# CHAPTER 1 INTRODUCTION

## Introduction:

The aim of developing ERP for a hospital system is to replace the traditional way of managing the information related to health care and aids in the job completion of health care providers with computerized system.

Another important reason to introduce HMS was to solve the complications coming from managing all the paper works of every patient associated with the various departments of hospitalization with confidentiality.

ERP for a hospital system has a very large scope. This PHP project can be used by any hospital or clinic for keeping the records of their patients, staff and other assets.

This project is easy, fast and accurate. It requires less disk space. Online Hospital Management System uses MySQL Server as backend so there is not any chance of data loss or data security.

## Technical specifications:

Specifications of the System used to develop and test the website-: **Operating Systems:**

* + - Windows 10
    - Ubuntu 18.04 LTS
    - Android 8.1 Oreo **Web Browsers:**
    - Google Chrome 68
    - Mozilla Firefox 61

**Desktop:**

* + - Processor: AMD FX-6300 Black Edition Hexa-Core @ 3.5 GHz
    - RAM: 1 x 8 GB HyperX Ripjaws DDR3 @ 1866 MHz **Mobile:**
    - Chipset: Qualcomm MSM8937 Snapdragon 430
    - CPU: Octa-Core 1.4 GHz Cortex-A53
    - GPU: Adreno 505
    - RAM: 3GB **Toolkit:**
    - Visual Studio Code, Notepad++, Sublime
    - XAMPP
    - Adobe XD CC **Platform:**
    - PHP
    - ***Back-end***

- MySQL Database Server

* + - ***Front-End***

-HTML, CSS, Bootstrap

-JavaScript and its Libraries & Frameworks

# CHAPTER 2 TECHNOLOGIES USED

## Front and Back End tools:

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Particulars** | **Technology** |
| **1.** | Server-side language | PHP |
| **2.** | Database | MySQL |
| **3.** | Designing | CSS 3 |
| **4.** | Validation, Animation | JQuery |
| **5.** | Retrieve data from the database | AJAX |
| **6.** | Structure Designing | HTML 5 |
| **7.** | Program the behavior of web pages | JavaScript |
| **8.** | Responsive | Bootstrap 4 |

Table 2.1: Front and Back End tools

## Front End Technologies:

### HYPERTEXT MARKUP LANGUAGE (HTML)

A simple markup language used to create hypertext documents that are portable from one platform to another. HTML files are simple ASCII text files with codes embedded (indicated by markup tags) to denote formatting and hypertext links. Many people who use HTML to create Web pages or other documents find Notepad a useful tool for writing in HTML. Because Notepad supports only very basic formatting, you cannot accidently save special formatting in documents that need to remain pure text. This is especially useful when creating HTML documents for a Web page because special characters or other formatting may not appear in your published Web page or may even cause errors. Many word processors provide additional tools or converters to help you create HTML documents. But, if you are creating simple pages or if you want to make a few quick changes Notepad opens files quickly. Also, Notepad shows all of the HTML tags so you can troubleshoot your page. Not all word processors or converters make the HTML code available. Text editors like Notepad++, Visual Studio Code, Brackets, Sublime provides predefined tags making the use of HTML tags easy.

### HYPERTEXT PRE PROCESSOR (PHP)

The full form of PHP is “Hypertext Pre Processor” but its original name was “Personal Home Page”. Rasmus Lerdorf, software engineer, Apache team member is the creator and original driving force behind PHP. By the middle of 1997, PHP was being used on approximately 50,000 sites across the worldwide.

PHP is a server-side scripting language, which can be embedded in HTML or used as a stand- alone. PHP does not do anything about what a page looks and sounds like. In fact, most of what PHP does is invisible to the end user. Someone looking at PHP page can not necessarily be able to tell it was not written purely in HTML because usually, the result of PHP is HTML.

PHP is fully cross-platform meaning it runs native on several flavours of UNIX, as well as on Windows base system. The PHP allows web developers to create dynamic pages for e- commerce and web applications that are interact with database. When we embed the PHP code into HTML code at that time if we make a PHP file then we have to give the extension of the file is.php. It provides those servers with functionality similar to that provided to windows platform by Active Server Pages technology. It is database supported means we can access commercial and non- commercial databases including Informix, Microsoft SQL server, MySQL, ODBC, Oracle etc.

### JQuery

JQuery is light weight, “write less, do more” JavaScript library. The purpose of JQuery is to make it much easier to use Java Script on your website. JQuery takes a lot of common tasks that required many lines of Java Script code to accomplish, and wraps it into methods that you can call with a single line of code. JQuery also simplifies a lot of complicated things from Java Script, like AJAX calls and DOM manipulation. The JQuery library contains the following features:

* + - 1. HTML/DOM manipulation
      2. Point CSS manipulation
      3. HTML event methods
      4. Effects and Animations
      5. AJAX
      6. Utilities

### JAVASCRIPT

JavaScript is an object-oriented scripting language used to enable programmatic access to objects within both the client application and other applications.

It is primarily used in form of client-side JavaScript, implemented as an integrated component of the web browser, allowing the development of enhanced user interfaces and dynamic websites. JavaScript was first developed by Netscape as an open scripting language to create interactive web pages. JavaScript as an open language implies that it can be used by anyone; no license is required to use JavaScript.

JavaScript has the ability to function both as an object-oriented language as well as procedural language. Using JavaScript, you can create objects, attach methods and properties.

JavaScript helps in performing the following tasks:

* + - 1. JavaScript gives HTML designers a programming tool – HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small “snippets” of code into their HTML pages.
      2. JavaScript can read and write HTML elements - JavaScript can read and change the context of an HTML element.
      3. JavaScript can be used to validate data - A JavaScript can be used to validate from data before it is submitted to server. This saves the server from extra processing.
      4. JavaScript can be used to create cookies – A JavaScript can be used to store and retrieve information on the visitor’s computer.

### Asynchronous JavaScript and XML (AJAX)

AJAX is a technique for creating fast and dynamic web pages. AJAX is not itself a technology. AJAX allows web pages to be updated asynchronously by exchanging a small amount of data with the server behind the scenes. This means that it is possible to update parts of a web page without reloading the whole page.

AJAX is a way of mixing well-known programming techniques in an uncommon way to develop web-applications with appealing user interfaces.

AJAX works as an extra layer between the user’s browser and the web server to fill up the lapses of traditional web applications development. Examples of applications using AJAX: Google Maps, Gmail, YouTube, and LinkedIn etc.

### CASCADING STYLE SHEET (CSS)

CSS is used to describe the presentation semantics (that is the look and formatting) of a document written in a markup language. CSS is designed primarily to enable the presentation of document content (Written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts.

Cascading Style Sheet is used to style HTML elements. Three Ways to Insert CSS in a web page:

* + - 1. External style sheet: In separate style sheet files (CSS files).
      2. Internal style sheet: In the style element in the HTML head section.
      3. Inline style: In the style attribute in single HTML elements.

### BOOTSTRAP

Bootstrap is a free and open source front-end framework for developing websites and web applications. It contains HTML and CSS based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many earlier web frameworks, it concerns itself with [front-end development](https://en.wikipedia.org/wiki/Front-end_web_development) only.

Bootstrap 4 supports the latest versions of the [Google Chrome](https://en.wikipedia.org/wiki/Google_Chrome), [Firefox](https://en.wikipedia.org/wiki/Firefox), [Internet](https://en.wikipedia.org/wiki/Internet_Explorer) [Explorer](https://en.wikipedia.org/wiki/Internet_Explorer), [Opera,](https://en.wikipedia.org/wiki/Opera_(web_browser)) and [Safari](https://en.wikipedia.org/wiki/Safari_(web_browser)) . Since 2.0, Bootstrap supports [responsive web design](https://en.wikipedia.org/wiki/Responsive_Web_Design). This means the layout of web pages adjusts dynamically, taking into account the characteristics of the device used (desktop, tablet, mobile phone). Starting with version 3.0, Bootstrap adopted a [mobile-first design](https://en.wikipedia.org/wiki/Mobile-first_design) philosophy, emphasizing responsive design by default.

Bootstrap is modular and consists of a series of style sheets that implement the various components of the toolkit. These style sheets are generally compiled into a bundle and included in web pages, but individual components can be included or removed. Bootstrap provides a number of configuration variables that control things such as color and padding of various components. Grid system and responsive design come standard with an 1170-pixel- wide [grid layout.](https://en.wikipedia.org/wiki/CSS_Grid_Layout) Alternatively, the developer can use a variable-width layout. For both cases, the toolkit has four variations to make use of different resolutions, and types of devices: mobile phones, portrait and landscape, tablets and PCs with low and high resolution. Each variation adjusts the width of the columns.

## Back End Technology:

### MySQL SERVER

A Database management system must be able to reliably manage a large amount of data in a multi-user environment so that many users can concurrently access the data. A Database management system should also provide security and failure recovery.

MySQL is the name of a database management system Developed by Apache Software Foundation. Apache Software Foundation provides various softwares related to web development; the biggest advantage is that all software’s are license free. MySQL is the most popular database management tool. It provides better security and data recovery against system failure.

MySQL works faster compared to other Database Management tools. Apache Software Foundation provides GUI to work with MySQL, because of its GUI environment it is very easy to understand, any novice user can quickly learn MySQL by using phpMyAdmin developer. It provides GUI so it is easy to understand and learn.

MySQL is a database management system, which helps us to manage data stored in a system database. Programming skills are not required to use MySQL as a database management system. PhpMyAdmin hides all the complex tasks of managing a database and gives a user- friendly way of managing a database.

# CHAPTER 3

**SYSTEM ANALYSIS & SYSTEM STUDY**

## Project Analysis:

Project Analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the project. It is a problem-solving activity that requires intensive communication between the project users and project developers. Project Analysis or study is an important phase of any project development process.

The Project Analyst plays a role of the interrogator and dwells deep into the working of the present project. Our project is viewed as a whole and the inputs to the project are identified. The outputs from the organizations are traced to the various processes. Project Analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and the various factors and determining an optimal or at least a satisfactory solution or program of action. Preliminary Study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary Study is the problem-solving activity that requires intensive communication between the Project Users and Project Developers.

Here in “Sohn Mariens Hospital Management System”, a detailed study of Existing System is carried along with all the steps in Project Analysis. An idea of creating a better project was carried and the next steps were followed. The following problems were identified by us in the existing system:

* Non-existent online management system.
* Maximum amount of information is still stored in old fashioned pen-paper method.
* High chances of human error
* Weak measures of security and confidentiality.

## Feasibility Study:

The preliminary investigation examines a project’s feasibility, the likelihood that the system will be useful in the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are unlimited resources and infinite time. In our project, there are three aspects in the feasibility study portion of the preliminary investigation:

1. Technical Feasibility
2. Economic Feasibility
3. Operational Feasibility

### Technical Feasibility

The technical issues are usually raised during the feasibility study stage of the investigation which includes the following:

* + - 1. All the necessary technology was used to do the work.
      2. The proposed equipment had the technical capacity to hold the data required and we also used new systems that were not mentioned in synopsis.
      3. The proposed system provides an adequate response to inquiries, regardless of the number or location of users.
      4. The system can be upgraded at any point of time.
      5. Technical guarantees of accuracy, reliability, ease of access and data security are taken care of.

### Economic Feasibility

In economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financially benefits must either be equal or should exceed the costs. A system developed after a careful evaluation of the concerned factors saves a lot of money beforehand for the developer as well as the organization. It is the duty of developer to analyze all the pros and cons to create a system is a big investment for the organization.

### Operational Feasibility

Proposed web application is beneficial only if it can be turned into an information system that will meet the organization’s operational requirements. Operational feasibility aspects of this project are to be taken as an important part of the project implementation. Some of the important issues raised to test the operational feasibility of a project include the following:

* + - 1. There is sufficient support for the management of the users.
      2. The system can be used and will work properly.

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

## Cost & Benefit Analysis:

In developing cost estimates for a system, we need to consider several cost elements. Among them are: hardware, personnel, facility, operating and supply costs. A system is also expected to provide benefits. The first task is to identify each benefit and then assign a monetary value to it for cost & benefit analysis. Benefits may be tangible or intangible.

The two major benefits are improving performance and minimizing the cost of processing. The performance category emphasizes improvement in the accuracy of or access to information and easier access to the system by authorized users.

There is a difference between expenditure and investment. We spend to get what we need, but we invest to realize a return on investment. Building a computer-based system is an investment. Benefits are realized in the form of reducing operating costs, improved corporate image, staff efficiency, or revenues. To what extent benefits outweigh costs is the function of cost & benefit analysis.

Costs are incurred throughout its life cycle. Cost & benefit analysis a procedure that gives a picture of various costs, benefits, and rules associated with a system. The determination of cost costs and benefits entails the following steps:

1. Identify the cost and benefits pertaining to a given project.
2. Categorize the various costs and benefits for analysis.
3. Select a method for evaluation.
4. Interpret the results for analysis and take action

# CHAPTER 4 PROJECT DESIGN

## Data Dictionary:

A data dictionary is a collection of descriptions of the [data](https://searchdatamanagement.techtarget.com/definition/data) objects or items in a data model for the benefit of programmers and others who need to refer to them. When developing programs that use the data model, a data dictionary can be consulted to understand where a data item fits in the structure, what values it may contain, and basically what the data item means in real-world terms.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Page Name** | **Identifier** | **Use** |
| 1. | admit\_cdetails.php | $id | To get the patient admit id. |
| 2. | admit\_cdetails.php | $r1 | Result of the query string. |
| 3. | admit\_cdetails.php | #back | To apply CSS |
| 4. | admit\_cdetails.php | #reprint | To apply CSS |
| 5. | admit\_details.php | #example | To create a data table for user details. |
| 6. | admit\_details.php | #area | Creating an area for printing the dynamically changing tables. |
| 7. | admit\_details.php | $r | Result of the query string |
| 8. | admit\_details.php | $row | Store the result of the query string. |
| 9. | admit\_details.php | $i | Loop counter |
| 10. | appointment\_details.php | #example | To create a data table for user details. |
| 11. | appointment\_details.php | #area | Creating an area for printing the dynamically changing tables. |
| 12. | complete\_app.php | $x | Store the received value. |
| 13. | complete\_app.php | $r | Result of the query string |
| 14. | cpu.php | checkpass() | To check if the inserted passwords match or not. |

|  |  |  |  |
| --- | --- | --- | --- |
| 15. | cpu.php | var x | Variable for password one. |
| 16. | cpu.php | var y | Variable for repeated password. |
| 17. | cpu.php | $id | To get the employee id of the current user. |
| 18. | cpu.php | $r | Result of the query string. |
| 19. | cpu.php | $row | To store the result of the query string. |
| 20. | cpu.php | #back | To apply CSS |
| 21. | cpu.php | .box-body | To apply CSS |
| 22. | cpu.php | #det | To apply CSS and animations. |
| 23. | cpu.php | $pass | Stores the new password. |
| 24. | cpu.php | $passc | Stores the repeated password |
| 25. | cpu.php | #pass | To apply validation using JavaScript. |
| 26. | cpu.php | #passc | To apply validation using JavaScript. |
| 27. | cpu.php | #msg | To apply CSS |
| 28. | cpu.php | .error1 | To apply CSS |
| 29. | cpu.php | #btttn | To apply CSS |
| 30. | cpu.php | #btt2 | To apply CSS |
| 31. | cpu1.php | $id | To get the user ID |
| 32. | cpu1.php | $pass | To get the changed password |
| 33. | dashboard.php | $date | To get current date |
| 34. | dashboard.php | $timestamp | To get the current time stamp |
| 35. | dashboard.php | $day | To store the current day |
| 36. | dashboard.php | $days | Array of all the day names |
| 37. | dashboard.php | $x | Store the final result of day. |
| 38. | dashboard.php | .table-body | To apply CSS |
| 39. | dashboard.php | #example | To create a data table |
| 40. | dashboard.php | #area | Creating an area to print the dynamically changing tables. |
| 41. | dashboard.php | $r | Result of query string. |
| 42. | dashboard.php | $row | Store the result of the query string. |
| 43. | department.php | #example | To create a data table |
| 44. | department.php | .table-body | To apply CSS. |
| 45. | department.php | #area | Creating an area to print the dynamically changing tables. |
| 46. | department.php | $con | Connection Variable |
| 47. | department.php | $r | Result of the query string |
| 48. | department.php | $row | Store the result of the query string. |
| 49. | header.php | $\_SESSION[“eid”] | Stores the id of the user that is logged in. |
| 50. | header.php | $\_SESSION[“dept”] | Stores the id of the user that is logged in. |
| 51. | header.php | $eid | To store the id |
| 52. | header.php | $r1 | Result of the query string |
| 53. | header.php | $row | Stores the result of the query string. |
| 54. | header.php | $dept | Stores the department of the user |
| 55. | header.php | $r | Result of the query string |
| 56. | header.php | .hname | To apply CSS |
| 57. | header.php | #logo | To apply CSS |
| 58. | header.php | #mySidebar | To create a dynamic sidebar |
| 59. | header.php | w3\_close() | Function to close the sidebar |
| 60. | header.php | .sb | To apply CSS |
| 61. | header.php | w3\_open() | Function to open the sidebar |
| 62. | header.php | dc | To apply CSS |
| 63. | header.php | day | To apply CSS |
| 64. | header.php | cc | To apply CSS |
| 65. | header.php | clock | To apply CSS |
| 66. | header.php | uname | To apply CSS |
| 67. | header.php | user-name | To apply CSS |
| 68. | header.php | $dname | To print the username |
| 69. | header.php | .logout | To apply CSS |
| 70. | header.php | #lo | To apply CSS |
| 71. | header.php | showTime() | Function to put a dynamically changing clock |
| 72. | header.php | time | To store the object of Date class. |
| 73. | header.php | days | Array of the name of days |
| 74. | header.php | dayName | Gets the name of the day today |
| 75. | header.php | hour | Gets the hours |
| 76. | header.php | min | Gets the value of minute |
| 77. | header.php | sec | Gets the value of seconds |
| 78. | header.php | currentTime | Stores the final formatted form of time. |
| 79. | pathology\_part.php | $id | Gets the id of the patient |
| 80. | pathology\_part.php | $r | Result of the query string |
| 81. | pathology\_part.php | $row | Stores the result of query string |
| 82. | pathology\_part.php | #dts | To apply CSS |
| 83. | pathology\_part.php | .heading | To apply CSS |
| 84. | pathology\_part.php | .address | To apply CSS |
| 85. | pathology\_part.php | .heading1 | To apply CSS |
| 86. | pathology\_part.php | .patient-details | To apply CSS |
| 87. | pathology\_part.php | $tests[] | To select the list of all the diagnostic tests to be administered. |
| 88. | pathology\_part.php | #bttn | To apply CSS |
| 89. | pathology\_part.php | #bttn2 | To apply CSS |
| 90. | patient.php | #example | To create a data table |
| 91. | patient.php | .table-body | To apply CSS |
| 92. | patient.php | .heading | To apply CSS |
| 93. | patient.php | #area | To create an area for dynamically changing data tables. |
| 94. | patient.php | $r | Result of query string |
| 95. | patient.php | $row | Stores the result of the index string |
| 96. | prescription.php | #dts\_sec | To apply CSS |
| 97. | prescription.php | #dts\_btn | Button to toggle the display |
| 98. | prescription.php | #pres\_sec | To apply CSS |
| 99. | prescription.php | #pres\_btn | Button to toggle the display |
| 100. | prescription.php | dts() | Function to hide one section and show another |
| 101. | prescription.php | pres() | Function to hide one section and show another |
| 102. | prescription\_db.php | $id | To store the patient ID |
| 103. | prescription\_db.php | $diag | To store the diagnosis |
| 104. | prescription\_db.php | $type | To store the type of medicine |
| 105. | prescription\_db.php | $med | To store the name of the medicine |
| 106. | prescription\_db.php | $freq | To store the frequency prescribed. |
| 107. | prescription\_db.php | $dur | To store the duration of the treatment. |
| 108. | prescription\_db.php | $i | Loop Counter |
| 109. | prescription\_db.php | $med[] | Final value of medicine |
| 110. | prescription\_db.php | $r | Result of the query string |
| 111. | prescription\_db.php | $tests | Final value of all the diagnostic tests. |
| 112. | prescription\_print.php | $id | To store the patient id |
| 113. | prescription\_print.php | $r | Result of the query string |
| 114. | prescription\_print.php | $row | Stores the result of the query string |
| 115. | prescription\_print.php | #pres | To apply CSS |
| 116. | prescription\_print.php | .box-body | To apply CSS |
| 117. | prescription\_print.php | .heading | To apply CSS |
| 118. | prescription\_print.php | .address | To apply CSS |
| 119. | prescription\_print.php | .heading1 | To apply CSS |
| 120. | prescription\_print.php | .patient-details | To apply CSS |
| 121. | prescription\_print.php | #med | To apply CSS |
| 122. | print\_prescription.php | $x | To store the user ID |
| 123. | print\_prescription.php | #back | To apply CSS |
| 124. | print\_prescription.php | #reprint | To apply CSS |
| 125. | print\_prescription.php | .heading | To apply CSS |
| 126. | print\_prescription.php | .box-body | To apply CSS |
| 127. | print\_prescription.php | #bttn | To apply CSS |
| 128. | index.php | .box-body | To apply CSS |
| 129. | index.php | $name | Store the username |
| 130. | index.php | $pass | Store the password |
| 131. | index.php | $\_SESSION[“err”] | Shows error if the login fails |
| 132. | mydetails.php | $r | Result of the query string |
| 133. | mydetails.php | $r1 | Result of the query string |
| 134. | mydetails.php | .bg-modal | To apply CSS |
| 135. | mydetails.php | .modal-content | To apply CSS |
| 136. | mydetails.php | close-button | To close the modal |
| 137. | mydetails.php | .btne2 | To apply CSS |
| 138. | mydetails.php | .btne | To apply CSS |
| 139. | warddetails.php | #wno | To filter the wards |
| 140. | warddetails.php | #area | To create an area for dynamically changing data tables. |
| 141. | warddetails.php | .table-body | To apply CSS |
| 142. | warddetails.php | .heading | To apply CSS |
| 143. | warddetails.php | $i | Loop Counter |
| 144. | warddetails.php | $x | To print bed no. |
| 145. | warddetails.php | $row | To store the result of the string query. |

Table 4.1: Data dictionary

## Data Flow Diagram (DFD):

A DFD is a graphical representation of the “flow” of data through an Information System. A DFD can also be used for the visualization of Data Processing. It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then “exploded” to show the detail of the system being modeled. A DFD represents of flow of data through a system. Data flow diagrams are commonly used during problem analysis. It views a system as a function that transforms the input into desired output.

A DFD shows the movement of data through the different transformations or processes in the system. DFD can be used to provide the end user with a physical structure. The input ultimately has an effect upon the structure of the whole system from order to dispatch to restock how any system developed can be determined through a data flow diagram. The appropriate register saved in the database and maintained by appropriate authorities.

### Zero Level DFD:

Figure 4.2.1: L Zero Level Diagram

## Level one DFD for 1.0:

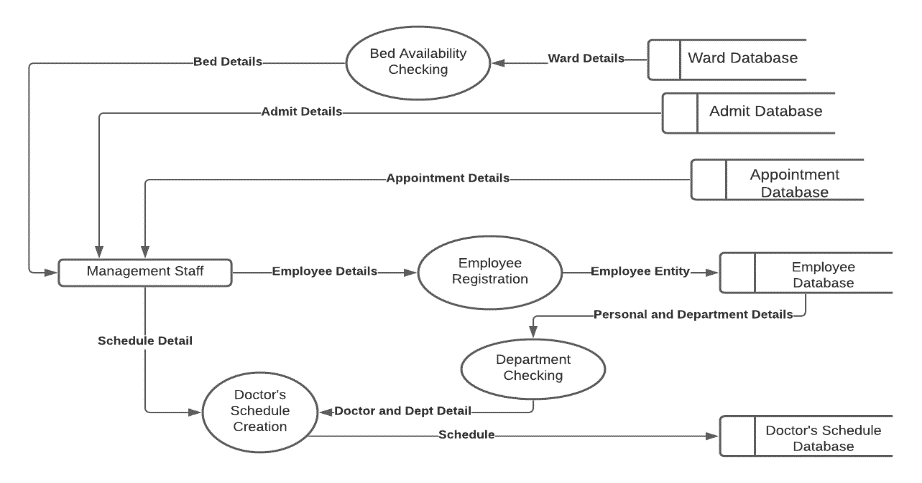


Figure 4.2.2: Level one DFD for 1.0 diagram

## Level one DFD for 2.0:

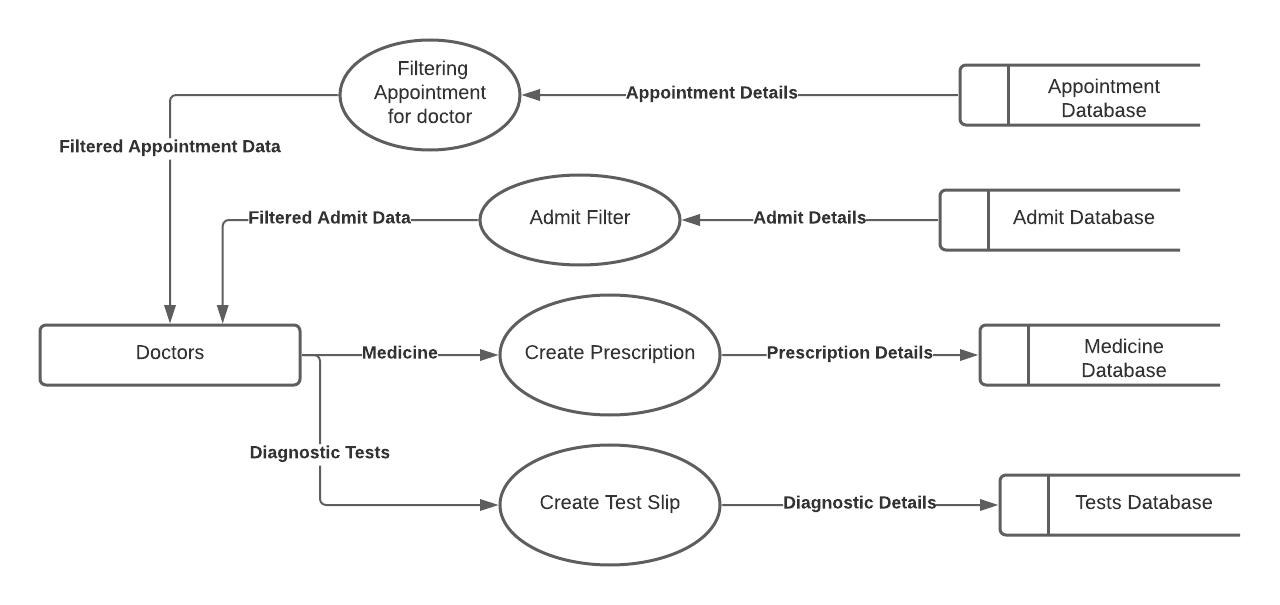


Figure 4.2.3: Level one DFD for 2.0 diagram

## Level one DFD for 3.0:

Figure 4.2.4: Level one DFD for 3.0 diagram

## Level one DFD for 4.0:

Figure 4.2.5: Level one DFD for 4.0 diagram

## Database structures:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Size** | **Constraints** |
| id | Int | 11 | PRIMARY KEY, AUTO INCREMENT |
| Username | Varchar | 110 | NOT NULL |
| eid | Varchar | 200 | NOT NULL |
| password | Varchar | 110 | NOT NULL |
| department | Varchar | 110 | NOT NULL |
| isActive | Tinyint | 1 | NOT NULL |
| isOnline | Tinyint | 1 | NOT NULL |
| At\_date | date |  | NOT NULL |
| At\_time | time |  | NOT NULL |

Table 4.3.1: Database structure of user login credentials

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Size** | **Constraints** |
| Pid | Int | 11 | PRIMARY KEY, AUTO INCREMENT |
| Patid | Int | 11 | NOT NULL |
| Pname | Varchar | 110 | NOT NULL |
| Age | Int | 11 | NOT NULL |
| Gender | Varchar | 100 | NOT NULL |
| Ad\_date | Date |  | NOT NULL |
| Ad\_time | Time |  | NOT NULL |
| Department | Int | 11 | NOT NULL |
| Doc\_name | Varchar | 200 | NOT NULL |
| Cname | Varchar | 200 | NOT NULL |
| Rel | Varchar | 200 | NOT NULL |
| Phno | Varchar | 15 | NOT NULL |
| Mail | Varchar | 100 | NOT NULL |
| Ward | Varchar | 100 | NOT NULL |
| Bno | Int | 20 | NOT NULL |
| Isadmit | Tinyint | 1 | NOT NULL |
| Dis\_date | Date |  | NULL |
| Dis\_time | Time |  | NOT NULL |
| Vistag | Tinyint | 1 | NOT NULL |

Table 4.3.2: Database structure of admit patient details

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Size** | **Constraints** |
| Dept\_id | Int | 11 | PRIMARY KEY, AUTO INCREMENT |
| Name | Varchar | 25 | NOT NULL |
| isDoctor | Tinyint | 1 | NOT NULL |
| isloginallowed | Tinyint | 1 | NOT NULL |

Table 4.3.3: Database structure of departments in the hospital

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Size** | **Constraints** |
| Id | Int | 11 | PRIMARY KEY, AUTO INCREMENT |
| Eid | Int | 10 | NOT NULL |
| Mon | Varchar | 50 | NOT NULL |
| Tues | Varchar | 50 | NOT NULL |
| Wed | Varchar | 50 | NOT NULL |
| Thurs | Varchar | 50 | NOT NULL |
| Fri | Varchar | 50 | NOT NULL |
| Sat | Varchar | 50 | NOT NULL |
| Sun | Varchar | 50 | NOT NULL |

Table 4.3.4: Database structure of doctor’s duties

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Size** | **Constraints** |
| ID | Int | 10 | PRIMARY KEY, AUTO INCREMENT |
| Name | Varchar | 111 | NOT NULL |
| Fname | Varchar | 111 | NOT NULL |
| Mname | Varchar | 111 | NOT NULL |
| Gender | Varchar | 111 | NOT NULL |
| Dob | Date |  | NOT NULL |
| Aadhar | Varchar | 250 | NOT NULL |
| Photo | Varchar | 111 | NOT NULL |
| Mob | Varchar | 111 | NOT NULL |
| Mobb | Varchar | 111 | NOT NULL |
| Email | Varchar | 111 | NOT NULL |
| R\_address | Varchar | 200 | NOT NULL |
| P\_address | Varchar | 200 | NOT NULL |
| Lastjob | Varchar | 40 | NOT NULL |
| Oname | Varchar | 111 | NOT NULL |
| Pan | Varchar | 10 | NOT NULL |
| Bname | Varchar | 300 | NOT NULL |
| Salary | Float |  | NOT NULL |

Table 4.3.5: Database structure of employee details

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Size** | **Constraints** |
| Id | Int | 11 | PRIMARY KEY, AUTO INCREMENT |
| Diag | Varchar | 255 | NOT NULL |
| Type | Varchar | 255 | NOT NULL |
| med | Varchar | 255 | NOT NULL |
| Freq | Varchar | 255 | NOT NULL |
| Dur | Int | 11 | NOT NULL |
| Apid | Varchar | 255 | NOT NULL |

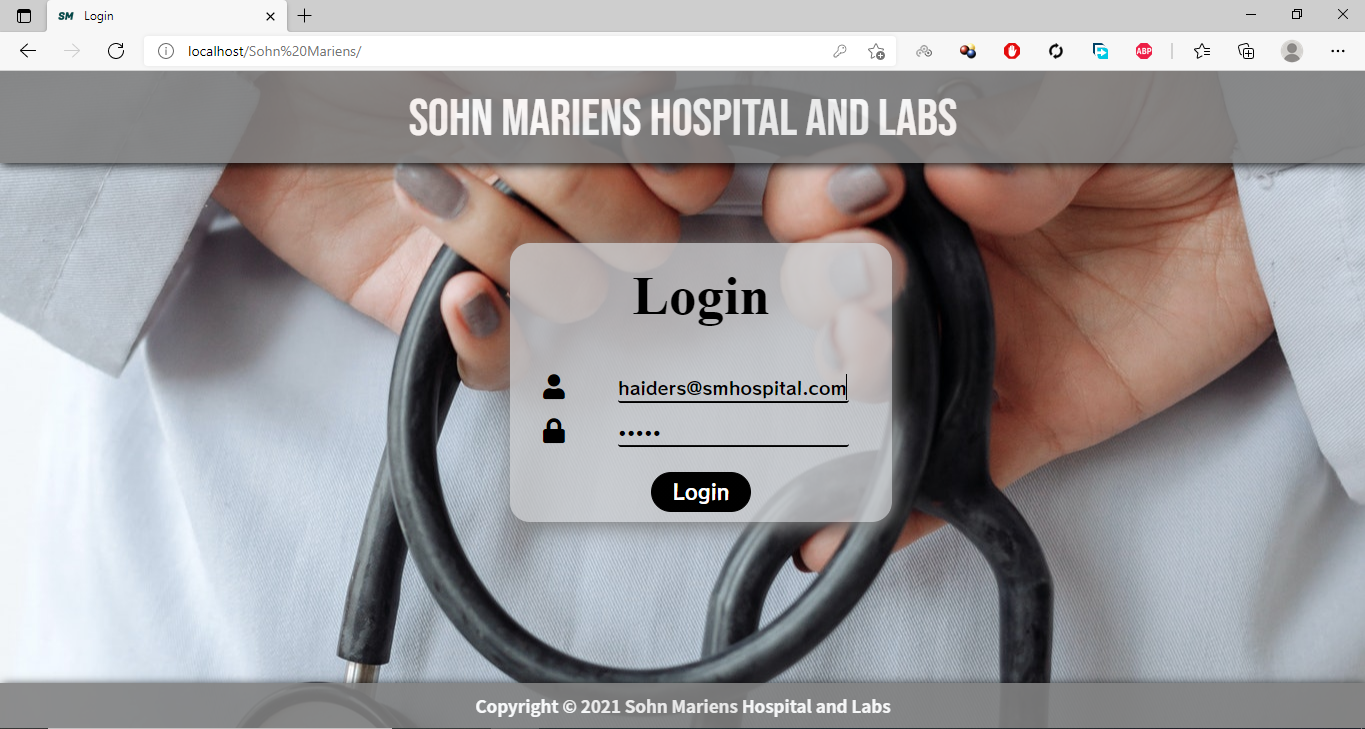
Table 4.3.6: Database structure for medicine prescription

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Size** | **Constraints** |
| Id | Int | 11 | PRIMARY KEY, AUTO INCREMENT |
| Sname | Varchar | 50 | NOT NULL |
| Saddress | Varchar | 200 | NOT NULL |
| Adate | Date |  | NOT NULL |
| Atime | Varchar | 255 | NOT NULL |
| SentTo | Varchar | 25 | NOT NULL |
| isdis | Tinyint | 1 | NOT NULL |
| Dis\_date | Date |  | NULL |
| Dis\_time | Time |  | NOT NULL |

Table 4.3.7: Database structure for parcel details

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Size** | **Constraints** |
| Id | Int | 11 | PRIMARY KEY, AUTO INCREMENT |
| Pid | Int | 11 | NOT NULL |
| Name | Varchar | 255 | NOT NULL |
| Fname | Varchar | 255 | NOT NULL |
| Age | Int | 11 | NOT NULL |
| Gender | Varchar | 20 | NOT NULL |
| Date | Date |  | NOT NULL |
| Doc\_name | Int | 11 | NOT NULL |
| Test | Varchar | 2000 | NOT NULL |

## Snap Shots of designing phase:

Image 4.4.1: Login Page for Hospital Management System

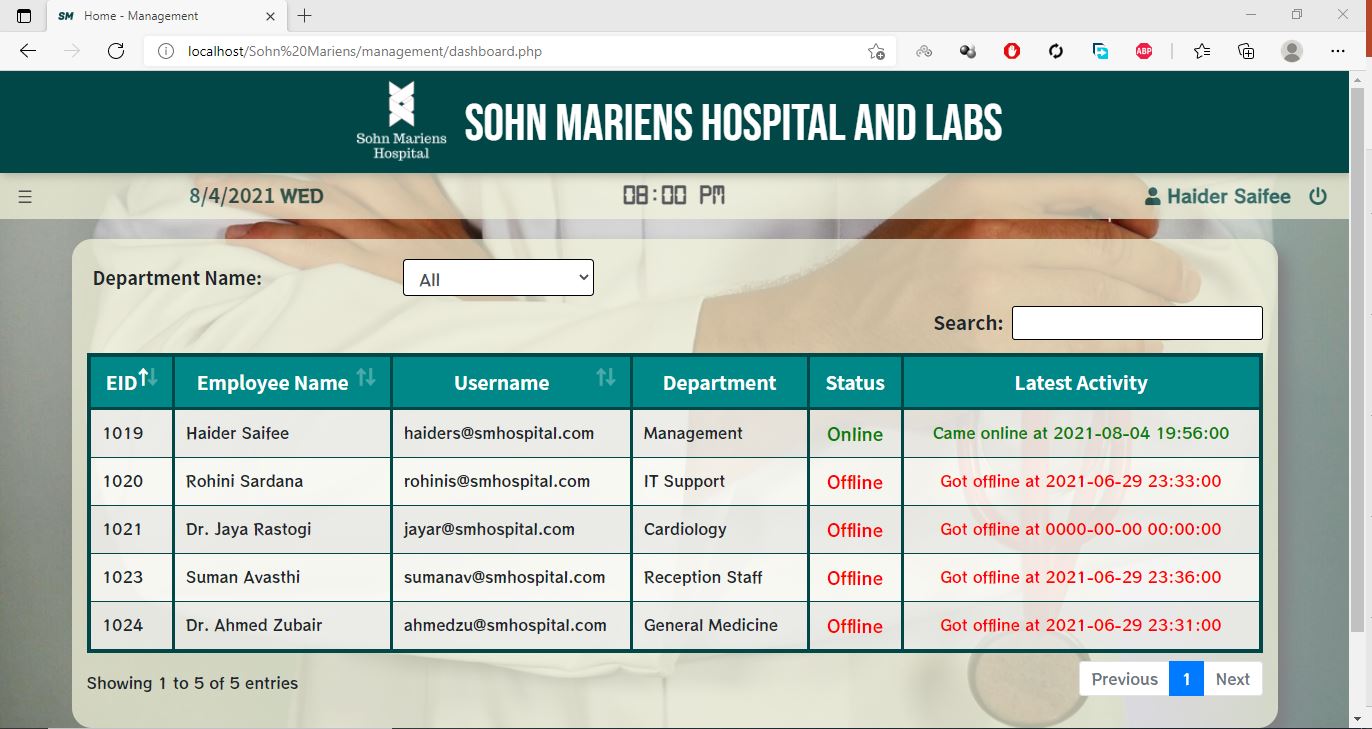
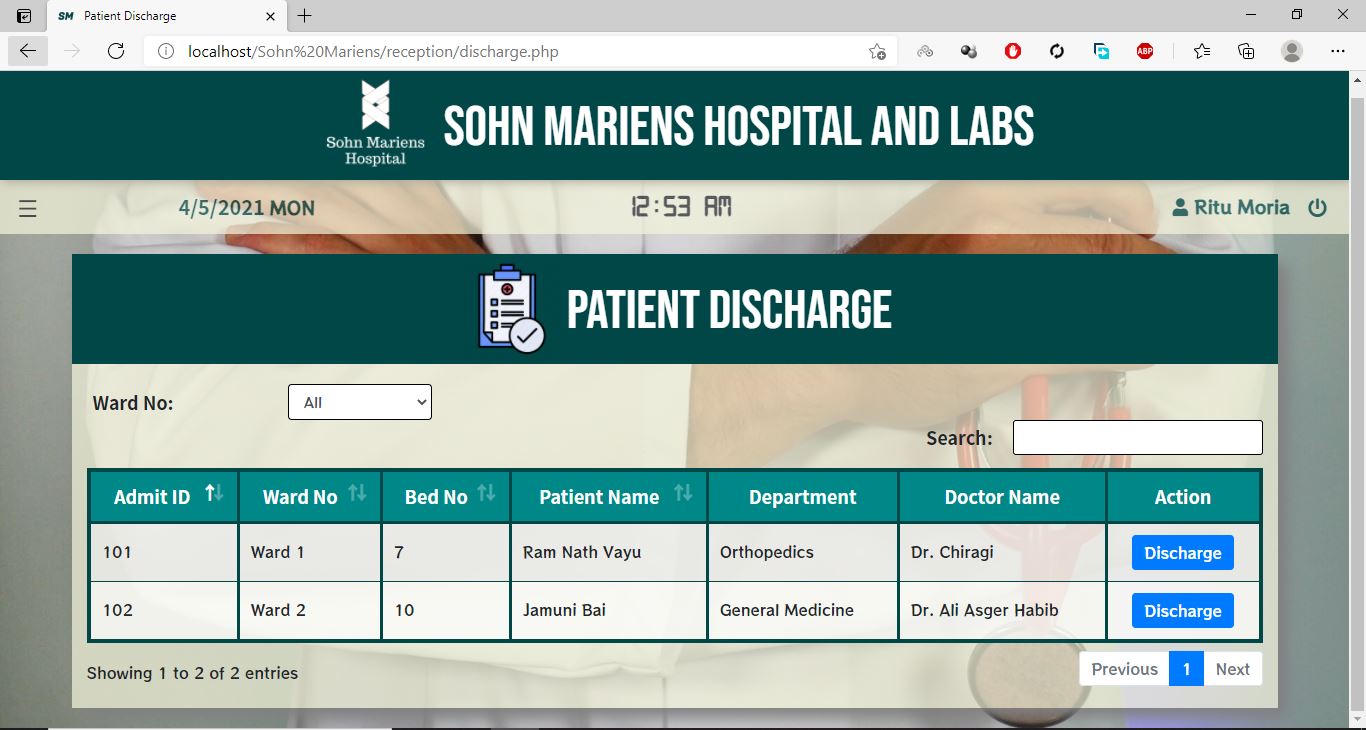


Image 4.4.2: Dashboard for management users

Image 4.4.3: Portal for patient discharge

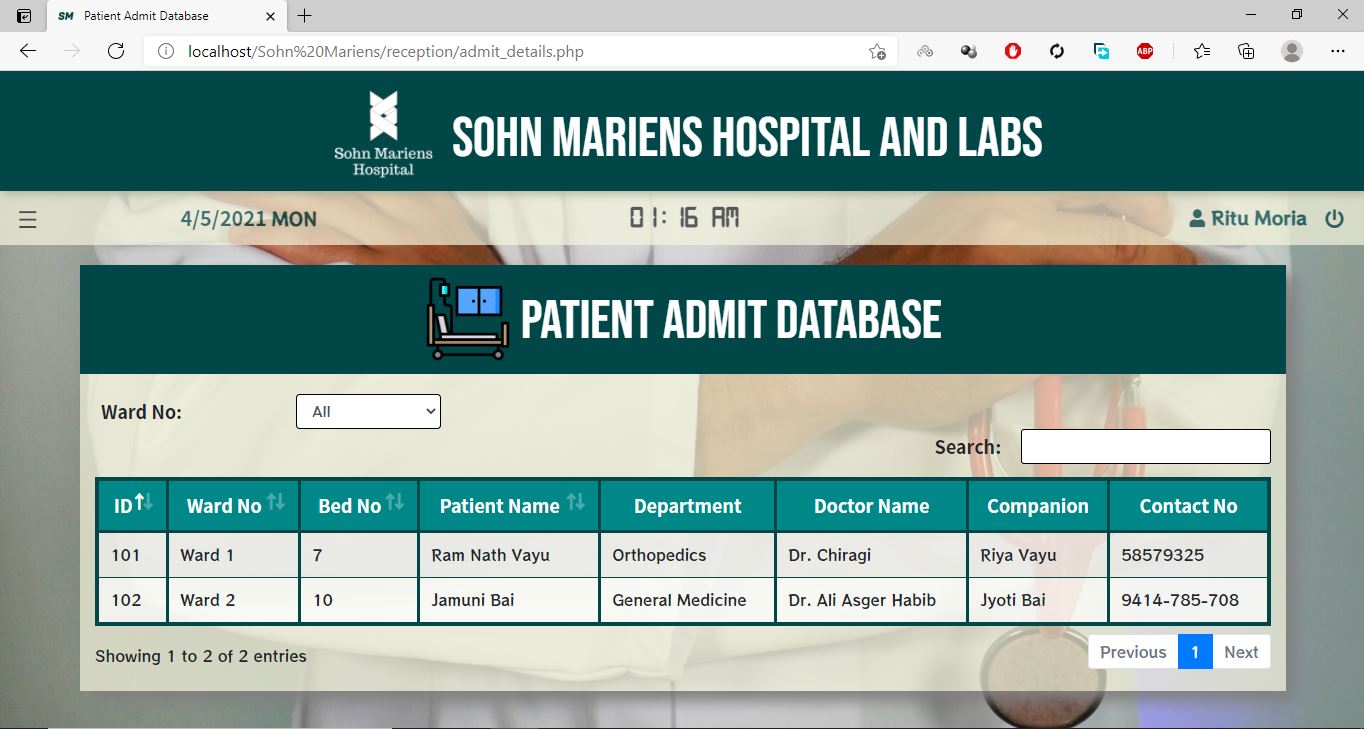


Image 4.4.4: Portal for admitted patient details

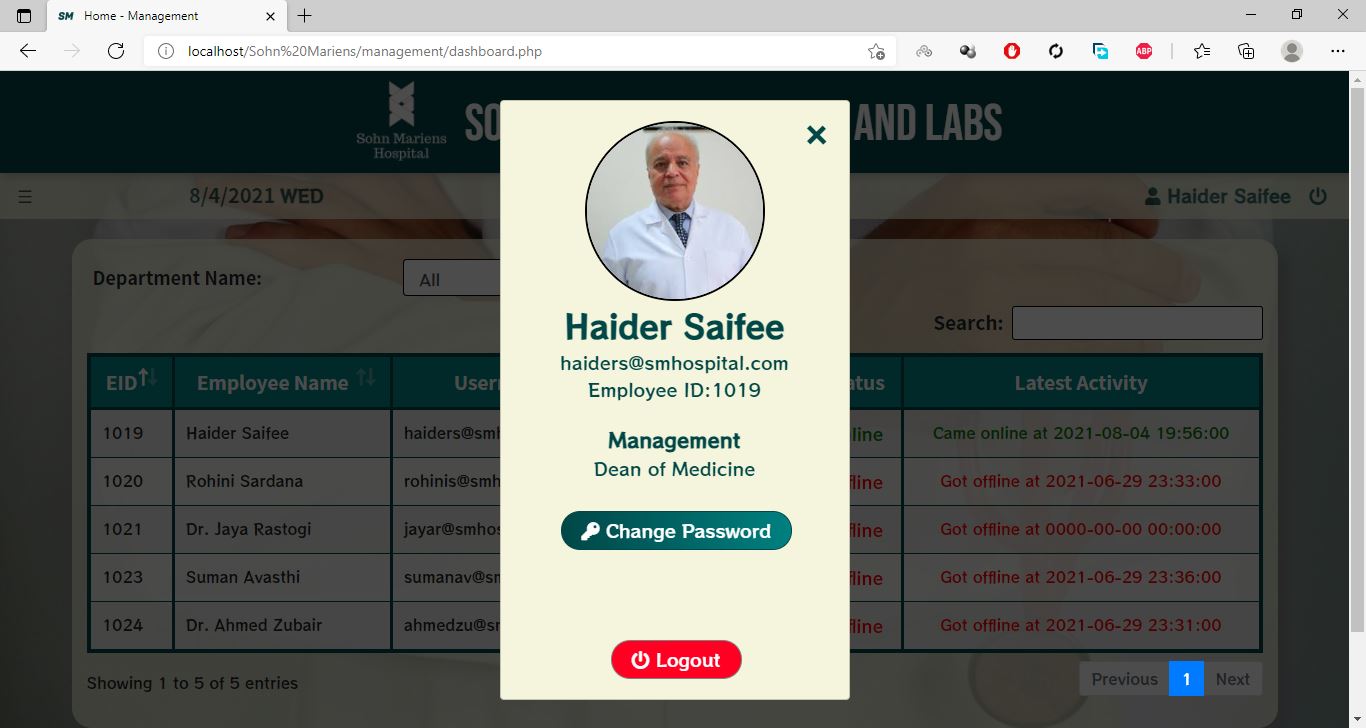


Image 4.4.5: Modal provided to each user for basic details and to change password

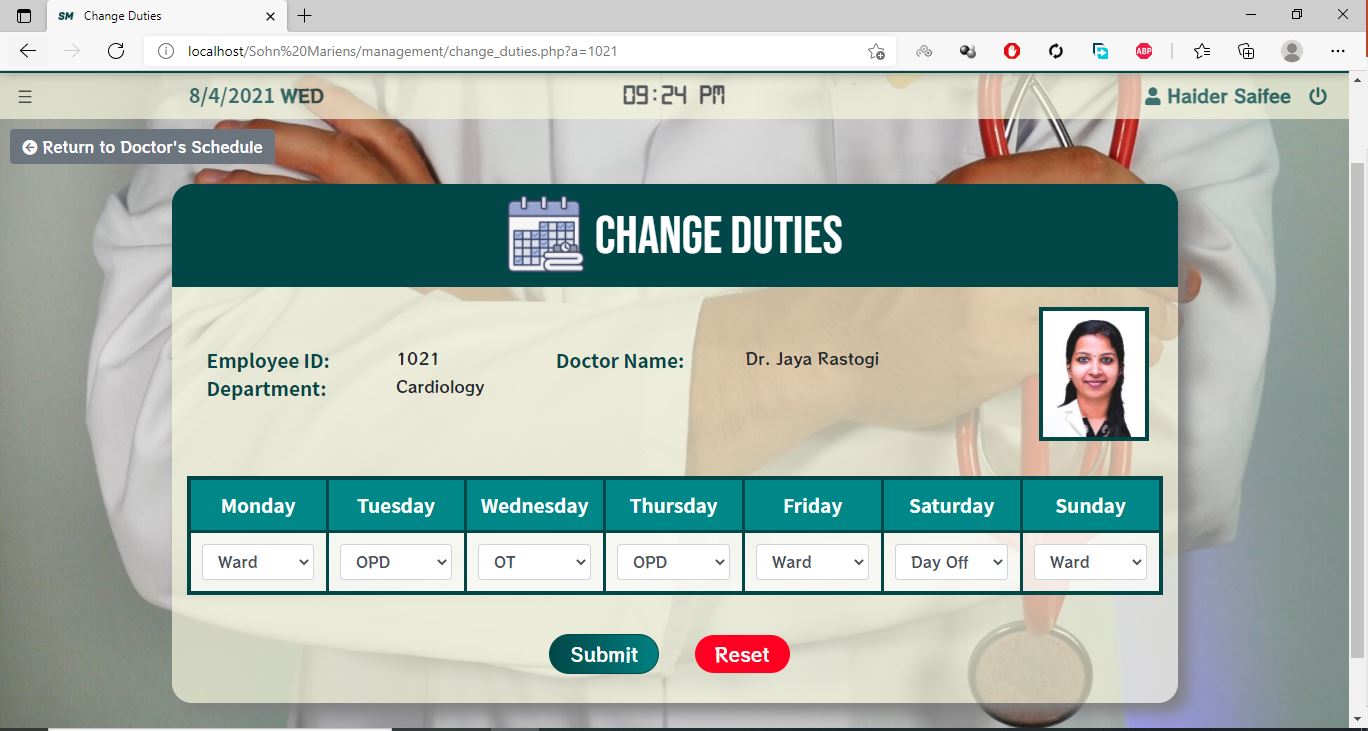


Image 4.4.6: Portal for Changing Doctor Duties

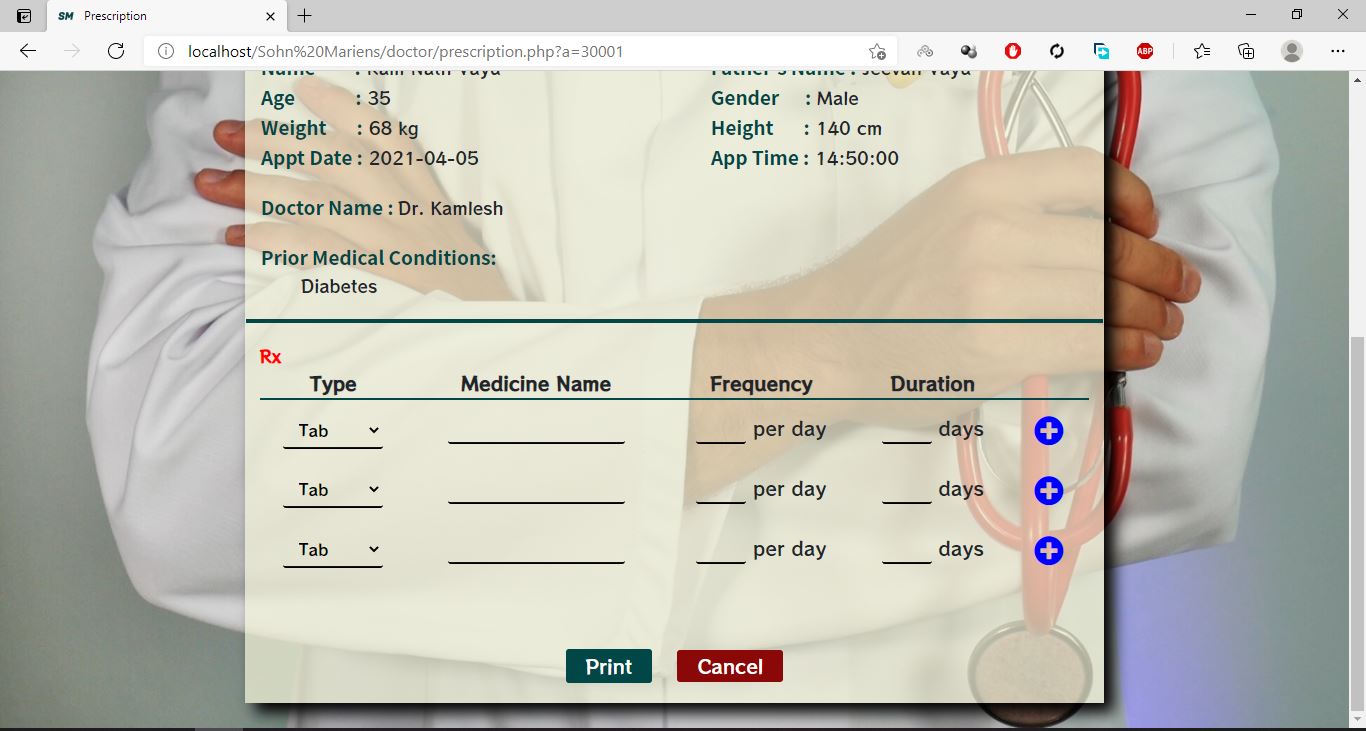


Image 4.4.7: Prescription creator

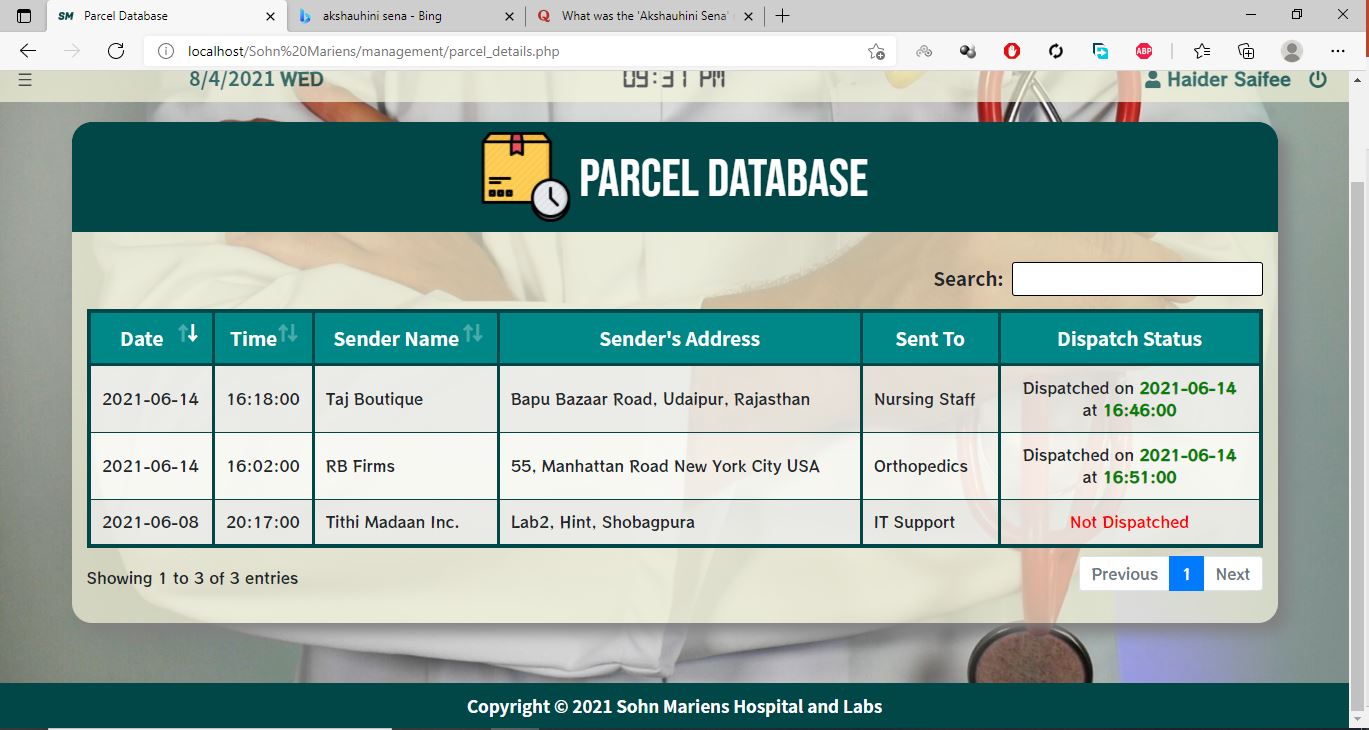


Image 4.4.8: Parcel Database

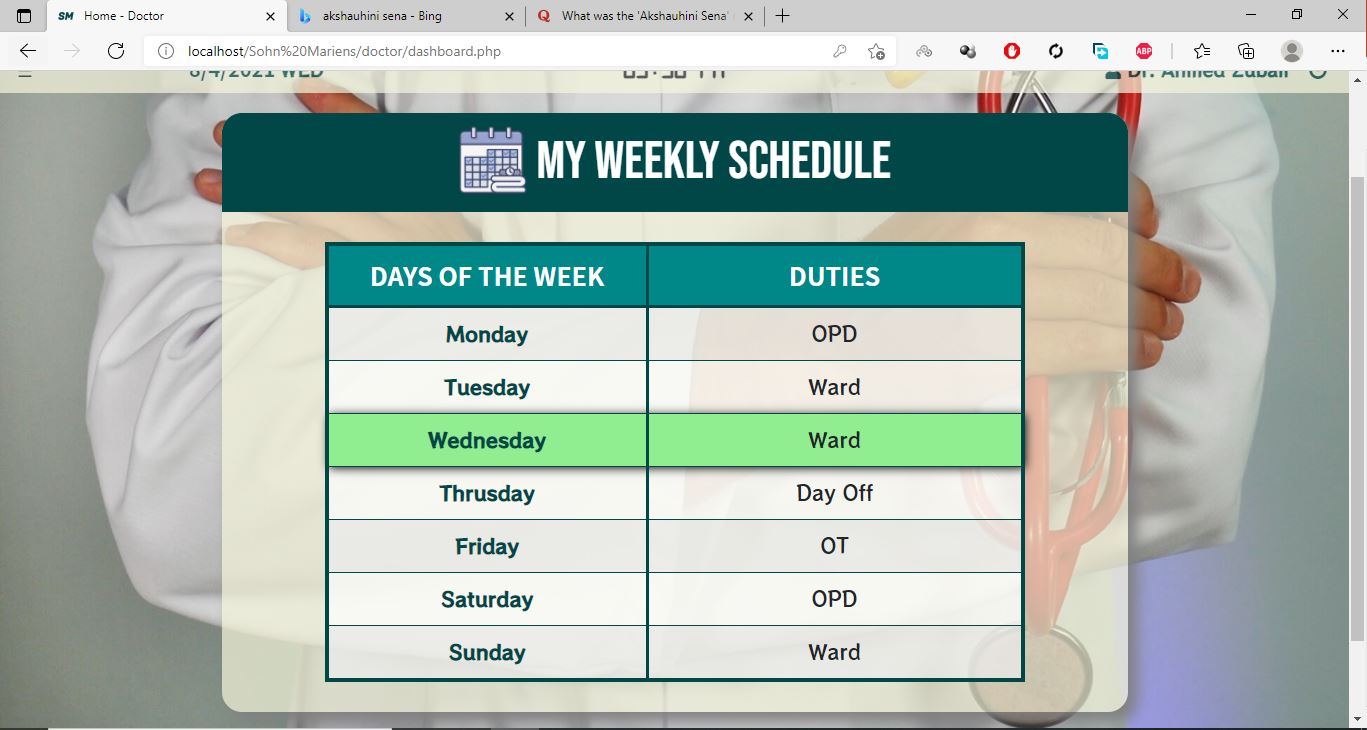


Image 4.4.9: Dashboard for doctors

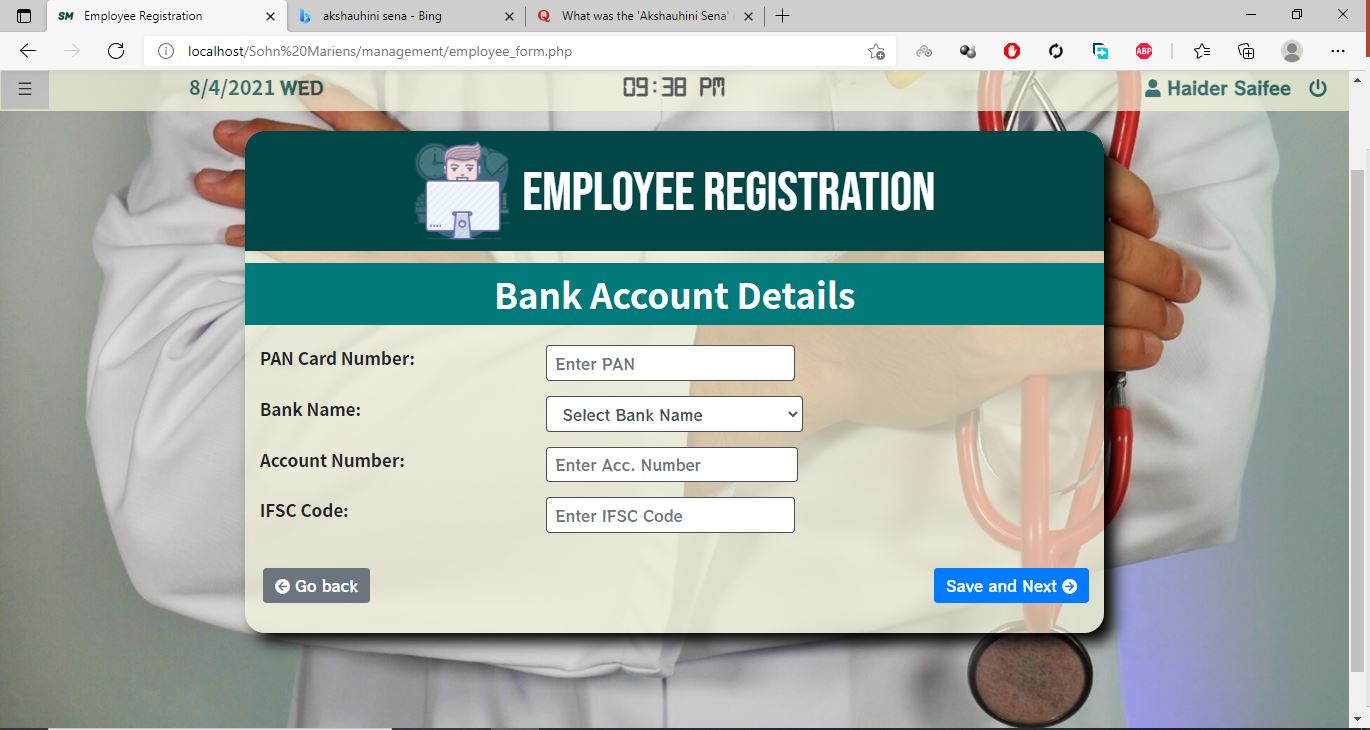


Image 4.4.10: Employee Registration Module

# CHAPTER 5 MECHANISMS USED

## Testing overview:

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. In this project “**White Box Testing**” was performed, which is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

### System testing:

Testing is an important part of software development. It is the process of finding errors and missing operations and also a complex verification to determine whether the objectives are met and the user requirements are satisfied. Following testing was carried out:

### Unit Testing:

The first includes unit testing, where each module is tested to provide its correctness, validity and also determine any missing operations and to verify whether the objectives have been met. Errors are noted down and corrected immediately. Unit testing is the important and major part of the project, so errors are rectified easily in a particular module and program clarity is increased. Our project is divided into several modules and is developed individually which helped in conducting unit testing.

### Integration Testing:

The second step includes integration testing. It needs not to be the case, the software whose modulus run individually and showing perfect results will also show perfect results when running as a whole. The individual modules are clipped under this major module and tested again verified the results. This is due to poor interfacing which may result in data being lost across an interface. A module can have an inadvertent, adverse effect on any other or on the global data structures, causing serious problems**.**

### Validation:

The final step involves validation and testing which determines whether the software functions as the user expected.

* + - * 1. Client-Side Validation:

Various client-side validations are used to ensure in the client side that only valid data is entered. A client-side validation saves time and load to handle invalid data. Some checks imposed are:

The script is used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.

Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time load

Tab-indexes are set according to the need and taking into account the ease of user, while working with the system.

* + - * 1. Server-Side Validation

Some checks cannot be applied on the client-side. Server-side checks are necessary to save the system from failing and intimating the user that some invalid operations have been performed or the performed operation is restricted. Some of the server checks imposed are:

Server side constraints have been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated.

The user is intimated through messages about the successful operations or exceptions occurring at the server-side.

Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on the system and can have access according to their category. User-name, passwords, and permissions are controlled the server side.

# CHAPTER 6 IMPLEMENTATION

## Implementation mechanisms used:

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage is achieving a new successful system and in giving confidence in the new system for the users that it works efficiently and effectively.

The implementation phase comprises of several activities. In the first phase, the required hardware and software acquisition was carried out. Implementing a hospital management system involved careful planning, investigation of the current system and its constraints on implementation, the design of methods to achieve the required features and facilities and the evaluation of methods apart from planning.

After which the initial draft of the user dashboards and secondary information tables were created were generated. After knowing the design of our project we looked up for the technologies that could help us in creating a software which would be functional in a hospital system.

Once we decided the technologies to be used then we created a data flow diagram up to level one and we also made a sitemap so as to keep a track of the pages we were going to work on.

Once all the things were planned out we started their implementation and in the process, we did the following things:

* + 1. **Patient Registration Module:**

When a patient arrives in the hospital, this module is used to enter the details of the patients in the hospital database. This has been achieved by using HTML form elements to create a form, post request is used to take the entered content from the client side to the server. Moreover, JavaScript and JQuery are used to create validation in the registration panel so that only verified information can be entered.

* + 1. **Appointment Creation Module:**

In this module, the patients’ appointments can be booked. Validating doctor’s schedule with date, time and appointment availability have been made sure using JQuery to ensure that no clerical errors can be made by the operator.

* + 1. **Appointment Handler:**

The reception users are provided with tabular details of the timing, status and other finer details of the patient appointments. The data-tables are used to provides filters and search bars so that the users can easily search and find the desired record.

* + 1. **Prescription Creator Module:**

The users created for the doctors at the hospital are provided with the facility to create, edit and print prescription and diagnostic slips for their patients.

* + 1. **Automatic Attendance system:**

The management and IT department users are provided with a system that records the login status as well as time of each user(employee).

* + 1. **Responsiveness:**

The responsiveness of the web application for multiple devices is achieved through the bootstrap library. Pages are designed with “row” and “col” class that helps to imitate a tabular design pattern which maintains the page design gracefully in smaller devices.

* + 1. **Employee Registration**

In very module, the users of the management department are given the feature to create edit and delete the personal, contact, education and other professional details of the employees of the hospital.

# CHAPTER 7 LIMITATION

## Limitations:

After thorough testing, some drawbacks were found that were not working according to the specifications. They are:

* + - * 1. We were not able to include the financial and stock handling side of the hospital in our system.
        2. The application does not have an independent payment gateway.
        3. The facility to keep a record of all the medicines and drugs administered while the patient is admitted in the wards have not been included in the management system.

# CHAPTER 8 CONCLUSION

## 8.1. Conclusion:

The main purpose of the project is to make a system that could effectively manage a hospital of any size from a multi-specialty hospital to a small clinic with less or no training in operating the system.

Another important reason to introduce HMS was to solve the complications coming from managing all the paper works of every patient associated with the various departments of hospitalization with confidentiality.

The project was developed with a modular approach. All the modules in the system have been tested valid data and invalid data and everything works successfully. Thus the system has fulfilled all the objectives identified and is able to replace the existing system. The constraints are met and overcome successfully. The system is designed as it was decided in the designing phase.

# CHAPTER 9 FUTURE SCOPE

## 9.1. Future scope:

* + - * 1. The software will be able to include a client for mail server so that communication for the employees could be eased.
        2. The admission, discharge and appointment features will be accompanied by bill creating software.
        3. An independent payment gateway will be included in the software.