**ONLINE DOCTOR APPOINTMENT**

**A Mini-Project Report**

Submitted

By

T.Amrutha

168w1f0057

**Under the esteemed guidance of**

**Mr. R. Madhukanth M.C.A,Mtech.,**

**Assistant Professor**



**DEPARTMENT OF COMPUTER APPLICATIONS**

**V.R.SIDDHARTHA ENGINEERING COLLEGE**

***(Autonomous)***

**KANURU, VIJAYAWADA-520007**

**ANDHRA PRADESH**

**2017-2018**

**INDEX**

|  |  |
| --- | --- |
| **Content** | **Page No.** |
| **Abstract** | **I** |
| **Chapter 1: Introduction** |  |
| 1.1. Background | 01 |
| 1.2. Objectives | 01 |
| 1.3. Purpose, Scope and Applicability | 01 |
| 1.3.1. Purpose | 01 |
| 1.3.2. Scope | 01 |
| **Chapter 2: System Analysis** |  |
| 2.1. Existing System | 02 |
| 2.2. Proposed System | 02 |
| 2.3. Requirement Analysis | 02 |
| 2.3.1. Problem Definition | 02 |
| 2.3.2. Requirement Specification | 02 |
| 2.3.3. Planning and Scheduling | 03 |
| 2.4. Hardware Requirements | 04 |
| 2.5. Software Requirements | 04 |
| 2.6. Justification of selection of Technology | 04 |
|  |  |
| **Chapter 3: System Analysis** |  |
| 3.1. Module Division | 14 |
| 3.2. Data Dictionary | 15 |
| 3.3. E-R Diagrams | 17 |
| 3.4. UML Diagrams | 19 |
| 3.4.1. Use case Diagram | 20 |
| 3.4.2. Class Diagram | 21 |
| 3.4.3. Activity Diagram | 22 |
| 3.4.4. Sequence Diagram | 23 |
| **Chapter 4: Implementation and Testing** |  |
| 4.1. Code | 24 |
| 4.2. Testing | 51 |
| 4.2.1. Unit Testing(Test cases and Test Results) | 51 |
| 4.2.2. Integration Testing(Test cases and Test Results) | 52 |
| **Chapter 5: Results and Discussions** | 53 |
| **Chapter 6: Conclusion and Future Work** | 62 |
| **References** | 63 |

**ABSTRACT**

Online Doctor appointment is a smart web application, this provides a registration and login for both doctors and patients. He can also view the feedback given by the patient. The patients must be registered and log in to book a doctor basing the category and the type of problem patient is facing and the location. The search results will show the list of doctors matching patients required criteria and he can select one and send a request the request will be forwarded to admin and admin forward to doctor and if he is available he will send the confirmation request back to admin the admin update the booking request and says confirmed to the patient. the patient can view the status in the status tab and also he will get the mail saying the booking is Confirmed.

**Chapter-1**

**Introduction**

* 1. **Background**

The project “Online doctor appointment system” is a smart appointment booking system that provides patients or any user an easy way of booking a doctor’s appointment online.

This is a web based application that overcomes the issue of managing and booking appointments according to user’s choice.

**1.2. Objectives**

The objective of the project on Doctor Appointment System is to manage the details of Doctor,Appointment,Patient,Booking. It manages all the information about doctor,medicine.

The project is totally built at administrative end and thus only the administrator is guarantedthe access.

**1.3. Purpose, Scope**

**1.3.1. Purpose**

* The purpose of the project is build an application program to reduce the manual work for managing the Doctor,Apointment,Patient.
* This system helps to reduce the waiting time of the patient.

**1.3.2. Scope**

* The scope of this project includes all online patient activities and interactions.
* This application can easily implemented under various situations.
* Reusability of this application is also possible.
* Have a good user interface.
* Be easy to operate.

**Chapter 2**

**System Analysis**

In system analysis, the main focus is researching the business process and current systems to fully understand the issues, model the existing processes and suggest process improvements. Once these items are completed, the physical design of the system takes place.

**2.1. Existing System**

In the existing system the patient needs to visit the doctor for booking we need to wait and the booking will be done manually so to maintain everything is always a problem.

* 1. **. Proposed System**

In the proposed system the doctors patients are brought to one platform will allow patients to be more flexible they can register and search for the doctors basing on the location the list of doctors will be shown and patient can book by selecting  the time slots and the admin will confirm the booking so everything is computerized an done very fast which will save time.

* 1. **. Requirement Analysis**

**2.3.1. Problem Definition**

The aim of “Online Doctor Appointment System” that provides patients or any user an easy way of booking a doctor’s appointment online.

**2.3.2. Requirement Specification**

The following are the requirements of the project Online Cab Booking System.

* Study about Online Doctor Apointment System.
* Understanding the need of people.
* Collecting the variety of Cabs.

**2.3.3. Planning and Scheduling**

Project planning spans across various aspects of application. Generally project planning is considered to be a process of estimating, scheduling and assigning the projects resources in order to deliver an end product of suitable quality. However it is much more as it can assume a very strategic role, which can determine the very success of the project. A project plan is one of the crucial steps in Project Planning in general.

**Schedule Development**

The time schedule for the project can be arrived at based on the activities, interdependence and effort required for each of them. The schedule may influence the cost estimates, the cost benefit analysis and so on.

Project Scheduling is one of the most important task of Project Planning and also the most difficult tasks. In very large projects it is possible that several teams work on developing the project. They may work on it in parallel. However their work may be interdependent.

**2.4. Hardware Requirements**

* **Processor :** Intel I3
* **Speed :** 700 MHZ
* **RAM Capacity :** 8GM
* **Hard Disk :** 60GB

**2.5. Software Requirements**

* **Operating System :** Windows8
* **Front-end :** HTML and CSS.
* **Server-side scripting :** php
* **IDE :**Xampp
* **Back-end :** MySql
* **Compatible browsers :** Chrome 5.X onwards.

**2.6. Justification of selection of Technology**

There are several technologies to do this project

**PHP**

PHP is an interpreted scripting language that is embedded within an HTML web page in order to add dynamic processing to that page.

PHP is supported by a wide range of commercial and open-source web servers, including RedHat Linux, and can also be installed as an Apache module. Its widespread availability and its relative simplicity mean that it is an excellent way to introduce dynamic features into your web pages. As it is an open, non-proprietary standard, PHP developers are not restricted by the limitations imposed by some commercial suppliers of server-side scripting software, neither do they have to purchase expensive licenses in order to use it.

PHP can be used to do anything that any CGI program can do, such as:

* Collect and process form data
* Generate dynamic page content
* Send and receive cookies

**<html>**

**<head>**

**<title>Example</title>**

**</head>**

**<body>**

**<?php echo "Hello, World!"; ? >**

**</body>**

**</html>**

**PHP Variables**

In PHP a variable is prefixed by a $ sign. A variable does not need formal declaration, nor do variables need to have declared types. A variable can even change its meaning and type during its lifetime.

The following example uses a string variable called **$greeting**, which is assigned with the value "Hello". If you are familiar with C or Perl then it should be fairly easy to work out what this example does.

**<?php**

**$greeting="Hello";**

**echo $greeting ", World!";**

**?>**

**PHP Functions**

As it is a structured programming language, PHP allows you to organize your script by the use of functions. A function is defined using the keyword 'function', followed by the name of the function and an argument list in parenthesis. The body of the function is enclosed in curly brackets, exactly like a C function. For example:

**function example($variable1, $variable2) {**

**echo "Example function\n" ;**

**return $retval ;**

**}**

Other than the 'function' keyword, this syntax will be familiar to anyone who is familiar with C functions.There is also a library of built-in functions available. These give you the ability to perform tasks such as:

* setting cookies
* send mail
* perform FTP operations
* interface with the server's filesystem
* interface with databases
* execute other programs on the server

There are also freely downloadable libraries available that also allow you to carry out the following:

* Manipulate images
* Create Shockwave flash files
* Create PDF documents
* Perform encryption/decryption

**HTML Forms and PHP**

When an HTML form is submitted to a PHP script, the variables from that form are automatically be made available to the script by PHP. Therefore, if a user enters data into a named input field, a variable with that name becomes available to the PHP script, already initialized with the value entered by the user.

The following example illustrates this:

**<FORM METHOD="GET" ACTION="submit.php">**

**What's your name? <INPUT NAME="name" SIZE=3>**

**</FORM>**

This fragment of HTML merely creates a form with one data entry field called "name". When the user enters some text into the name field and the form is submitted, the server calls the script submit.php with the variable $name already initialized. The following snippet of PHP code shows how this variable can be used.

**<?php**

**echo "Hello, $name!";**

**?>**

**Why PHP?**

PHP stands for Hypertext Preprocessor. It is a server-side scripting language that powers some of the most popular websites in the world, including WordPress and Facebook. It is open source, relatively easy to learn, and works perfectly with MySQL, making it a popular choice for web developer.

* PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
* PHP is compatible with almost all servers used today (Apache, IIS, etc.)
* PHP supports a wide range of databases
* PHP is free. Download it from the official PHP resource:
* PHP is easy to learn and runs efficiently on the server side

**Why CSS?**

Cascading Style Sheets or CSS are an important way to control how your Web pages look. CSS can control the fonts, text, colors, backgrounds, margins, layout and variations in display for different devices and screen sizes.

**Internal style sheets:**

These are styles that are placed in the <head> of the HTML document. These styles affect only the document they are in, and cannot be referenced by any other Web document. They allow you to test the styles in the context of the entire site without breaking any page but the one you are testing. This is especially useful when you're working with things like email where you have only one HTML file.

**External Style Sheets:**

The external style sheet is generally used when you want to make changes on multiple pages. It is ideal for this condition because it facilitates you to change the look of the entire web site by changing just one file .It uses the <link> tag on every pages and the <link> tag should be put inside the head section.

<head>

<title></title>

<style type=”text/css”>

CssContent Goes Here

</style>

</head>

<body><link rel=”stylesheet” type=”text/css” href=”Path to stylesheet.css”/>

**Reduced File Size Web Server: localhost**

"The **localhost** is the default name describing the local computer address also known as the **loopback address**. For example, typing: **ping localhost** would ping the local IP address of 127.0.0.1 (the loopback address). When setting up a web server or software on a web server, 127.0.0.1 is used to point the software to the local machine."

**Why HTML?**

**HTML is easy to use and understand**

Almost anyone in the web development business would know HTML – be it a freelancer or a large agency. If at any point in time you need to hire the services of a different web design firm or professional for making changes or updates to your website, it would be relatively easy to find cost-effective and affordable solution providers who can make the changes you need to your website.

**All browsers support HTML**

Almost – if not all – browsers support HTML. Certainly more browsers support HTML than any other web programming language. As a result, when you build a website using HTML, it would show up on most browsers around the world, as long as the programmer takes care to optimize the website for the most commonly used browsers. Optimizing an HTML based website for browser compatibility is neither difficult nor complex.

**Most development tools support HTML**

Whether it is FrontPage, Dreamweaver or any other programming tool, thereare more web development tools that allow you to create HTML based websites, than any other web programming language.

**HTML Forms**

Creating a form usually involves two independent steps : creating the layout for the form itself and then writing a script program on the server side to process the formation you get back from a form.

To create a form, we can use <FORM> tag. Inside the opening and closing form tag are each of the individual form elements plus any other html content to create a layout for that form. The opening tag of the form element usually includes the attributes: METHOD and ACTION. ----The METHODattribute can either GET/POST which determines how the form data is sent to the script to process. ---The ACTION attribute is a pointer to the script that processes the form on the server side. The action can be included by a relative path or by a full URL to a script on your server or somewhere else.

**Example :** <form method=’GET’ action=’http://localhost:4040/sobha/customer/login.jsp’>

……………… </form>

GET METHOD:

* Appends form-data into the URL in name/value pairs
* The length of a URL is limited (about 3000 characters)
* Never use GET to send sensitive data! (will be visible in the URL)
* Useful for form submissions where a user want to bookmark the result
* GET is better for non-secure data, like query strings in Google

POST METHOD:

* Appends form-data inside the body of the HTTP request (data is not shown is in URL)
* Has no size limitations
* Form submissions with POST cannot be bookmarked

**Why Xammp?**

XAMPP stands for Cross-Platform (X), Apache (A), MySQL (M), PHP (P) and Perl (P). It is a simple, lightweight Apache distribution that makes it extremely easy for developers to create a local web server for testing purposes. Everything you need to set up a web server – server application (Apache), database (MySQL), and scripting language (PHP) – is included in a simple extractable file. XAMPP is also cross-platform, which means it works equally well on Linux, Mac and Windows. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server is extremely easy as well. Web development using XAMPP is especially beginner friendly.

**MySQL**

MySQL is an [open-source](https://en.wikipedia.org/wiki/Open-source) [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS).  In July 2013, it was the world's second most widely used RDBMS, and the most widely used open-source [client–server model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model) RDBMS. It is named after [Michael Widenius](https://en.wikipedia.org/wiki/Michael_Widenius)' (who is a co-founder of MySQL) daughter, My, while "[SQL](https://en.wikipedia.org/wiki/SQL)" stands as the abbreviation for [Structured Query Language](https://en.wikipedia.org/wiki/Structured_Query_Language). MySQL was created by a Swedish company, MySQL AB, founded by [David Axmark](https://en.wikipedia.org/wiki/David_Axmark), Allan Larsson and [Michael "Monty" Widenius](https://en.wikipedia.org/wiki/Michael_(Monty)_Widenius). The first version of MySQL appeared on 23 May 1995. MySQL provides a implementation of a SQL database very well suited for small to medium web pages. The database is free and open source with a commercial license available (MySQL is now owned by Oracle after they bought Sun).  
Common applications for MySQL include php and java based web applications that require a DB storage backend, e.g. Dokuwiki, Joomla, xwiki etc. Very many applications that use MySQL are geared towards the LAMP stack.

**Features and Advantages**

**Relational Database System:**

Like almost all other database systems on the market, MySQL is a relational database system.

**Client/Server Architecture:**

MySQL is a client/server system. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc. The clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

**SQL compatibility:**

MySQL supports as its database language -- as its name suggests – SQL (Structured Query Language). SQL is a standardized language for querying and updating data and for the administration of a database. There are several SQL dialects (about as many as there are database systems). MySQL adheres to the current SQL standard (at the moment SQL:2003), although with significant restrictions and a large number of extensions.

**SubSELECTs:**

Since version 4.1, MySQL is capable of processing a query in the form SELECT \* FROM table1 WHERE x IN (SELECT y FROM table2) (There are also numerous syntax variants for subSELECTs.)

**Views:**

Put simply, views relate to an SQL query that is viewed as a distinct database object and makes possible a particular view of the database. MySQL has supported views since version 5.0.

**Stored procedures:**

Here we are dealing with SQL code that is stored in the database system.Stored procedures (SPs for short) are generally used to simplify certain steps, such as inserting or deleting a data record. For client programmers this has the advantage that they do not have to process the tables directly, but can rely on SPs.

**Triggers:**

Triggers are SQL commands that are automatically executed by the server in certain database operations (INSERT, UPDATE, and DELETE).MySQL has supported triggers in a limited form from version 5.0, and additional functionality is promised for version 5.1.

**Unicode:**

MySQL has supported all conceivable character sets since version 4.1, including Latin-1, Latin-2, and Unicode (either in the variant UTF8 or UCS2).

**Full-text search:**

Full-text search simplifies and accelerates the search for words that are located within a text field. If you employ MySQL for storing text (such as in an Internet discussion group), you can use full-text search to implement simply an efficient search function.

**Replication:**

Replication allows the contents of a database to be copied (replicated) onto a number of computers. In practice, this is done for two reasons: to increase protection against system failure (so that if one computer goes down, another can be put into service) and to improve the speed of database queries.

**Transactions:**

In the context of a database system, a transaction means the execution of several database operations as a block. The database system ensures that either all of the operations are correctly executed or none of them. This holds even if in the middle of a transaction there is a power failure, the computer crashes, or some other disaster occurs. Thus, for example, it cannot occur that a sum of money is withdrawn from account A but fails to be deposited in account B due to some type of system error.

**GIS functions:**

Since version 4.1, MySQL has supported the storing and processing of two-dimensional geographical data. Thus MySQL is well suited for GIS (geographic information systems) applications.

**ODBC:**

MySQL supports the ODBC interface Connector/ODBC. This allows MySQL to be addressed by all the usual programming languages that run under Microsoft Windows (Delphi, Visual Basic, etc.). The ODBC interface can also be implemented under Unix, though that is seldom necessary.

**Platform independence:**

It is not only client applications that run under a variety of operating systems; MySQL itself (that is, the server) can be executed under a number of operating systems. The most important are Apple Macintosh OS X, Linux, Microsoft Windows, and the countless Unix variants, such as AIX, BSDI, FreeBSD, HP-UX, OpenBSD, Net BSD, SGI Iris, and Sun Solaris.

**Speed:**

MySQL is considered a very fast database program. This speed has been backed up by a large number of benchmark

**Advantages of using MYSQL**

**It’s Easy To Use:**

MySQL is very easy to install, and thanks to a bevy of third-party tools that can be added to the database, setting up an implementation is a relatively simple task. In addition, it’s also an easy database to work with. So long as you understand the language, you shouldn’t run into too many problems.

**Support Is Readily Available Whenever Necessary:**

Although Oracle’s history of supporting its customers can be spotty at best, the nature of MySQL – which got its start as an open-source platform – means that there’s a large and thriving community of developers and enthusiasts to which one can turn for help. This is due in large part to the popularity of the solution, the end result of which is no shortage of experts.

**It’s Open-Source (Sort Of):**

Oracle’s purchase of Sun Microsystems (and by association, MySQL) was [met with some contention from the development community](http://techcrunch.com/2012/08/18/oracle-makes-more-moves-to-kill-open-source-mysql/). The general fear was that Oracle would transform the tool into a closed, proprietary ecosystem. Thankfully, though Oracle has tightened its grip on MySQL somewhat, it can still be considered an open-source database option, as [the code is still available for free online](http://www.mysql.com/).

**Chapter 3**

**System Design**

**3.1. Module Division:**

This system contains two modules namely:

1. Admin Module

2. Patient Module

3.Doctor Module

**Description of Modules:**

In software engineering a module is a portion of a project that carries out a specific function and may be used alone or combined with other modules of the same project.

**1. Admin Module:**

Admin needs to login with username and password and in the admin home screen, he can see the basic functionalities of admin. Admin can view the registered doctors and patients. He can also view the patient’s request and doctors requests and he will confirm the patients and doctors requests.

**2. Patient Module:**

The patient needs to be registered and log in after logging on he can search for the doctor by giving the location, the reason or problem. Basing on the doctor availability the admin will confirm the booking request and will send to mail that the booking is confirmed he can also view in the status and he can also give feedback basing the performance of the doctor.

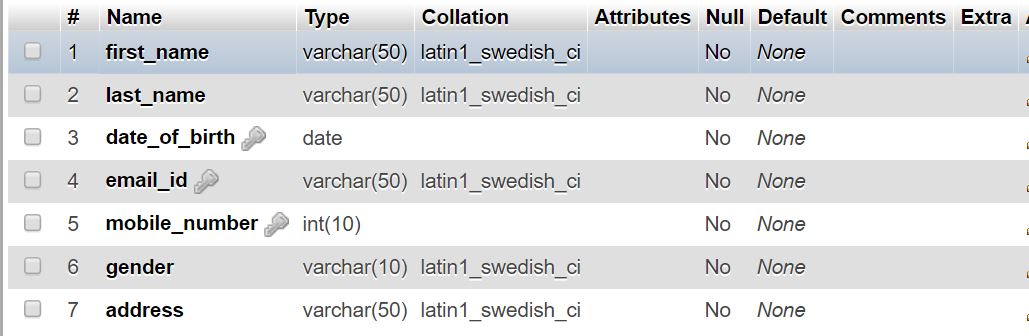
**3.Doctor Module:**

Doctor need to be registered by giving the necessary details like experience, timing, fees etc. After registering he need to log in and in the home screen he can view the basic functionalities. He can view the patient request forwarded from admin and he can accept and he can also view the feedback given by patients.

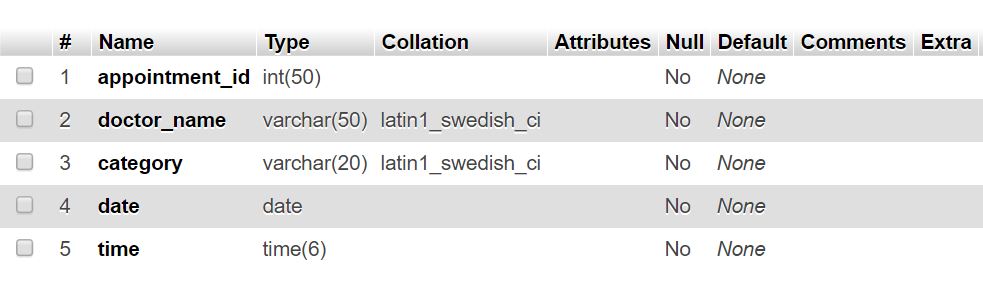
* 1. **Data Design**

**3.2.1. Data Dictionary**

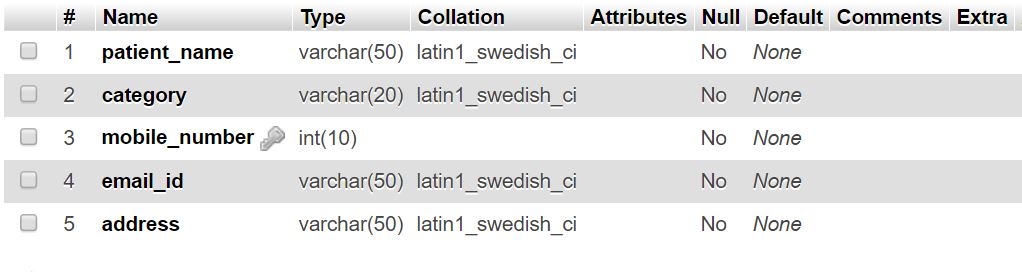
**Table 3.1:** Description of Registration

****

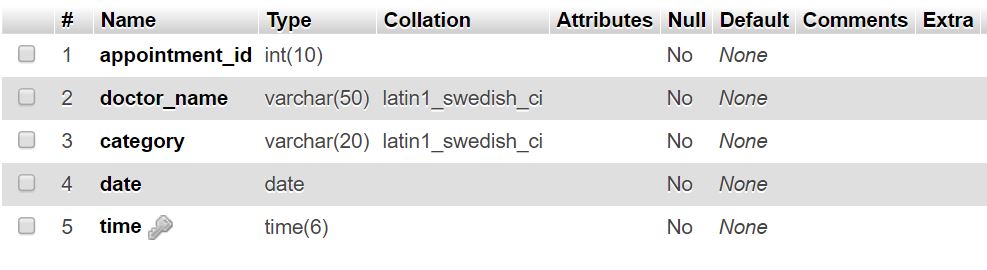
**Table 3.2:** Description of Adding Doctors



**Table 3.3:** Description of Adding Patients

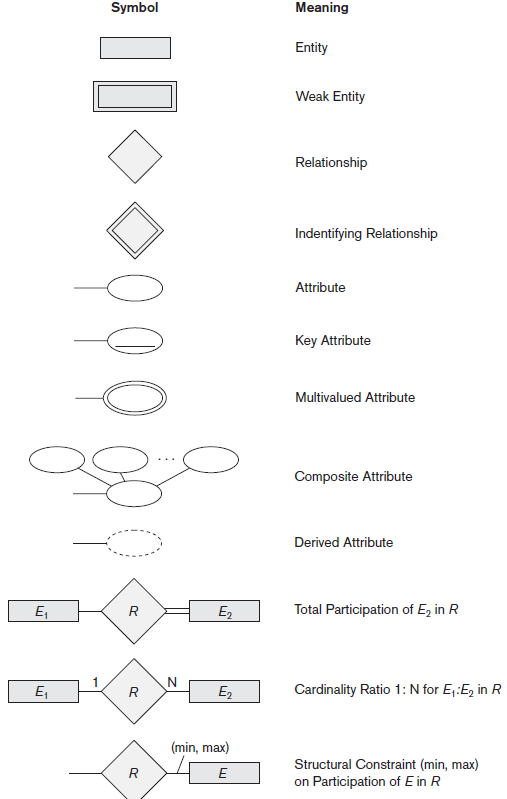


**Table 3.4:** Description of Booking Appointment



**3.5. E-R Diagrams**

This type of diagramming is used to show relationships between objects in a system. A relational database can have several different types of relationships, depending on the needs that the database is to fulfill. An E-R Diagram graphically models those relationships as shown in the below Figures.

****

**Fig. 3.1:** Summary of the notation for E-R diagrams

## Entity:

## Entities are represented by means of rectangles. Rectangles are named with the entity set they represent

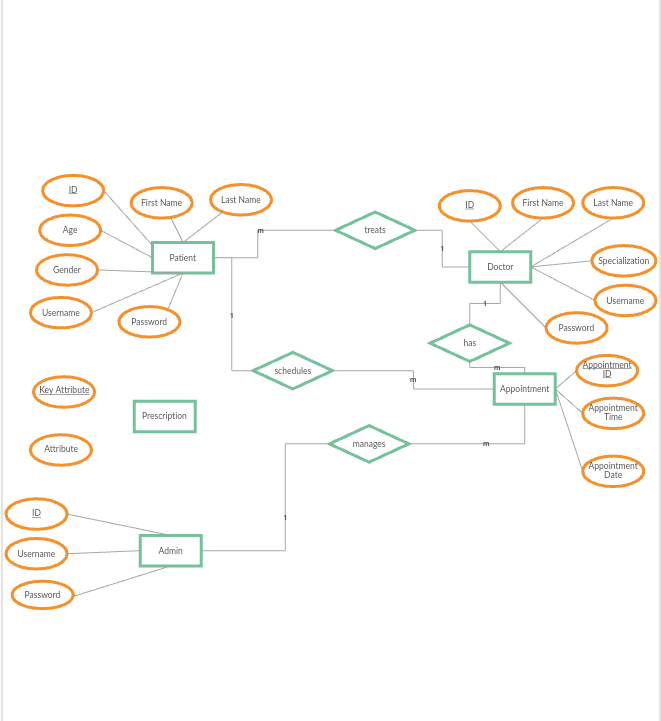
**Attributes:**

Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity.

**Constraints**:

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints**. There are three main integrity constraints

* Key constraints
* Domain constraints
* Referential integrity constraints



**Fig. 3.2:** E-R diagram

**3.6.UML Diagrams**

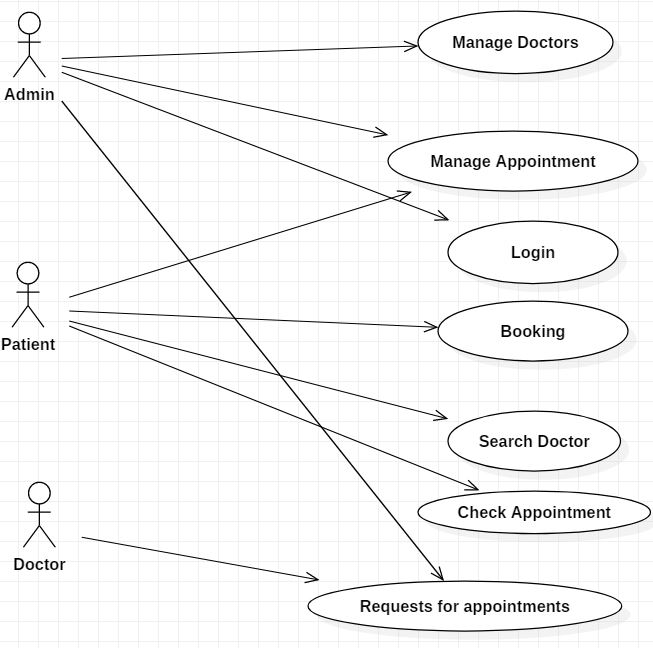
* UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.
* The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation.
* The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.
* The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.
* The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**Goals:**

The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**3.4.1. Use Case Diagram:**

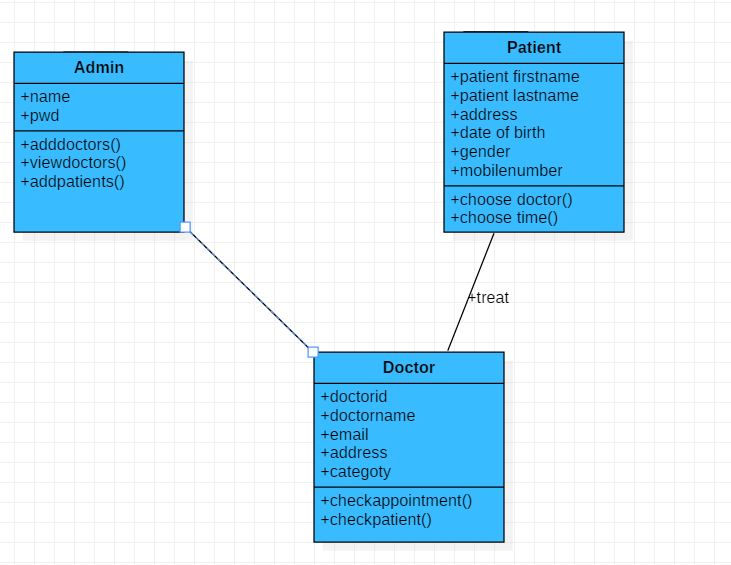
****

**Fig. 3.3:** Use case diagram

**Description:**

The above Fig 3.3: use case diagram contains three users namely admin, patient,doctor. The Admin maintain the manage appointment,manage doctors,login.Patient maintain the login,booking,search doctor.

**3.4.2. Class Diagram:**

****

**Fig. 3.4:** Class Diagram

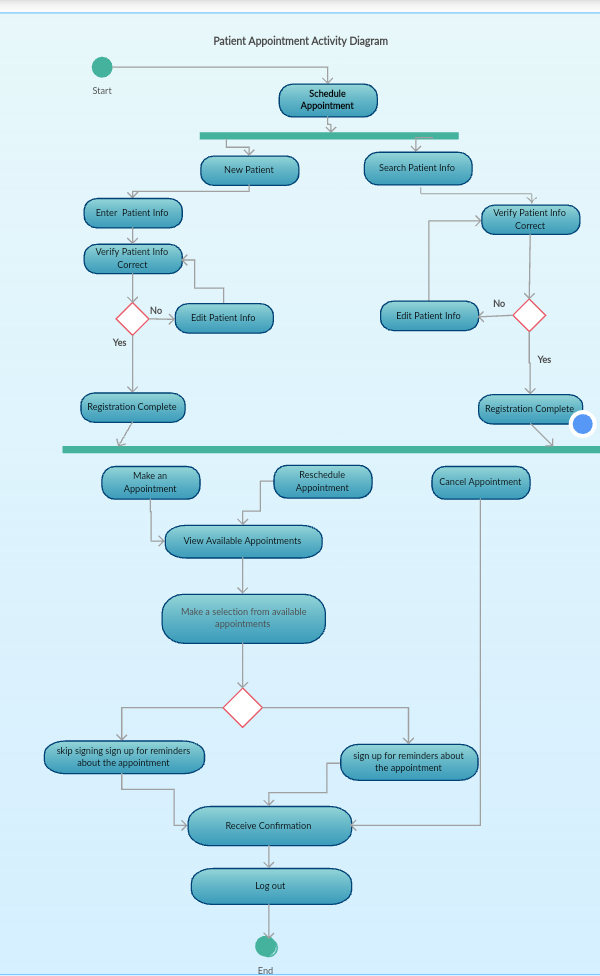
**Description:**

The above Fig. 3.4: class diagram is for the purpose to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

So the purpose of the class diagram can be summarized as:

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Attribute values may vary between instances.

**3.4.3. Activity Diagram:**

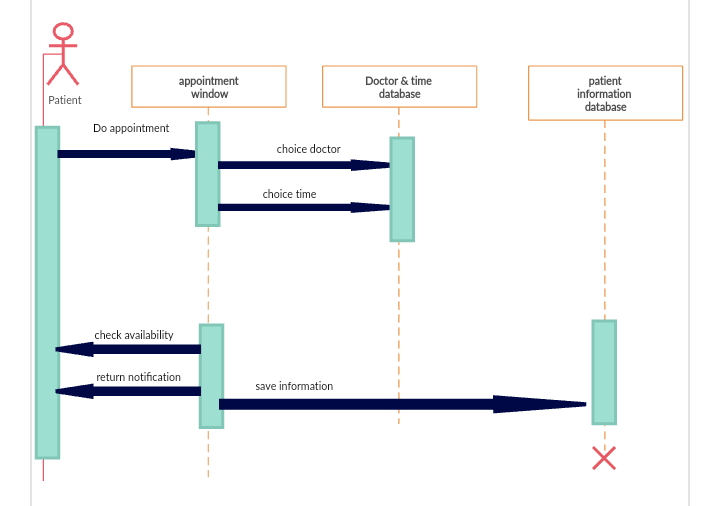
****

**Fig. 3.5:** Activity Diagram

**Description:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

**3.4.5. Sequence Diagram**

****

**Fig. 3.8:** Sequence Diagram

**Description:**

The above Sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom. The interaction starts near the top of the diagram and ends at the bottom.

**Chapter 4**

**Implementation and Testing**

**4.1. Code**

**reg.html:**

<!DOCTYPE html>

<html>

<head>

<title>Patient Registration Form</title>

<style>

body {

background-color: lightblue;

background-image: url('hos3.jpg');

background-position: fixed;

background-size: cover;

background-repeat: no-repeat;

}

h1 {

color: white;

text-align: center;

}

p {

font-family: verdana;

font-size: 20px;

}

</style>

<body>

<div class="alert">

</div>

<form action="patiregister.php" method="post">

</head>

<center>

<h3>PATIENT REGISTRATION FORM</h3>

</center>

<table align="center" cellpadding = "10">

<!----- First Name ---------------------------------------------------------->

<tr>

<td>FIRST NAME</td>

<td><input type="text" name="first\_name" maxlength="30" required/>

</td>

</tr>

<!----- Last Name ---------------------------------------------------------->

<tr>

<td>LAST NAME</td>

<td><input type="text" name="last\_name" maxlength="30" required/>

</td>

</tr>

<!----- Date Of Birth -------------------------------------------------------->

<tr>

<td>DATE OF BIRTH</td>

<td><input type="date" name="date\_of\_birth" maxlength="30" required /></td>

</tr>

<!----- Email Id ---------------------------------------------------------->

<tr>

<td>EMAIL ID</td>

<td><input type="text" name="email\_id" maxlength="100" required/></td>

</tr>

<!----- Mobile Number ---------------------------------------------------------->

<tr>

<td>MOBILE NUMBER</td>

<td>

<input type="text" name="mobile\_number" maxlength="10" required />

</td>

</tr>

<!----- Category ---------------------------------------------------------->

<tr>

<td>CATEGORY</td>

<td><select name="category">

<option value="-1" selected>select..</option>

<option value="Fever">Fever</option>

<option value="Bodypains">Bodypains</option>

<option value="Heart">Cardiaologist</option>

<option value="Cold">Dentist</option>

<option value="Cold">Dermatologist</option>required

</select></td>

</tr>

<!----- Gender ----------------------------------------------------------->

<tr>

<td>GENDER</td>

<td>

Male <input type="radio" name="gender" value="Male" />

Female <input type="radio" name="gender" value="Female" required/>

</td>

</tr>

<tr>

<td>ADDRESS <br /><br /><br /></td>

<td><textarea name="address" rows="4" cols="30"></textarea></td>

</tr>

<tr>

<td colspan="2" align="center">

<input type="submit" value="Register">

<input type="reset" value="Reset">

</td>

</tr>

</table>

</form>

<h1><a href ="patientlogin.html">Patient Login</a></h1>

</body>

</html>

**reg.php:**

<?php

$hostname="localhost";

$username="root";

$password="";

$databaseName="patiregistration";

$conn =mysqli\_connect("localhost", "root", "", "patiregistration");

$first\_name=$\_POST['first\_name'];

$last\_name=$\_POST['last\_name'];

$date\_of\_birth=$\_POST['date\_of\_birth'];

$email\_id=$\_POST['email\_id'];

$mobile\_number=$\_POST['mobile\_number'];

$gender=$\_POST['gender'];

$address=$\_POST['address'];

$sql ="INSERT INTO pregister(first\_name,last\_name,date\_of\_birth,email\_id,mobile\_number,gender,address)values('$first\_name','$last\_name','$date\_of\_birth','$email\_id','$mobile\_number','$gender','$address')";

if ($conn->query($sql) === TRUE) {

echo "Registration successfully";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

$conn->close();

?>

**Booking.html:**

**<**html>

**<**head>

<form action="booking.php" method="post">

</head>

<body >

<center>

<h1>Online Doctor Appointment Booking </h1><marquee>welcome to Online Doctor Appointment</marquee><br><br>

<div class="nav">

<ul>

<li><a href ="home.html">Home</a></li>

<li><a href ="booking.html"> Book Appointment</a></li>

<li><a href ="viewbooking.php"> View Booking</a></li>

<li><a href ="search.html">Search Doctor</a></li>

<li><a href ="logout.html"> Logout</a></li>

</ul>

</div>

<h1>Booking</h1>

<table align="center" cellpadding = "10">

<tr>

<td>APPOINTMENT ID</td>

<td><input type="text" name="appointment\_id" maxlength="30" required/>

</td>

</tr>

<tr>

<td>DOCTOR NAME</td>

<td><select name="doctor\_name">

<option value="-1" selected>select..</option>

<option value="Avanthi">Avanthi</option>

<option value="Srinivasa Rao">Srinivasa Rao</option>

<option value="Akhila">Akhila</option>

<option value="Neelima">Neelima</option>

</select></td>

</tr>

<tr>

<td>CATEGORY</td>

<td><select name="category">

<option value="-1" selected>select..</option>

<option value="Fever">Fever</option>

<option value="Bodypains">Bodypains</option>

<option value="Heart">Heart</option>

<option value="Cold">Cold</option>

</select></td>

</tr>

<tr>

<td>DATE</td>

<td><input type="date" name="date" maxlength="30" required/>

</td>

</tr>

<!----- Timings ---------------------------------------------------------->

<tr>

<td>TIMINGS</td>

<td><input type="time" name="time" maxlength="30" required/></td>

</tr>

</table>

<input type="submit" value="Book">

<input type="reset" value="Cancel">

</form>

</center>

</body>

</html>

**booking.php:**

<?php

$hostname="localhost";

$username="root";

$password="";

$databaseName="booking";

$appointment\_id=$\_POST['appointment\_id'];

$doctor\_name=$\_POST['doctor\_name'];

$category=$\_POST['category'];

$date = $\_POST['date'];

$time=$\_POST['time'];

$conn =mysqli\_connect("localhost", "root", "", "booking");

$sql ="INSERT INTO book(appointment\_id,category,doctor\_name,date,time)values('$appointment\_id','$category','$doctor\_name','$date','$time')";

if ($conn->query($sql) === TRUE) {

echo "Registered successfully";

} else

echo "The Time aleready booking";

$conn->close();

?>

**Viewbooking.html:**

<html>

<head>

<form action="viewbooking.php" method="post">

</head>

<body >

<center>

<h1>Online Doctor Appointment Booking </h1><marquee>welcome to Online Doctor Appointment</marquee><br><br>

<div class="nav">

<ul>

<li><a href ="home.html">Home</a></li>

<li><a href ="booking.html"> Book Appointment</a></li>

<li><a href ="viewbooking.php"> View Booking</a></li>

<li><a href ="search.html">Search Doctor</a></li>

<li><a href ="logout.html"> Logout</a></li>

</ul>

</div>

<h1>Booking</h1>

<table align="center" cellpadding = "10">

<tr>

<td>APPOINTMENT ID</td>

<td><input type="text" name="appointment\_id" maxlength="30" required/>

</td>

</tr>

<tr>

<td>DOCTOR NAME</td>

<td><select name="doctor\_name">

<option value="-1" selected>select..</option>

<option value="Avanthi">Avanthi</option>

<option value="Srinivasa Rao">Srinivasa Rao</option>

<option value="Akhila">Akhila</option>

<option value="Neelima">Neelima</option>

</select></td>

</tr>

<tr>

<td>CATEGORY</td>

<td><select name="category">

<option value="-1" selected>select..</option>

<option value="Fever">Fever</option>

<option value="Bodypains">Bodypains</option>

<option value="Heart">Heart</option>

<option value="Cold">Cold</option>

</select></td>

</tr>

<tr>

<td>DATE</td>

<td><input type="date" name="date" maxlength="30" required/>

</td>

</tr>

<!----- Timings ---------------------------------------------------------->

<tr>

<td>TIMINGS</td>

<td><input type="time" name="time" maxlength="30" required/></td>

</tr>

</table>

<input type="submit" value="Book">

<input type="reset" value="Cancel">

</form>

</center>

</body>

</html>

**Viewbooking.php:**

<?php

$link = mysqli\_connect("localhost", "root", "", "booking");

if ($link == false) {

die("ERROR: Could not connect. ".mysqli\_connect\_error());

}

$sql = "SELECT \* FROM book";

if ($res = mysqli\_query($link, $sql)) {

if (mysqli\_num\_rows($res) > 0) {

echo "<table border=2>";

echo "<tr>";

echo "<th>appointment\_id</th>";

echo "<th>doctor\_name</th>";

echo "<th>category </th>";

echo "<th>date</th>";

echo "<th>time</th>";

echo "</tr>";

while ($row = mysqli\_fetch\_array($res)) {

echo "<tr>";

echo "<td>".$row['appointment\_id']."</td>";

echo "<td>".$row['doctor\_name']."</td>";

echo "<td>".$row['category']."</td>";

echo "<td>".$row['date']."</td>";

echo "<td>".$row['time']."</td>";

echo "</tr>";

}

echo "</table>";

}

else {

echo "No matching records are found.";

}

}

else {

echo "ERROR: Could not able to execute $sql. ".mysqli\_error($link);

}

mysqli\_close($link);

?>

**Search.html:**

<html>

<head>

<form action="search.php" method="post">

</head>

<body >

<center>

<h1>Online Doctor Appointment Booking </h1><marquee>welcome to Online Doctor Appointment</marquee><br><br>

<div class="nav">

<ul>

<li><a href ="home.html">Home</a></li>

<li><a href ="booking.html"> Book Appointment</a></li>

<li><a href ="viewbooking.php"> View Booking</a></li>

<li><a href ="search.html">Search Doctor</a></li>

<li><a href ="logout.html"> Logout</a></li>

</ul>

</div>

<h1>Search</h1>

<table align="center" cellpadding = "10">

<tr>

<td>DOCTOR NAME</td>

<td><select name="doctor\_name">

<option value="-1" selected>select..</option>

<option value="Avanthi">Avanthi</option>

<option value="Srinivasa Rao">Srinivasa Rao</option>

<option value="Akhila">Akhila</option>

<option value="Neelima">Neelima</option>

</select></td>

</tr>

<tr>

<td>CATEGORY</td>

<td><select name="category">

<option value="-1" selected>select..</option>

<option value="Fever">Fever</option>

<option value="Bodypains">Bodypains</option>

<option value="Heart">Heart</option>

<option value="Cold">Cold</option>

</select></td>

</tr>

<tr>

</table>

<input type="submit" value="Search">

<input type="reset" value="Cancel">

</form>

</center>

</body>

</html>

**Search.php:**

<?php

$conn=mysqli\_connect("localhost","root","","booking");

$doctor\_name=$\_POST['doctor\_name'];

$category=$\_POST['category'];

$query="select \*from book where doctor\_name='$doctor\_name' AND category='$category'";

$result=$conn->query($query);

?>

<?php

if($result->num\_rows>0){

echo "<table border=2><tr><th>Appointment Id</th><th>doctor\_name</th><th>category</th><th>date</th>

<th>time</th></tr>";

while($row=$result->fetch\_assoc()){

echo "<tr>";

echo "<td>".$row['appointment\_id']."</td>";

echo "<td>".$row['doctor\_name']."</td>";

echo "<td>".$row['category']."</td>";

echo "<td>".$row['date']."</td>";

echo "<td>".$row['time']."</td>";

echo "</tr>";

}

echo "</table>";

}

else

echo "No matching records are found.";

?>

**myappointments.html:**

<html>

<head>

<form action="myappointments.php" method="post">

</head>

<body >

<center>

<h1>Online Doctor Appointment Booking </h1><marquee>welcome to Online Doctor Appointment</marquee><br><br>

<div class="nav">

<ul>

<li><a href ="home.html">Home</a></li>

<li><a href ="myappointments.php">My Appointments</a></li>

<li><a href ="viewpatients.php"> ViewPatients</a></li>

<li><a href ="logout.html"> Logout</a></li></h1>

</ul>

</div>

<center>

<h1>My appointments</h1>

</center>

<table align="center" cellpadding = "10">

<tr>

<td>ID</td>

<td><input type="text" name="id" maxlength="30" required/>

</td>

</tr>

<tr>

<td>DOCTOR NAME</td>

<td><select name="doctor\_name">

<option value="-1" selected>select..</option>

<option value="Avanthi">Avanthi</option>

<option value="Srinivasa Rao">Srinivasa Rao</option>

<option value="Akhila">Akhila</option>

<option value="Neelima">Neelima</option>

</select></td>

</tr>

<tr>

<td>CATEGORY</td>

<td><select name="category">

<option value="-1" selected>select..</option>

<option value="Fever">Fever</option>

<option value="Bodypains">Bodypains</option>

<option value="Heart">Heart</option>

<option value="Cold">Cold</option>

</select>

</td>

</tr>

<tr>

<td>DATE</td>

<td><input type="date" name="date" maxlength="30" required/>

</td>

</tr>

<!----- Timings ---------------------------------------------------------->

<tr>

<td>TIMINGS</td>

<td><input type="time" name="time" maxlength="30" required/></td>

</tr>

</table>

<input type="submit" value="Book">

<input type="reset" value="Cancel">

</form>

</center>

</body>

</html>

**myappointments.php:**

<?php

$link = mysqli\_connect("localhost", "root", "", "registration");

if ($link == false) {

die("ERROR: Could not connect. " .mysqli\_connect\_error());

}

$sql = "SELECT \* FROM Doctor";

if ($res = mysqli\_query($link, $sql)) {

if (mysqli\_num\_rows($res) > 0) {

echo "<table border=2>";

echo "<tr>";

echo "<th>id</th>";

echo "<th>doctor\_name</th>";

echo "<th>category </th>";

echo "<th>date</th>";

echo "<th>time</th>";

echo "</tr>";

while ($row = mysqli\_fetch\_array($res)) {

echo "<tr>";

echo "<td>".$row['id']."</td>";

echo "<td>".$row['doctor\_name']."</td>";

echo "<td>".$row['category']."</td>";

echo "<td>".$row['date']."</td>";

echo "<td>".$row['time']."<

echo "</tr>";

}

echo "</table>";

}

else {

echo "No matching records are found.";

}

}

else {

echo "ERROR: Could not able to execute $sql. " .mysqli\_error($link);

}

mysqli\_close($link);

?>

**4.2. Testing Approach**

**4.2.1Unit Testing**

Unit testing focuses verification on small unit of software. This is known as form testing. The testing is done individually on each form. Using the unit test plan, prepared in design phase of system development as s guide, important control paths are tested to uncover within the boundary of the module. In this step, the module is working satisfactory as a regard to the expected output from the module.

**4.2.2. Integration Testing**

Data can be lost across an interface, one module can have adverse effect on another

sub function, when combined, may not produce the desired major function.Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover error associated with the interface. All the modules are combined in the testing step. Then the entire program is as whole.

Different integrated test plans like top down integration and bottom up integration are tested and different errors found in the system are corrected using them. Finally, all the combined modules are performed well.

**Chapter 5**

**Results and Discussions**

**HOME**

****

**ADMIN LOGIN**



**ADMIN HOME PAGE:**



**ADD DOCTORS:**



**VIEW DOCTORS:**



**ADD PATIENTS:**



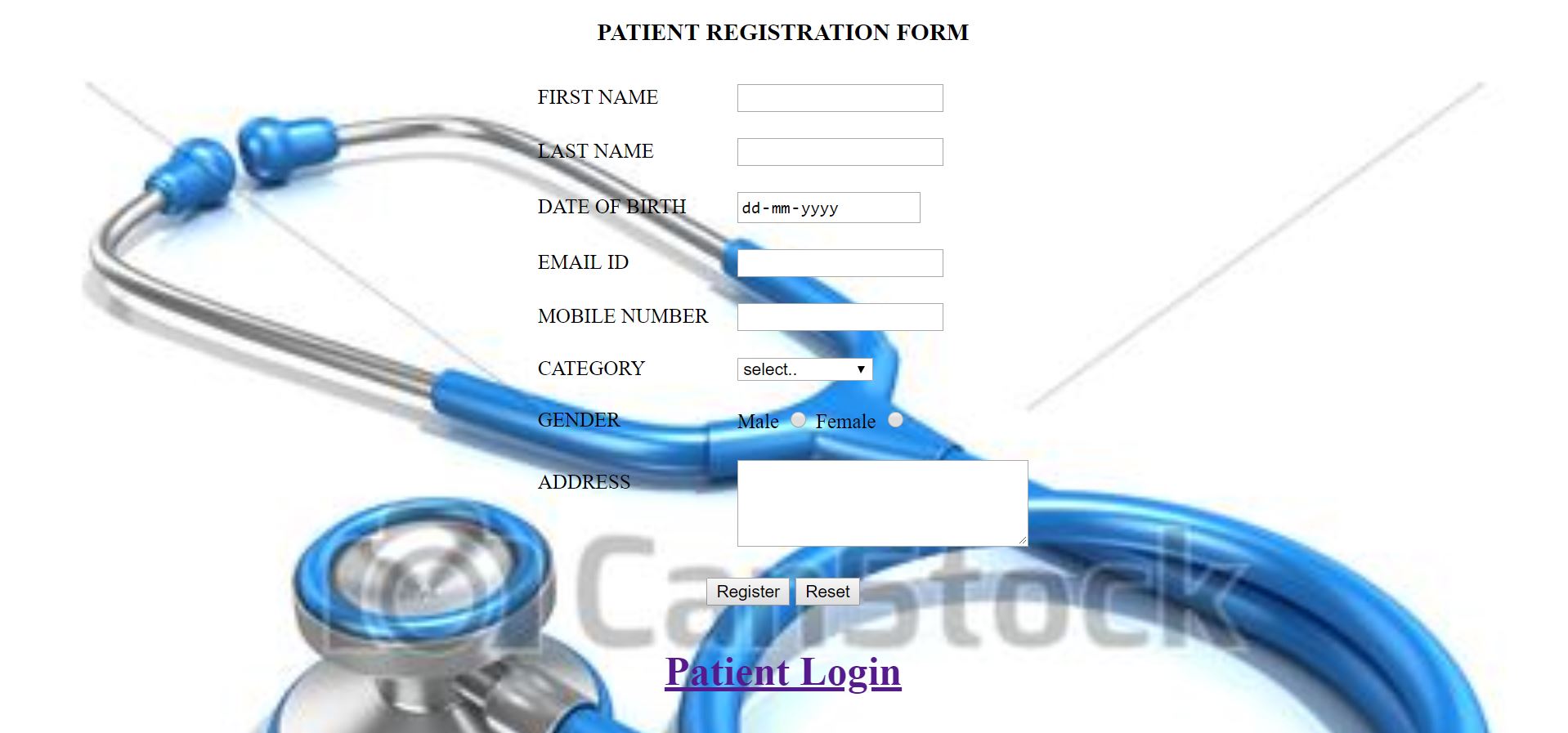
**PATIENT LOGIN:**



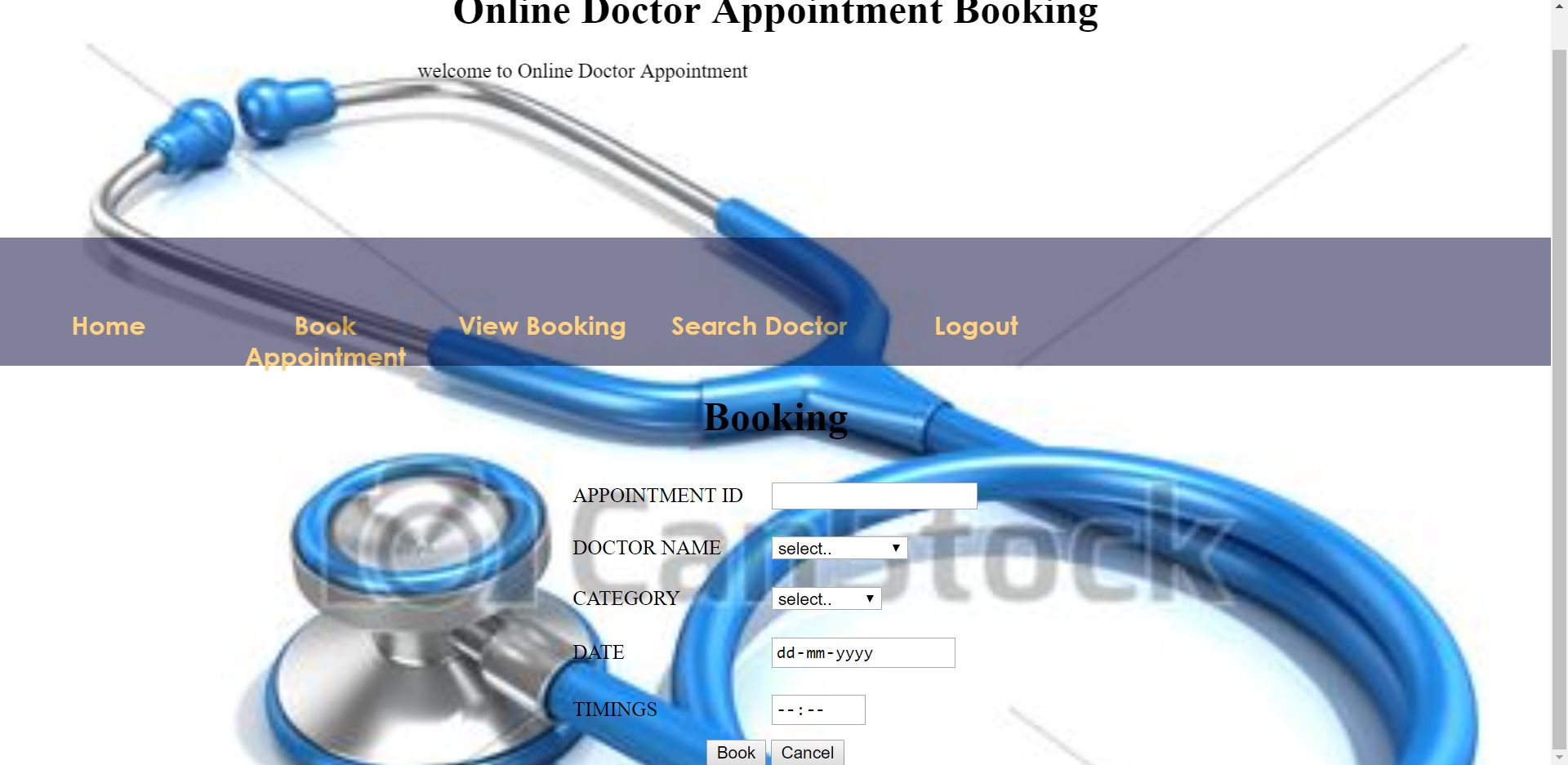
**PATIENT HOME PAGE:**



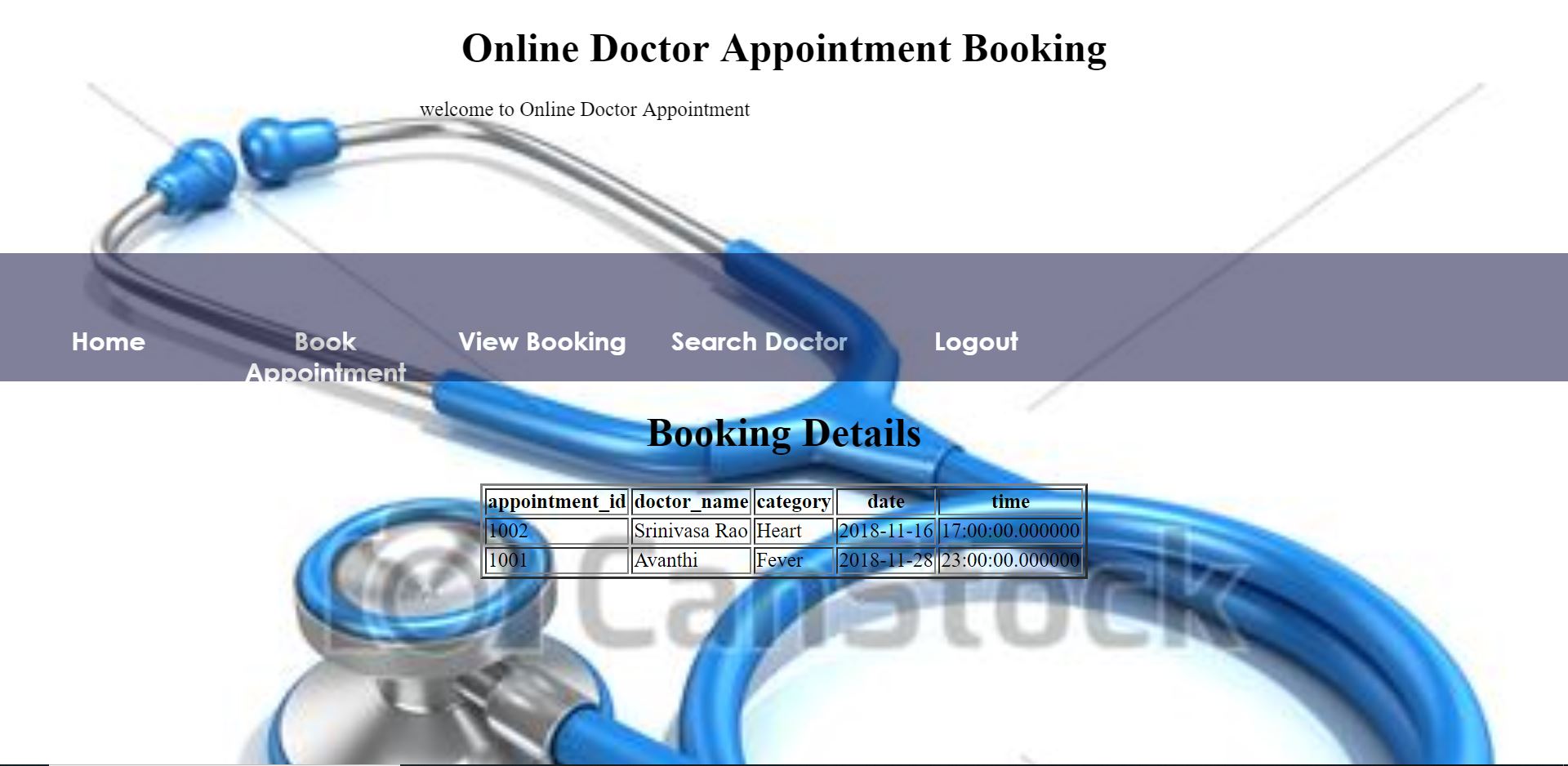
**PATIENT REGISTRATION:**



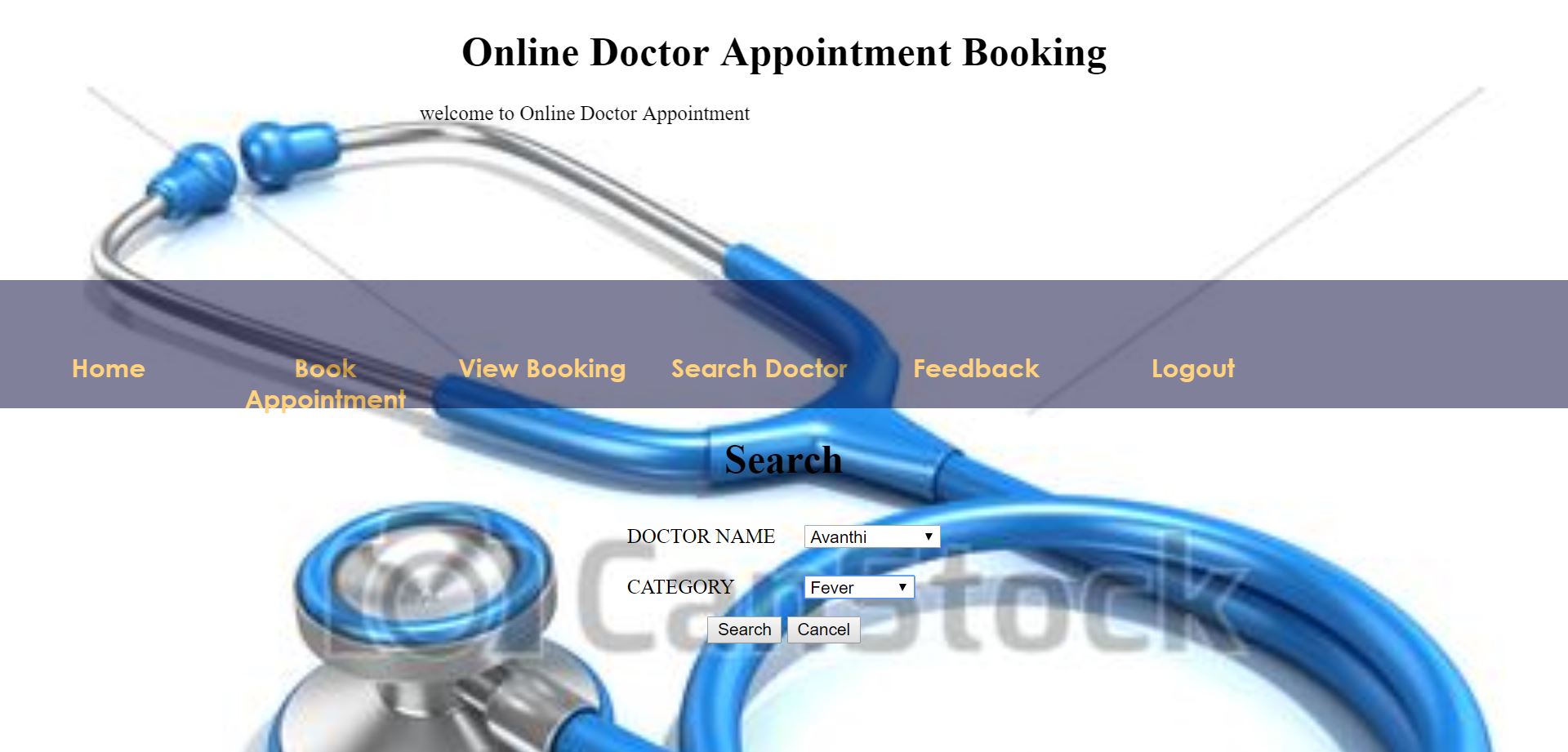
**BOOKING APPOINTMENT:**



**VIEW BOOKING**:



**SEARCH:**



**SEARCH RESULTS:**



**SEARCH RESULTS-ERROR:**



**DOCTOR LOGIN:**



**DOCTOR HOME PAGE:**



**MYAPPOINTMENTS:**



**VIEW PATIENTS:**



**LOG OUT:**



**Chapter 6**

**Conclusion and Future Work**

* This system helps to reduce the waiting time of the patient.
* User can select the appointment time according to his preference.
* Available and booked slots are shown in effective graphical user interface.

**References**

[1] <https://www.tutorialspoint.com>

[2] <https://www.freeprojectz.com>

[3] <https://www.1000projects.com>

[4] <https://www.w3schools.com>