]

## 5.3.2 Admin:



[Figure 5.2 Flow Chart - Admin]

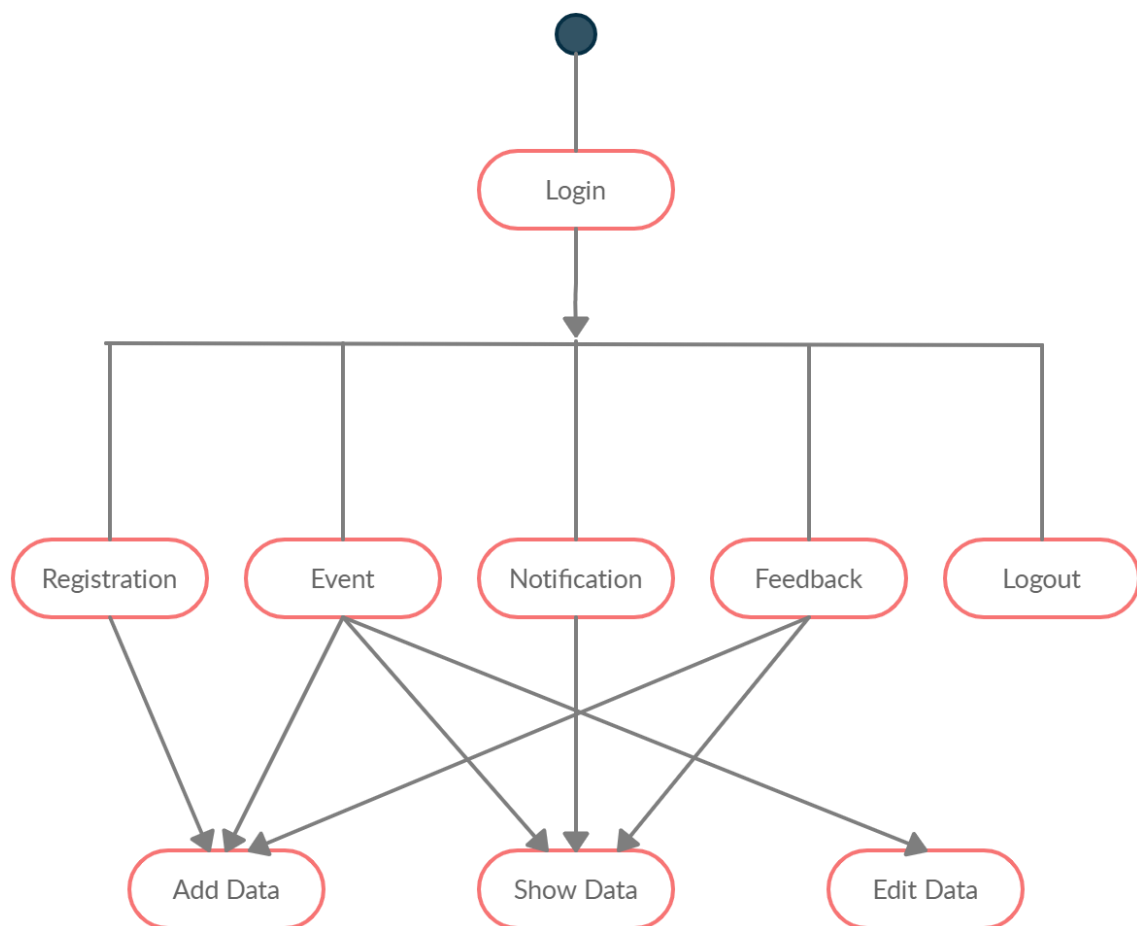
## 5.3.3 Doctor:



[Figure 5.3 Flow Chart - Doctor]

## 5.4 Input / Output and Interface Design:

### 5.4.1 State chart diagram



[Figure 5.4 State Chart Diagram]

**CHAPTER: 6**

**IMPLEMENTATION AND**

**PLANNING DETAILS**



**CHAPTER: 6****IMPLEMENTATION AND PLANNING DETAILS****6.1 Implementation Environment**

- Our project is suitable to all type of users like single and multi-users.
- Multi users are allowed to operate the application at the same time.
- We provide the interface which is user friendly.
- We have GUI (graphical user interface) by which all type of users can easily access the application.
- One user at a time and also multi users can access the application at the same time and use all the services.
- If we don't provide the GUI in the application then user won't like our application.
- For better performance and reliability, we have to include GUI in the application
- So, for the more security and performance we have to use the GUI.

**6.2 Security Features****User authentication:**

- Identification and authentication are used to establish a user's identity.
- Each user is required to log in to the system.

**Password Protection:**

- Every user who is to be allowed to access the application is given his own username and password and given his own access rights so that only authorized and authenticated users can access the project.

**Confidentiality:**

- We provide confidentiality to all the users.
- In that one user cannot access the data of the other users.
- For that we provide one key to each user to secure its data.

**Scalability:**

- We provide the scalable application to make sure that every user can access the application in a proper order.
- User likes that type of application which are in one particular order that user cannot wait for the usage of the services

**6.3 Coding Standards**

- Every company follows a different coding standard based on their best practices.
- Coding standard is required because there may be many developers working on different modules so if they will start inventing their own standards then source will become very unmanageable and it will become difficult to maintain that source code in future.
- Here are several reasons why to use coding specifications:
  - I. Your peer programmers have to understand the code you produce.
  - II. A coding standard acts as the blueprint for all the team to decipher the code.
  - III. Simplicity and clarity achieved by consistent coding saves you from common mistakes.
  - IV. If you revise your code after some time then it becomes easy to understand that code.
- There are few guidelines which can be followed while coding in C#.

**Coding Standards for Components:** It is recommended to write components name by its purpose. This approach improves the readability and maintainability of code.

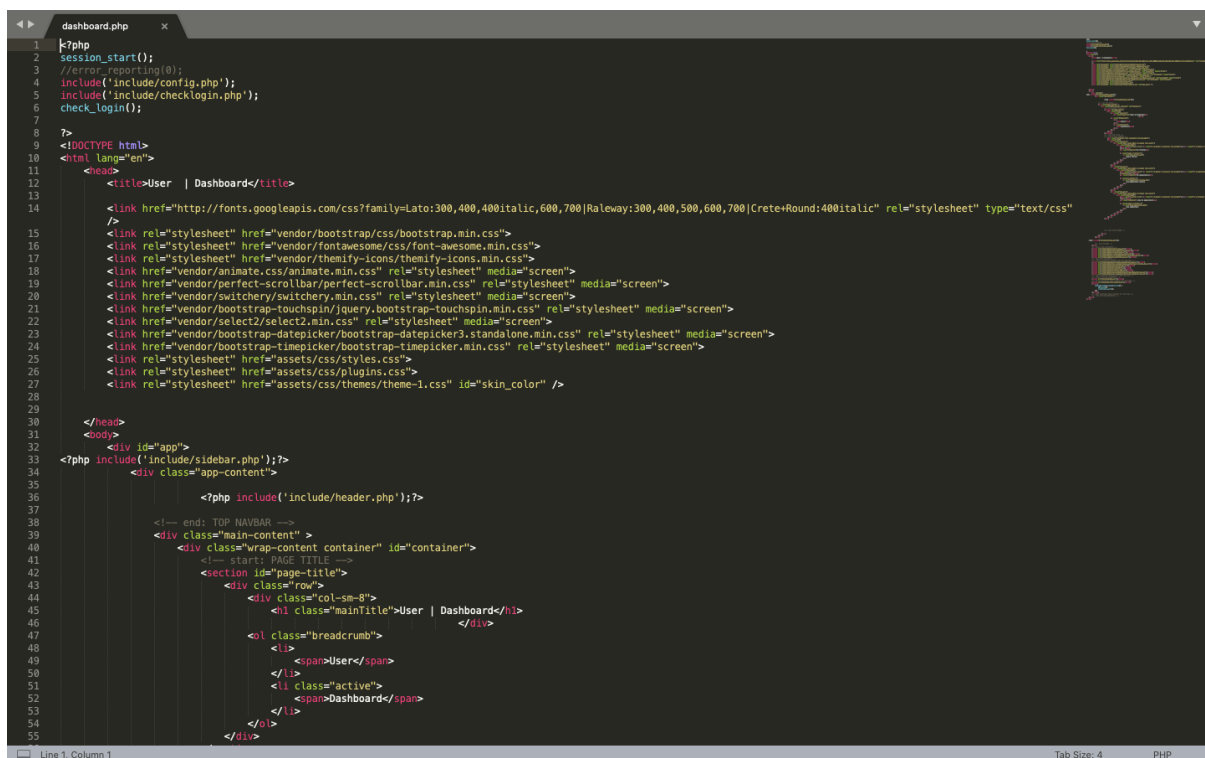
**Coding Standards for Classes:** Usually class name should be noun starting with uppercase letter. If it contains multiple words than every inner word should start with uppercase. Eg: String, String Buffer, and Class.

**Coding Standards for Interface:** Usually interface name should be adjective starting with uppercase letter. If it contains multiple word than every inner word should start with uppercase. Eg: Runnable, Serializable, Comparable

**Coding Standards for Methods:** Usually method name should either be verb or verb noun combination starting with upper letter. If it contains multiple word than every inner word should start with uppercase. Eg: Print(), Sleep(), SetSalary().

## 6.4 Sample Coding





```
1  <?php
2  session_start();
3  //error-reporting(0);
4  include('include/config.php');
5  include('include/checklogin.php');
6  check_login();
7
8  ?>
9  <!DOCTYPE html>
10 <html lang="en">
11 <head>
12 <title>User | Dashboard</title>
13
14 <link href="http://fonts.googleapis.com/css?family=Lato:300,400,400italic,600,700|Raleway:300,400,500,600,700|Crete+Round:400italic" rel="stylesheet" type="text/css" />
15 <link rel="stylesheet" href="vendor/bootstrap/css/bootstrap.min.css">
16 <link rel="stylesheet" href="vendor/fontawesome/css/font-awesome.min.css">
17 <link rel="stylesheet" href="vendor/themify-icons/themify-icons.min.css">
18 <link href="vendor/animate.css/animate.min.css" rel="stylesheet" media="screen">
19 <link href="vendor/perfect-scrollbar/perfect-scrollbar.min.css" rel="stylesheet" media="screen">
20 <link href="vendor/switchery/switchery.min.css" rel="stylesheet" media="screen">
21 <link href="vendor/bootstrap-touchspin/jquery.bootstrap-touchspin.min.css" rel="stylesheet" media="screen">
22 <link href="vendor/select2/select2.min.css" rel="stylesheet" media="screen">
23 <link href="vendor/bootstrap-datepicker/bootstrap-datepicker3.standalone.min.css" rel="stylesheet" media="screen">
24 <link href="vendor/bootstrap-timepicker/bootstrap-timepicker.min.css" rel="stylesheet" media="screen">
25 <link rel="stylesheet" href="assets/css/styles.css">
26 <link rel="stylesheet" href="assets/css/plugins.css">
27 <link rel="stylesheet" href="assets/css/themes/theme-1.css" id="skin_color" />
28
29 </head>
30 <body>
31 <div id="app">
32 <?php include('include/sidebar.php');?>
33 <div class="app-content">
34 <?php include('include/header.php');?>
35
36 <!-- end: TOP NAVBAR -->
37 <div class="main-content">
38 <div class="wrap-content container" id="container">
39 <!-- start: PAGE TITLE -->
40 <section id="page-title">
41 <div class="row">
42 <div class="col-sm-8">
43 <h1 class="mainTitle">User | Dashboard</h1>
44
45 <div class="breadcrumb">
46 <ul>
47 <li><span>User</span></li>
48 <li class="active"><span>Dashboard</span></li>
49 </ul>
50 </div>
51 </div>
52 </div>
53 </div>
54 </div>
55 </div>
```

[Figure 6.3 Coding Screenshot]

# **CHAPTER: 7**

## **TESTING**

## CHAPTER: 7

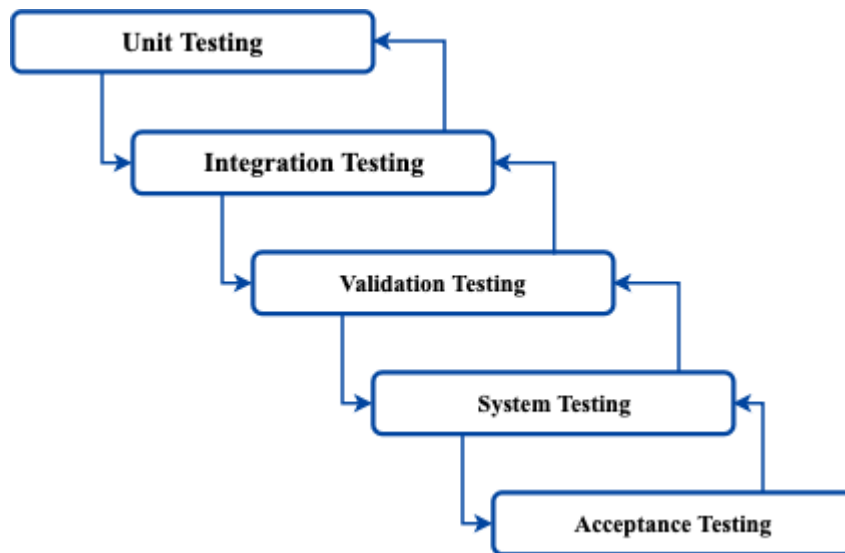
### TESTING

#### 7.1 Testing Plan

- A test plan is a systematic approach to testing a system such as a machine or software.
- Depending on the product and the responsibility of the organization to which the test plan applies, a test plan may include one or more of the following:
  - I. Design Verification or Compliance test-to be performed during the development or approval stages of the product, typically on a small sample of units.
  - II. Manufacturing or Production test-to be performed during preparation or assembly of the product in an ongoing manner for purposes of performance verification and quality control.
  - III. Acceptance or Commissioning test -to be performed at the time of delivery or installation of the product.
  - IV. Regression test -to be performed on an existing operational product, to verify that existing functionality didn't get broken when other aspects of the environment are changed (e.g., upgrading the platform on which an existing application runs)

#### 7.2 Testing Strategy

- The testing strategy followed by the company is unique in its own way.
- The developer first takes into account the UNIT Testing.
- Then the Integration testing is conducted to check the over functionality of the system.
- Then the Validation Testing is performed once the whole project is done. Alpha and Beta testing are done once by the testing team and the clients respectively.
- Then the over System testing is done and after that Acceptance testing is done.



[Fig 7.1 – Testing Strategy]

## 7.3 Testing Methods

- Software testing methods are traditionally divided into white and black-box testing.

### Unit Testing

- Black Box Testing** -Whether the particular class meets the requirements mentioned in the specification.
- White Box Testing** -The tester looks inside that class and checks if there is error in the code which is not found while testing the class as a black box.

### Integration Testing

- User Interface Testing** -Testing is done by moving through each and every menu item in the interface either in top-down manner or bottom-up manner.
- Interaction Testing** -When the system performs data processing, Interaction between various classes is tested.

### Validation Testing

- For Validation Testing stage, we have performed functional test cases and the results are compared in the form of actual and expected outcomes.
- The testing proved that the Validation was compliant with the requirements as specified in the Use Case and SRS (Software Requirement Specification).

- Integration of forms Designing, Login, Admin Management & Rights and Salary Management were tested and found to be successful.

### **System Testing**

- It is carried to see that functionality related sets of units used together function as designed.
- The system test specifications, incorrect operation of the system is narrowed down to incorrect operation of unit(s) and is taken care of by filing the units.
- Test data covers the possible values of each parameter based on the requirements.

### **Acceptance Testing**

- After each module completion, the system tester tested the system to check user acceptance and changes are made accordingly as per requirements.

## **7.4 Test Cases**

- Exhaustive testing of almost any non-trivial system is impractical due to the fact that domain of input values to most practical software systems is either extremely large or infinite.
- Therefore we must design an optimal test suite that is of reasonable size and can uncover as many errors in system as possible.
- The test cases to consider in the project are :-
  - 1) Separate authentication for both the front end as well as back end.
  - 2) Inclusion of all eligible data and modules to be tested.
  - 3) Testing individual module according to requirement.
  - 4) Privacy to the admin as well as the user who becomes the part of System.
  - 5) Updating of the information from time to time.

### **Purpose**

- The purpose of the test cases is to test the various input and see whether the output produces any error or not. There are different test cases according to the system. A Correct system must accomplish the following:
  1. Compute correct results.
  2. Operate safely, and cause the system containing the software to operate safely.
  3. Perform the tasks required by the system containing the software, as explained



in the software applications.

4. Achieve these goals for all inputs

**CHAPTER: 8**

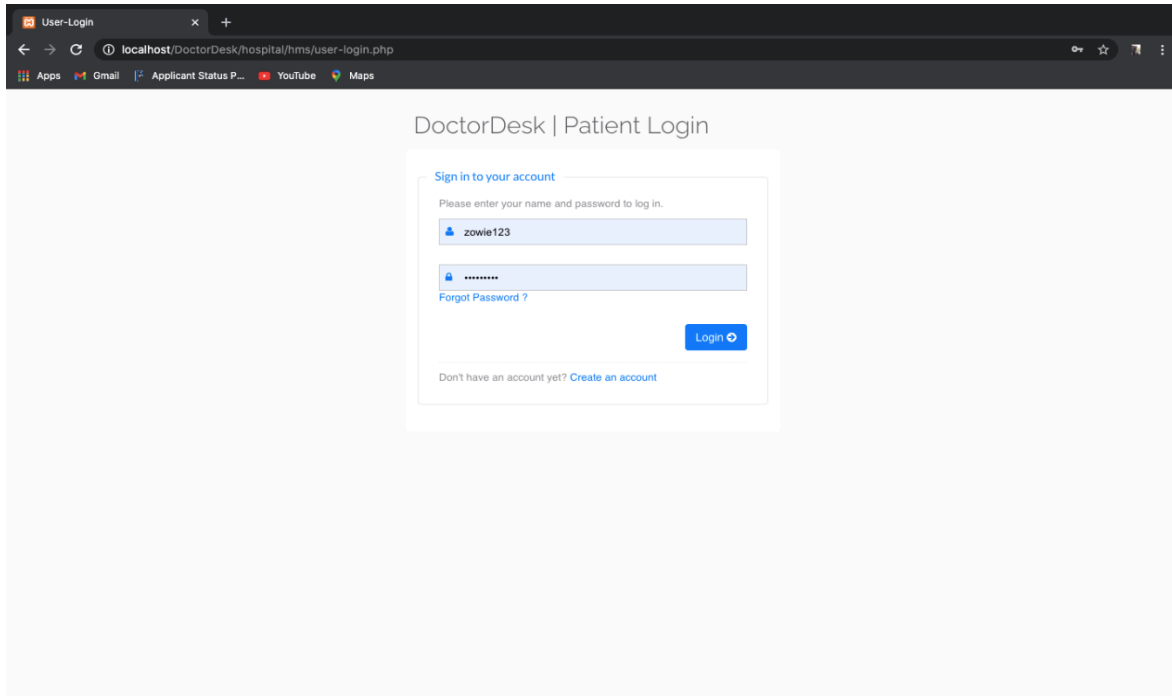
**SCREEN SHOT**

**AND**

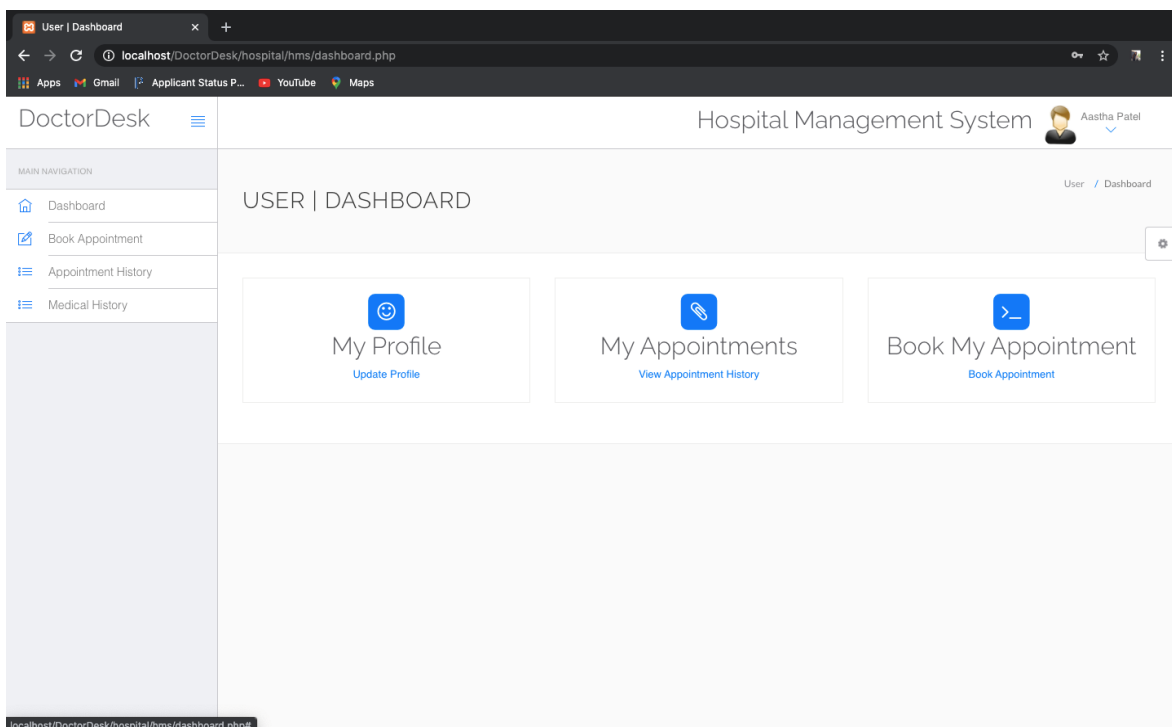
**USER MANUAL**

## CHAPTER: 8

### SCREENSHOT AND USER MANUAL



[Figure 8.1 User Login Screenshot]



[Figure 8.2 User Dashboard Screenshot]

DoctorDesk Hospital Management System Aastha Patel

MAIN NAVIGATION

- Dashboard
- Book Appointment
- Appointment History
- Medical History

USER | BOOK APPOINTMENT

Book Appointment

Doctor Specialization  
Dermatologist

Doctors  
Khushi Patel

Consultancy Fees  
5000

Date  
2021-04-02

Time  
12:15 AM  
eg : 10:00 PM

Submit

[Figure 8.3 User Book Appointment Screenshot]

DoctorDesk Hospital Management System Aastha Patel

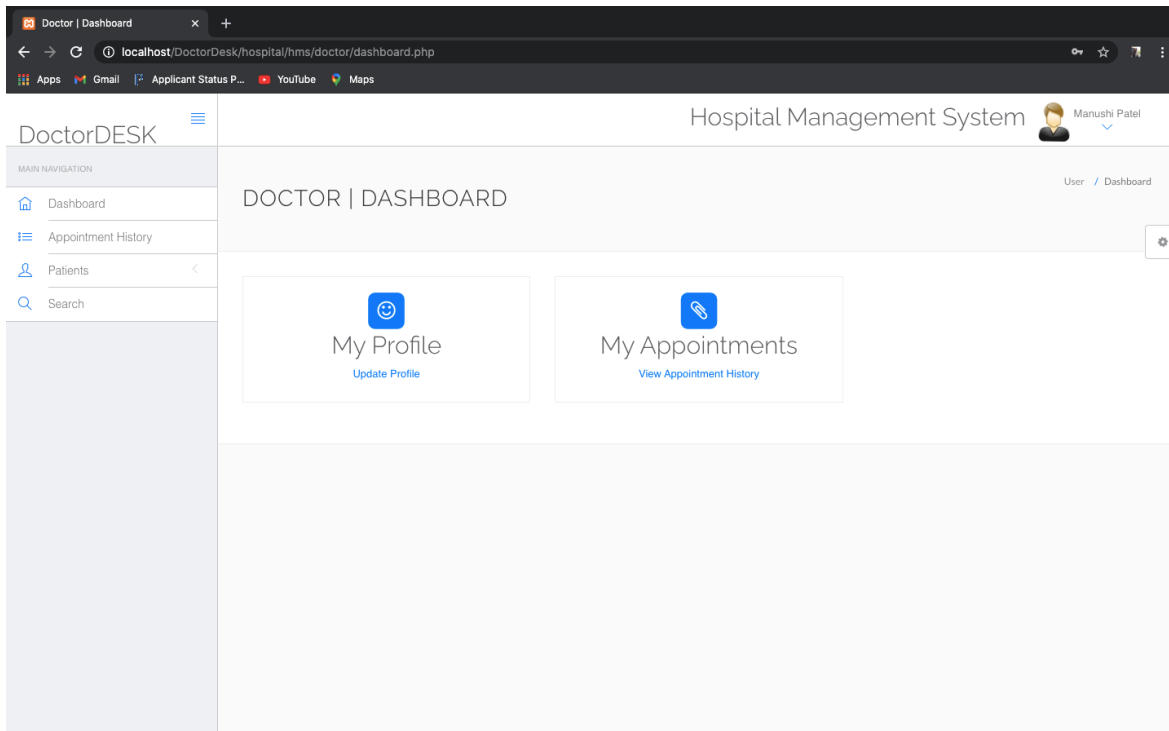
MAIN NAVIGATION

- Dashboard
- Book Appointment
- Appointment History
- Medical History

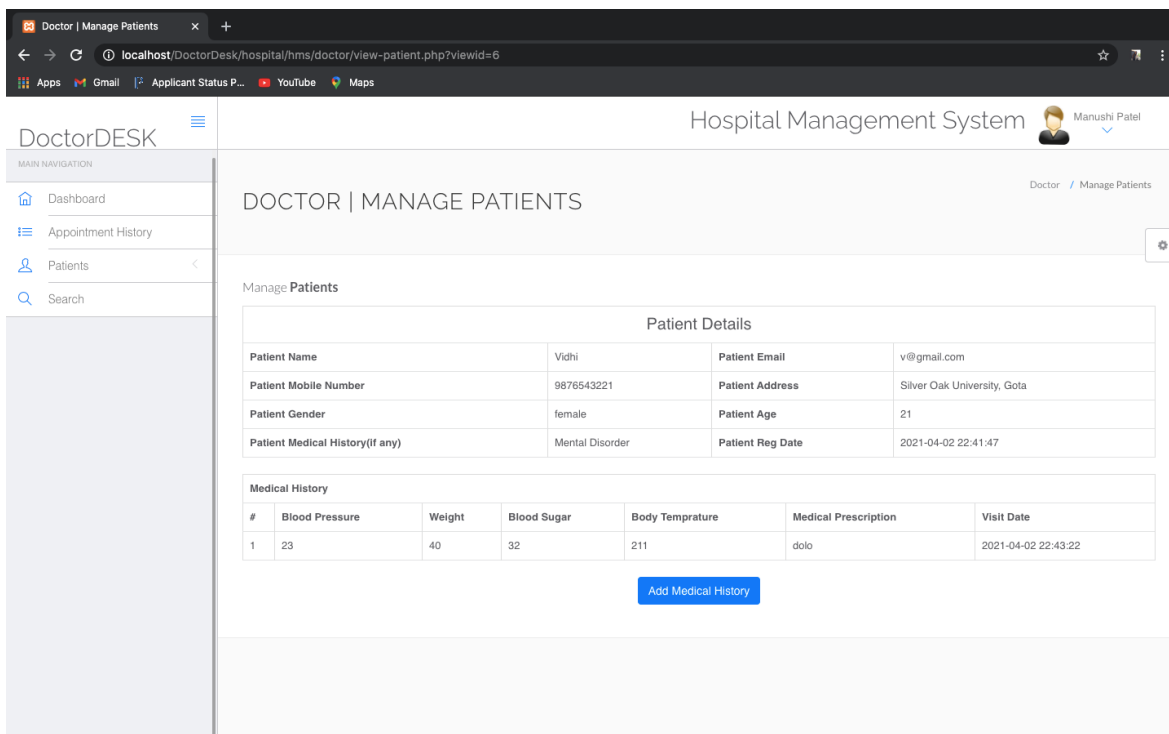
USER | APPOINTMENT HISTORY

#	Doctor Name	Specialization	Consultancy Fee	Appointment Date / Time	Appointment Creation Date	Current Status	Action
1.	Khushi Patel	Dermatologist	5000	2021-04-04 / 12:15 AM	2021-04-04 00:12:42	Active	<a href="#">Cancel</a>

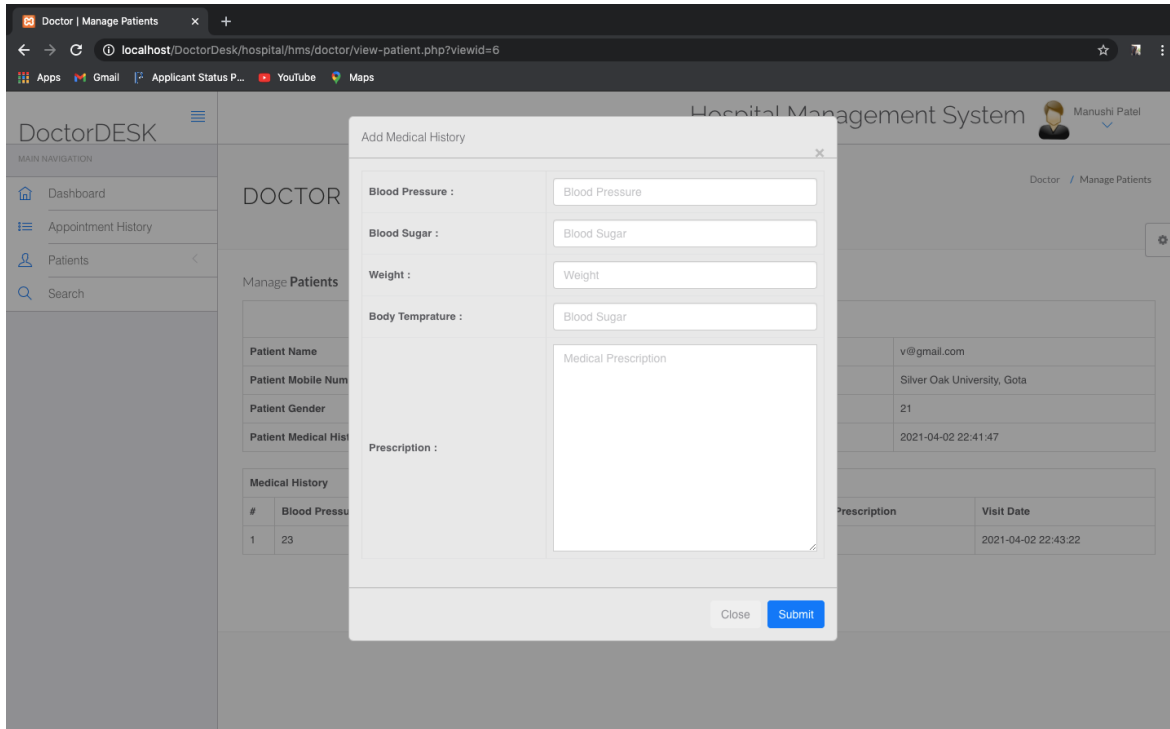
[Figure 8.4 User Appointment History Screenshot]



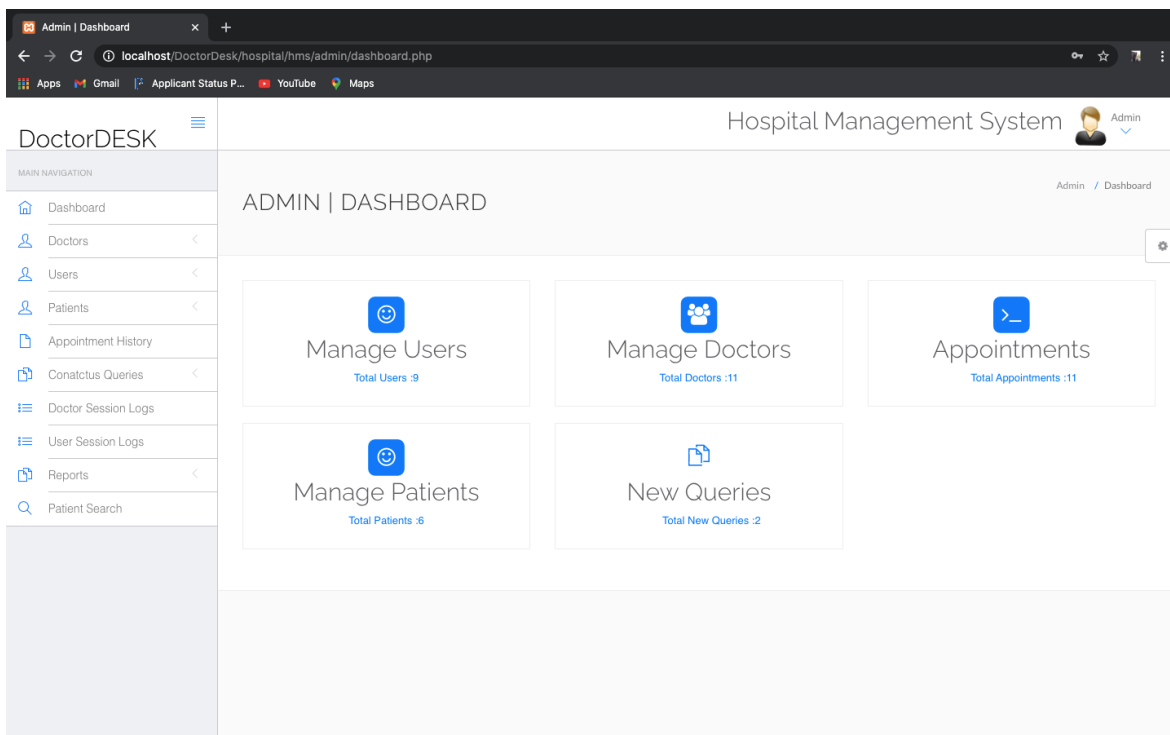
[Figure 8.5 Doctor Dashboard Screenshot]



[Figure 8.6 Doctor Manage Patient Screenshot]



[Figure 8.7 Doctor Add Medical History Screenshot]



[Figure 8.8 Admin Dashboard Screenshot]

The screenshot shows a web browser window with the URL `localhost/DoctorDesk/hospital/hms/admin/add-doctor.php`. The page title is "Admin | Add Doctor". The header includes "Hospital Management System" and a user profile for "Admin". The left sidebar contains navigation icons. The main content area is titled "ADMIN | ADD DOCTOR" and contains a form titled "Add Doctor". The form fields are:

- Doctor Specialization: Select Specialization
- Doctor Name: Enter Doctor Name
- Doctor Clinic Address: Enter Doctor Clinic Address
- Doctor Consultancy Fees: Enter Doctor Consultancy Fees
- Doctor Contact no: Enter Doctor Contact no
- Doctor Email: Enter Doctor Email id
- Password: (field partially visible)

[Figure 8.9 Admin Add Doctor Screenshot]

# **CHAPTER: 9**

## **LIMITATION AND FUTURE ENHANCEMENT**



## **CHAPTER: 9**

### **LIMITATION AND FUTURE ENHANCEMENT**

#### **9.1 Limitation:**

- Software loading time is high.
- More security requires.
- More features can to be Added.

#### **9.2 Future Enhancement:**

- We create an app so a user can use our Software.
- We add more features to create the website more secure.
- We focus to create the software more user-friendly.

# CONCLUSION

## **CONCLUSION**

- After successful completion of the project we expect that it would be helpful to local clinic to gain attraction.
- Also, the patients seeking for local clinics will get to know about it.

# REFERENCE

## **REFERENCES**

<https://www.reddit.com>