**HOSPITAL MANAGEMENT SYSTEM**

**A project report Submitted**

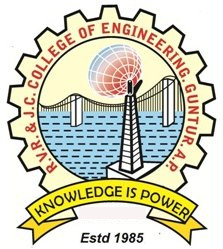
**in partial fulfillment of the Requirements for the Award of**

**MASTER OF COMPUTER APPLICATIONS**

By

**TUNGALA JYOTHI SINDU PRIYA**

**(Y20CA057)**

****

**R.V. R & J.C. COLLEGE OF ENGINEERING**

**(AUTONOMOUS)**

**(Approved by A.I.C.T.E) NAAC A+ Grade**

###### **Chandramoulipuram: Chowdavaram**

##### **GUNTUR – 522 019**

**HOSPITAL MANAGEMENT SYSTEM USING PHP**

**A project report Submitted**

**in partial fulfillment of the Requirements for the Award of**

**MASTER OF COMPUTER APPLICATIONS**

By

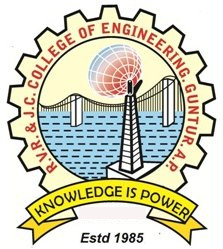
**TUNGALA JYOTHI SINDU PRIYA**

(Y20CA057)

**Under the guidance of**

**Dr.Ch. Suneetha**

(Associate Professor)

****

##### MAY-2022

##### **DEPARTMENT OF COMPUTER APPLICATIONS**

##### **R.V. R & J.C. COLLEGE OF ENGINEERING**

**(AUTONOMOUS)**

**GUNTUR-522 019**

##### **R.V. R & J.C. COLLEGE OF ENGINEERING**

**(AUTONOMOUS)**

**DEPARTMENT OF COMPUTER APPLICATIONS**

****

**BONAFIED CERTIFICATE**

This is to certify that this mini project work titled “**HOSPITAL MANAGEMENT SYSTEM USING PHP”** is done by **TUNGALA JYOTHI SINDU PRIYA** under my guidance in partial fulfillment of the requirements of **CA-253, MINI PROJECT WORK** for the completion of the course **MASTER OF COMPUTER APPLICATIONS,** during the academic year 2022.

(Dr.Ch. Suneetha) (Dr.K.Karteeka Pavan)

Assoc.prof., Prof., & H.O.D

**ABSTRACT**

Hospital is a health care facility that is primarily devoted to the care of all patients. Hospital can be privately operated or publically manage and funded and typically care their primary health care needs of population in local communities, in contracts to larger hospital which gives specialized treatment and admit in patient box our night stays. Some clinic grows to be institution as larger as major hospitals, or become associated with hospitals or medicals school, while retaining the name “clinic”. Health management includes information about admit patients, outdoor patients (OPD) status, outdoor patient’s appointments. Maintaining this information manually is time consuming and not suitable for clinics where number of appointment increases**.**

Problem Statement Since Hospital is associated with the lives of common people and their day-to-day routines so I decided to work on this project. The manual handling of the record is time consuming and highly prone to error. The purpose of this project is to automate or make online, the process of day to-day activities like Room activities; Admission of New Patient, Discharge of Patient, Assign a doctor, and ﬁnally compute the bill etc. I have tried my best to make the complicated process Hospital Management System as simple as possible using Structured Modular technique Menu oriented interface. I have tried to design the software in such a way that user may not have any diﬃculty in using this package further expansion is possible without much eﬀort. Even though I cannot claim that this work to be entirely exhaustive, the main purpose of my exercise is performing each Hospital’s activity in computerized way rather than manually which is time consuming. I am conﬁdent that this software package can be readily used by non-programming personal avoiding human handled chance of error.

**ACKNOWLEDGEMENT**

From the idea to the art, from the conception to the reality, from the emotion to the response, from the desire to the spasm, I led by those, about whom to write all words seem meek.

I express My sincere thanks to **Dr. Ch. Suneetha, Assoc. Prof**, project guide for the project, who has inspired me to select this project, and also for her valuable advice to complete this project.

I am very glad to express my special thanks to **Dr. Ch. Suneetha, Assoc. Prof**, project-in-charge who was inspired me to select this topic and also for her valuable advices in preparing this project.

I express my sincere thanks to **Dr. K. Karteeka Pavan**, Head of the Department of Computer Applications for her encouragement and support to carry out this project successfully.

I am very much thankful to **Dr. K. RAVINDRA**, Principal of **RVR & JC COLLEGE OF ENGINEERING**, Guntur for having allowed to delivering this project.

Finally, I submit my reserves thanks to Lab staff in **Department of Computer Applications** and to all my **friends** for their cooperation during the preparation and developing the project.

**TUNGALA JYOTHI SINDU PRIYA**

**(Y20CA057)**

**CHAPTER 1**

**INTRODUCTION**

As long as each stage implementation needs to be accurate and explicit, the clinic management system provides certain [automation](https://www.medicaldevice-network.com/features/expert-views-benefits-automation-healthcare-industry/) of many vital daily processes. The hospital system software covers the services that unify and simplify the work of healthcare professionals as well as their interactions with patients.

There is always the wide choice of features that can be included in the system. Moreover, the most important thing they are created to streamline various procedures that meet the needs of all the users. The hospital management system feature list is concentrated on providing the smooth experience of patients, staff and hospital authorities. It might seem that their expectations differ, they still are covered by components of the hospital information system. Quality and security still remain the main criteria of the medical industry. It is also known for the constant and rapid changes to improve the efficiency of medical services and satisfaction of the patients.

Hospital management has greatly changed over the last decades. Business expertise, modern technologies, [connected devices](https://existek.com/embedded-software-outsourcing-company-development-services/), mobile apps, and knowledge of healthcare are key elements for the implementation of hospital management system project. The number of healthcare providers has increased and the patients have a wide choice of medical specialists. The interactions between the hospital and the patient can be simplified for the convenience of both sides. Each institution has the opportunity to create the efficient, clear and fast delivering healthcare model.

**1.1 PROBLEM STATEMENT**

Hospital inventory has to deal with various fields and sections along with different modules. So, to make it simple so that user can handle this project in a simple way without moving their cursor from one screen to another screen.

## **Hospital Management System Project Overview**

When the patient will visit on clinic, they will ask to provide their name, address, sex, age and type or problems. As Hospital Management System Project will have built in feature that will assign the doctor by taking their type of problem as their input. After entering these fields, user will provide with a registration slip including their name, address, sex, age, name of doctor and chamber number. Each registration form will have unique registration number.

 The doctor will prescribe medicines using patient registration number. For taking medicine user have to give their registration number to the clinic medical shop, where they will get all the medicines and their bill details. User will also provide with an option for making payments. User can make their payments either online or by cash. The bill will be provided to the user after payment.

 This Hospital Management System Project will also be used for task scheduling process. This Hospital Management System Project will provide entry and exit details for each particular member of particular inventory department. Each and every user which will access this system should have a valid user id and password. This user id and password must be created by admin and saved in database. When the user will enter their details, it will retrieve its id and password and make comparison by the value stored in variable and the values which are stored in the database. If the given value will match then the user can access the system screen for further operations.

## **1.2 EXISTING SYSTEM:**

Hospitals currently use a manual system for the management and maintenance of critical information. The current system requires numerous paper forms, with data stores spread throughout the hospital management infrastructure. Often information (on forms) is incomplete, or does not follow management standards. Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost.

## **DISADVANTAGES OF EXISTING SYSTEM:**

Multiple copies of the same information exist in the hospital and may lead to inconsistencies in data in various data stores.

## **1.3 PROPOSED SYSTEM:**

The Hospital Management System (HMS) is designed for Any Hospital to replace their existing manual, paper-based system. The new system is to control the following information; patient information, room availability, staff and operating room schedules, and patient invoices.

## **ADVANTAGES OF PROPOSED SYSTEM:**

These services are to be provided in an efficient, cost-effective manner, with the goal of reducing the time and resources currently required for such tasks.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 SYSTEM REQUIREMENTS SPECIFICATION**

**2.1.1 HARDWARE REQUIREMENTS:**

* Minimum : Pentium IV Processor with 1.2GHZ
* Hard Disk : 500GB
* Ram : 2GB
* Display : LCD Monitor

**2.1.2 SOFTWARE REQUIREMENTS:**

* Operating system : Any Operating System after WINDOWS 2000
* Front End : PHP
* Database : My SQL Database

**2.2 USE CASE VIEW:**

Use-cases diagrams graphically represent system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider’s (actor’s) perspective. A use-case diagram may contain all or some of the use cases of a system.

**A use-case diagram can contain:**

* Actors (“things” outside the system)
* Use cases (system boundaries identifying what the system should do)
* 1Interactions or relationship between actors and use cases in the system including the associations, dependencies, and generalizations

Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

In its simple form, a use case can be described as a specific way of using the system from a user’s (actor’s) perspective. A more detailed description might characterize a use case as:

* A pattern of behavior the system exhibits
* A sequence of related transactions performed by an actor and the system
* Delivering something of value to the actor

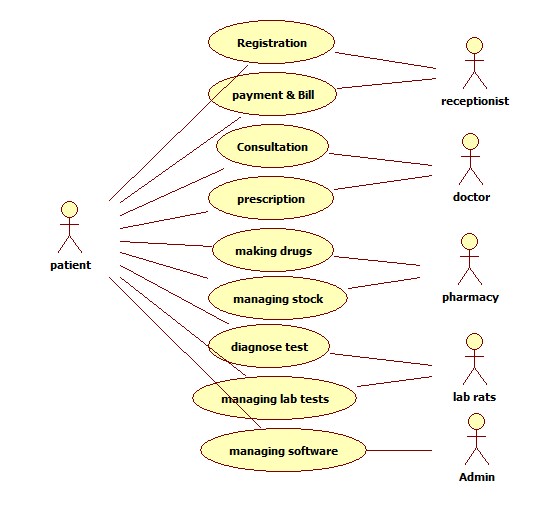


Fig 2.2 Use case diagram

**2.4 SEQUENCE DIAGRAM:**

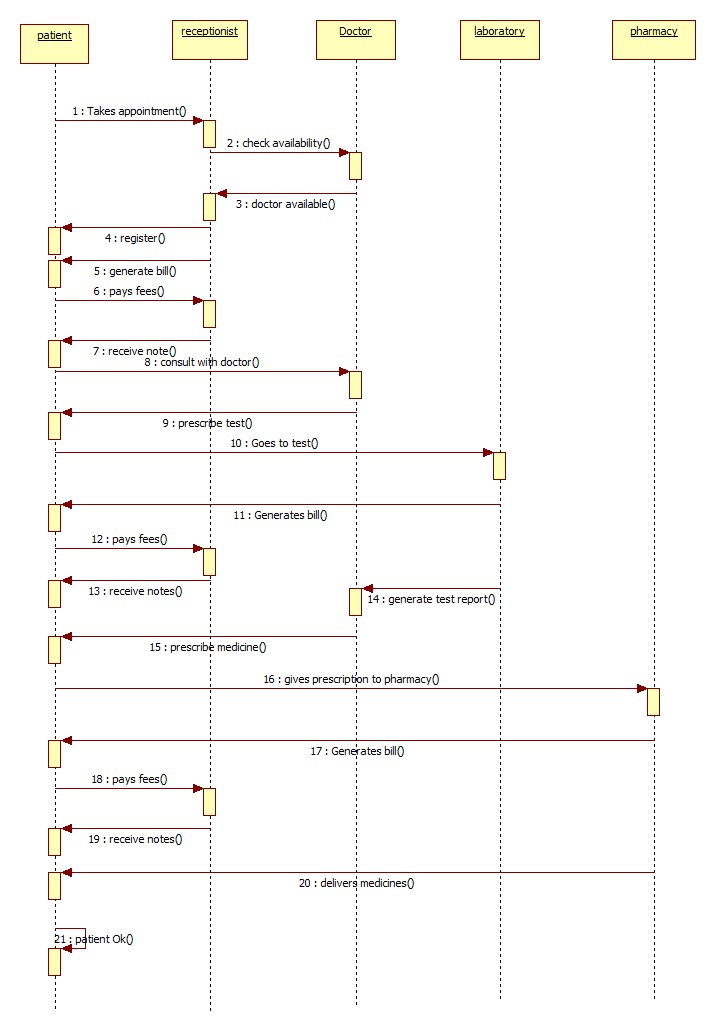
* A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence-what happens first, what happens next….
* Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.
* A sequence diagram has two dimensions: the vertical dimension represents time; the horizontaldimension represents different objects. The vertical line is called the object’s lifeline. The lifeline represents the object’s existence during the interaction.

**Steps:**

1.An object is shown as a box at the top of a dashed vertical line. Object names can be specific (e.g., Algebra 101, Section 1) or they can be general (e.g., a course offering). Often, an anonymous object (class name may be used to represent any object in the class.)

2. Each message is represented by an Arrow between the lifelines of two objects. The order in which these messages occur is shown top to bottom on the page. Each message is labeled with the message name.

3.The sequence diagram is very simple and has immediate visual appeal—this is its great strength. A sequence diagram is an alternative way to understand the overall flow of control of a program. Instead of looking at the code and trying to find out the overall sequence of behavior.

**Sequence diagram**

**2.4 ACTIVITY DIAGRAM:**

An Activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations.

The purpose of Activity diagram is to provide a view of flows and what is going on inside a use case or among several classes.

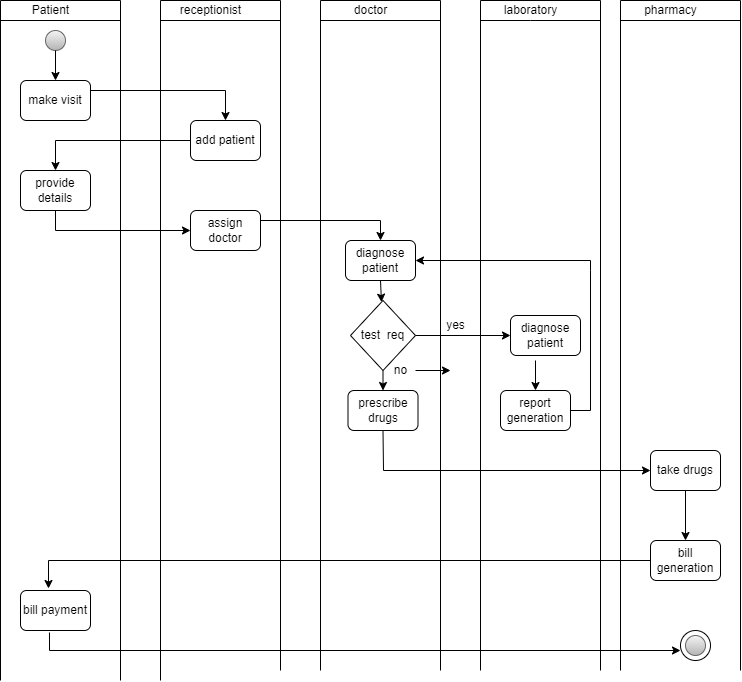
You can also use activity diagrams to model code-specific information such as a class operation.

Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity.

An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities.

* Activity diagrams also may be created at this stage in the life cycle. These diagrams represent the dynamics of the system.
* They are flow charts that are used to show the workflow of a system; that is, they show the flow of control from activity to activity in the system, what activities can be done in parallel, and any alternate paths through the flow.
* At this point in the life cycle, activity diagrams may be created to represent the flow across use cases or they may be created to represent the flow within a particular use case.
* Later in the life cycle, activity diagrams may be created to show the workflow for an operation.

**ACTIVITY DIAGRAM:**



**Fig 2.4 Activity Diagram**

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 Class Diagram**

At the starting, for the identification of classes we need to concentrate completely on uses cases. A further examination of the use cases also helps in identifying operations and the messages that classes need to exchange. However, it is easy to think first in terms of the overall responsibilities of a class rather than its individual operations.

A responsibility is a high-level description of something a class can do. It reflects the knowledge or information that is available to that class, either stored within its own attributes or requested via collaboration with other classes, and also the services that it can offer to other objects. A responsibility may correspond to one or more operations.

It is difficult to determine the appropriate responsibilities for each class as there may be many alternatives that all appear to be equally justified.

**CLASS DIAGRAM**



**Fig 3.1 Class Diagram**

**3.2 REFINING ATTRIBUTES, METHODS AND RELATIONSHIPS**

**Attributes:**

During analysis Stage to consider in detail the data types of the attributes also. Common primitive data types include Boolean (true or false), Character (any alphanumeric or special character), Integer (whole numbers) and Floating-Point (decimal numbers). In most object-oriented languages more complex data types, such as Money, String, Date, or Name can be constructed from the primitive data types or may be available in standard libraries. An attribute's data type is declared in UML using the following syntax:

name ':' type-expression '=' initial-value '{'property-string'}'

The name is the attribute name, the type-expression is its data type, the initial value is the value the attribute is set to when the object is first created and the property-string describes a property of the attribute, such as constant or fixed. The characters in single quotes are literals.

Attribute declarations can also include arrays also. For example, an Employee class might include an attribute to hold a list of qualifications that would be declared using the syntax: Qualification [O ... 10]: String

**Operations:**

Each operation also has to be specified in terms of the parameters that it passes and returns. The syntax used for an operation is:

Operation name' ('parameter-list ') “: “return-type-expression

An operation's *signature* is determined by the operation's name, the number and type of its parameters and the type of the return value if any.

**Object visibility:**

The concept of encapsulation is one of the fundamental principles of object-orientation. During analysis various assumptions have been made regarding the encapsulation boundary for an object and the way that objects interact with each other.

For example, it is assumed that the attributes of an object cannot be accessed directly by other objects but only via 'get' and 'set' operations (primary operations) that are assumed to be available for each attribute. Moving to design involves making decisions regarding which operations (and possibly attributes) are publicly accessible. In other words we must define the encapsulation boundary.

**Interfaces:**

Generally, a class may present more than one external interface to other classes or the same interface may be required from more than one class. An interface in UML is a group of externally visible (i.e., public) operations. The interface contains no internal structure, it has no attributes, no associations and the implementation of the operations is not defined. Formally, an interface is equivalent to an abstract class that has no attributes, no associations and only abstract operations.

The following figure shows two alternative notations for an interface. The simpler of the two UML interface notations is a circle. This is attached by a solid line to the classes that support the interface.

The alternative notation uses a stereotyped class icon. As an interface only specifies the operations and has no internal structure, the attributes compartment is omitted. This notation lists the operations on the diagram. The *realize* relationship, represented by the dashed line with a triangular arrowhead, indicates that the class supports at least the operations listed in the interface.

**Refining Attributes:**

In the analysis phase, the name of the attribute was sufficient. However, in the design phase, detailed information must be added to the model. There are three basic types of attributes. They are:

1) Single-value attributes.

2) Multiplicity or multi-value attributes.

3) Reference to another object, or instance connection.

Attributes represent the state of an object. The following is the attribute presentation:

**Visibility name: type-expression=initial-value**

Where visibility is one of the following:

+ public visibility

# Protected visibility

- private visibility

During analysis, we identified the following attributes for classes:

Refining attributes for Librarian class:

* name
* number

At this stage we need to add more information to these attributes, such as

visibility and implementation type.

+ name: String

+ number: String

Refining attributes for member class:

name

id

After refining

+ id: String

+ name: String

**Refining methods:**

A class can provide several types of methods.

• Constructor: Method that creates instances of the class.

• Destructor: The method that destroys instances.

• Conversion method: The method that converts a value from one unit of measure

to another.

• Copy method: The method that copies the contents of one instance to another

instance.

• Attribute set: The method that sets the values of one or more attributes.

• I/O methods: The methods that provide or receive data to or from a device.

• Domain specific: The method specific to the application.

The operation syntax is:

**Visibility name: (parameter-list): return-type-expression**

**3.3 IMPLEMENTATION DIAGRAMS**

**3.3.a. COMPONENT DIAGRAMS:**

* Component Diagrams show the dependencies between software components in the system. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time or at runtime.
* In a large project there will be many files that make up the system. These files will have dependencies on one another. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time, at link-time or at run-time. There are also dependencies between source code files and the executable files or byte code files that are derived from them by compilation. Component diagrams are one of the two types of implementation diagram in UML. Component diagrams show these dependencies between software components in the system. Stereotypes can be used to show dependencies that are specific to particular languages also

.

* A component diagram shows the allocation of classes and objects to components in the physical design of a system. A component diagram may represent all or part of the component architecture of a system along with dependency relationships.
* The dependency relationship indicates that one entity in a component diagram uses the services or facilities of another.
* Dependencies in the component diagram represent compilation dependencies.
* The dependency relationship may also be used to show calling dependencies among components, using dependency arrows from components to interfaces on other components.

**Types of components:**

* Deployment component
* Work product component
* Executable component

**Uses:**

* To model source code.
* To model executable releases.
* To model physical databases.
* To model adaptable systems.

**3.3.b. DEPLOYMENT DIAGRAMS:**

* The second type of implementation diagram provided by UML is the deployment diagram. Deployment diagrams are used to show the configuration of run-time processing elements and the software components and processes that are located on them.
* Deployment diagrams are made up of nodes and communication associations. Nodes are typically used to show computers and the communication associations show the network and protocols that are used to communicate between nodes. Nodes can be used to show other processing resources such as people or mechanical resources.
* Nodes are drawn as 3D views of cubes or rectangular prisms, and the following figure shows a simplest deployment diagram where the nodes connected by communication associations.

**Uses:**

* + To model embedded system.
  + To model client-server system.
  + To model distributed systems.

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

**4.1 CODE:**

**Code for login:**

**<!DOCTYPE HTML>**

**<html>**

**<head>**

**<title>Hospital Management System</title>**

**<link href="css/style.css" rel="stylesheet" type="text/css" media="all" />**

**<link href='http://fonts.googleapis.com/css?family=Ropa+Sans' rel='stylesheet' type='text/css'>**

**<link rel="stylesheet" href="css/responsiveslides.css">**

**<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>**

**<script src="js/responsiveslides.min.js"></script>**

**<script>**

**// You can also use "$(window).load(function() {"**

**$(function () {**

**// Slideshow 1**

**$("#slider1").responsiveSlides({**

**maxwidth: 1600,**

**speed: 600**

**});**

**});**

**</script>**

**</head>**

**<body>**

**<!--start-wrap-->**

**<!--start-header-->**

**<div class="header">**

**<div class="wrap">**

**<!--start-logo-->**

**<div class="logo">**

**<a href="index.html" style="font-size: 30px;"> HBR | Hospital Management System</a>**

**</div>**

**<!--end-logo-->**

**<start-top-nav-->**

**<div class="top-nav">**

**<ul>**

**<li class="active"><a href="index.html">Home</a></li>**

**<li><a href="contact.php">contact</a></li>**

**</ul>**

**<div>**

**<div class="clear"> </div>**

**<!--end-top-nav-->**

**</div>**

**<!--end-header-->**

**</div>**

**<div class="clear"> </div>**

**<!--start-image-slider---->**

**<div class="image-slider">**

**<!-- Slideshow 1 -->**

**<ul class="rslides" id="slider1">**

**<li><img src="images/slider-image2.jpg" alt=""></li>**

**</ul>**

**<!-- Slideshow 2 -->**

**</div>**

**<!--End-image-slider---->**

**<div class="clear"> </div>**

**<div class="content-grids">**

**<div class="wrap">**

**<div class="section group">**

**<div class="listview\_1\_of\_3 images\_1\_of\_3">**

**<div class="listimg listimg\_1\_of\_2">**

**<img src="images/grid-img3.png">**

**</div>**

**<div class="text list\_1\_of\_2">**

**<h3>Patients</h3>**

**<p>Register & Book Appointment</p>**

**<div class="button"><span><a href="hms/user-login.php">Click Here</a></span></div>**

**</div>**

**</div>**

**<div class="listview\_1\_of\_3 images\_1\_of\_3">**

**<div class="listimg listimg\_1\_of\_2">**

**<img src="images/grid-img1.png">**

**</div>**

**<div class="text list\_1\_of\_2">**

**<h3>Doctors Login</h3>**

**<div class="button"><span><a href="hms/doctor/">Click Here</a></span></div>**

**</div>**

**</div>**

**<div class="listview\_1\_of\_3 images\_1\_of\_3">**

**<div class="listimg listimg\_1\_of\_2">**

**<img src="images/grid-img2.png">**

**</div>**

**<div class="text list\_1\_of\_2">**

**<h3>Admin Login</h3>**

**<div class="button"><span><a href="hms/admin">Click Here</a></span></div> </div>**

**</div>**

**</div>**

**</div>**

**</div>**

**<div class="wrap">**

**<div class="content-box">**

**<div class="section group">**

**<div class="col\_1\_of\_3 span\_1\_of\_3 frist">**

**</div>**

**<div class="col\_1\_of\_3 span\_1\_of\_3 second">**

**</div>**

**<div class="col\_1\_of\_3 span\_1\_of\_3 frist">**

**</div>**

**</div>**

**</div>**

**</div>**

**<div class="clear"> </div>**

**<div class="footer">**

**<div class="wrap">**

**<div class="footer-left">**

**<ul>**

**<li><a href="index.html">Home</a></li>**

**<li><a href="contact.php">contact</a></li>**

**</ul>**

**</div>**

**<div class="clear"> </div>**

**</div>**

**</div>**

**<!--end-wrap-->**

**</body>**

**</html>**

**Check\_availability.php**

<?php

require\_once("include/config.php");

if(!empty($\_POST["email"])) {

$email= $\_POST["email"];

$result =mysqli\_query($con,"SELECT email FROM users WHERE email='$email'");

$count=mysqli\_num\_rows($result);

if($count>0)

{

echo "<span style='color:red'> Email already exists .</span>";

echo "<script>$('#submit').prop('disabled',true);</script>";

} else{

echo "<span style='color:green'> Email available for Registration .</span>";

echo "<script>$('#submit').prop('disabled',false);</script>";

}

}

?>

getdoctor.php

<?php

include('include/config.php');

if (!empty($\_POST["specilizationid"])) {

$sql = mysqli\_query($con, "select doctorName,id from doctors where specilization='" . $\_POST['specilizationid'] . "'"); ?>

<option selected="selected">Select Doctor </option>

<?php

while ($row = mysqli\_fetch\_array($sql)) { ?>

<option value="<?php echo htmlentities($row['id']); ?>">

<?php echo htmlentities($row['doctorName']); ?>

</option>

<?php

}

}

if (!empty($\_POST["doctor"])) {

$sql = mysqli\_query($con, "select docFees from doctors where id='" . $\_POST['doctor'] . "'");

while ($row = mysqli\_fetch\_array($sql)) { ?>

<option value="<?php echo htmlentities($row['docFees']); ?>">

<?php echo htmlentities($row['docFees']); ?>

</option>

<?php

}

}

?>

**logout.php**

<?php

include('include/config.php');

if (!empty($\_POST["specilizationid"])) {

$sql = mysqli\_query ($con, "select doctorName,id from doctors where specilization='" . $\_POST['specilizationid'] . "'"); ?>

<option selected="selected">Select Doctor </option>

<?php

while ($row = mysqli\_fetch\_array($sql)) { ?>

<option value="<?php echo htmlentities($row['id']); ?>">

<?php echo htmlentities($row['doctorName']); ?>

</option>

<?php

}

}

if (!empty($\_POST["doctor"])) {

$sql = mysqli\_query($con, "select docFees from doctors where id='" . $\_POST['doctor'] . "'");

while ($row = mysqli\_fetch\_array($sql)) { ?>

<option value="<?php echo htmlentities($row['docFees']); ?>">

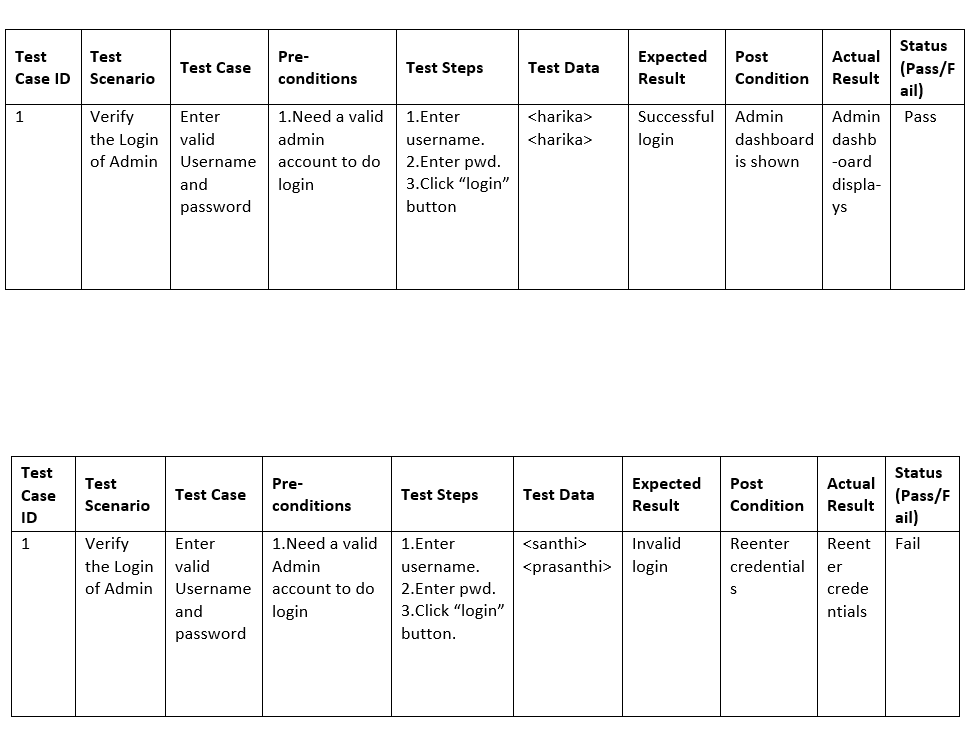
<?php echo htmlentities($row['docFees']); ?>

</option>

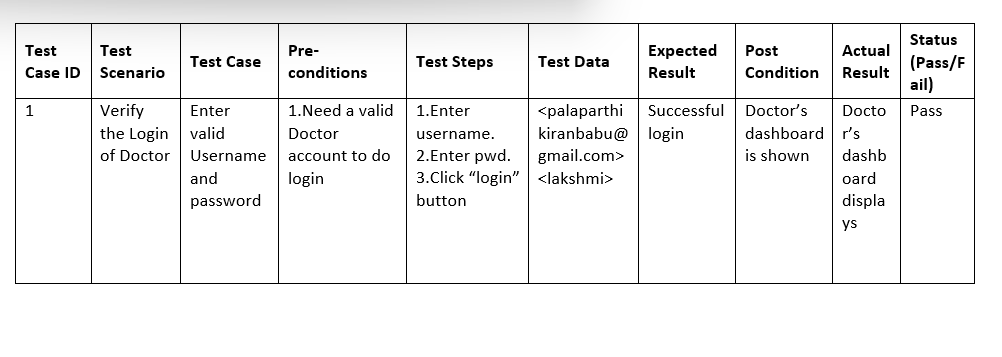
<?php>

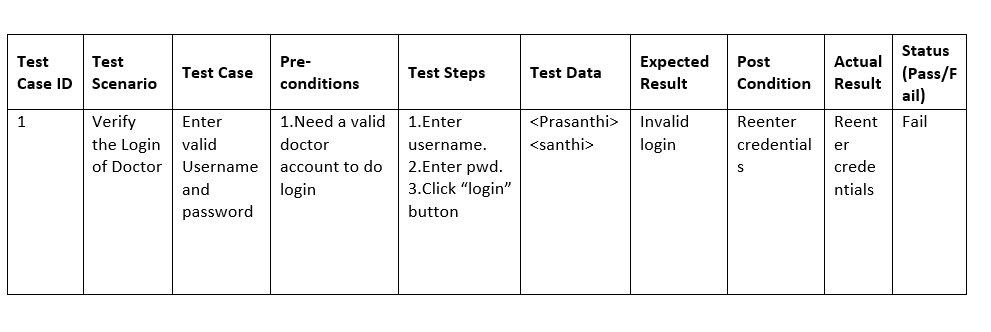
**4.2 TEST CASES:**

**Test case for Admin login:**

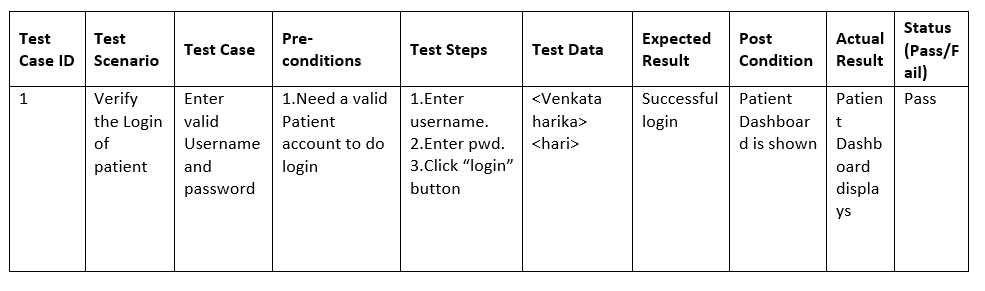
****

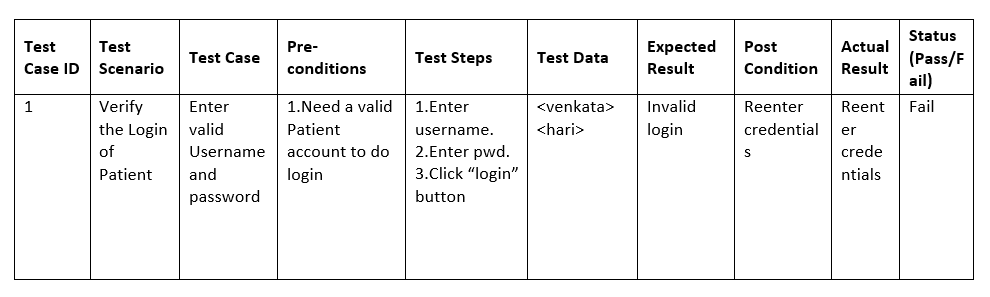
**Test cases for doctor login:**

****

****

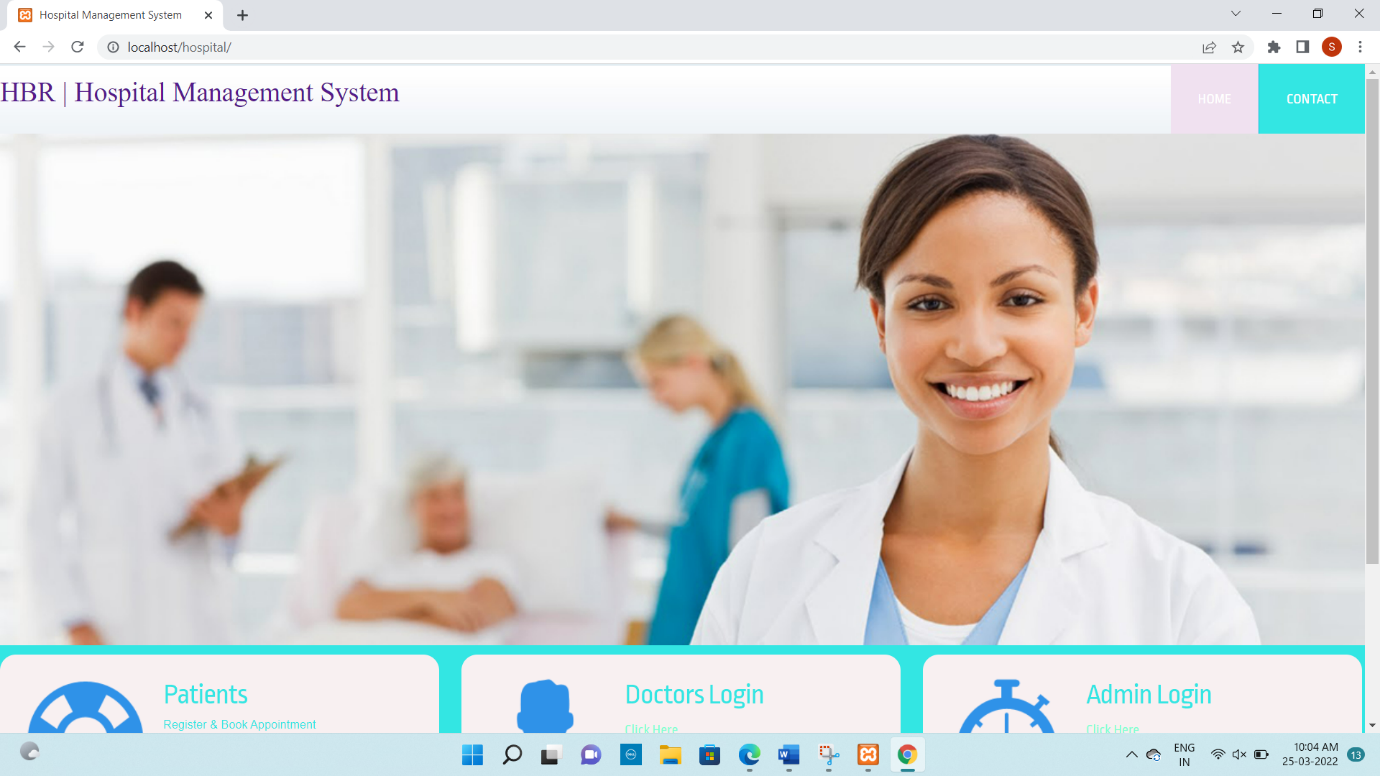
**Testcase for Patient login:**

****

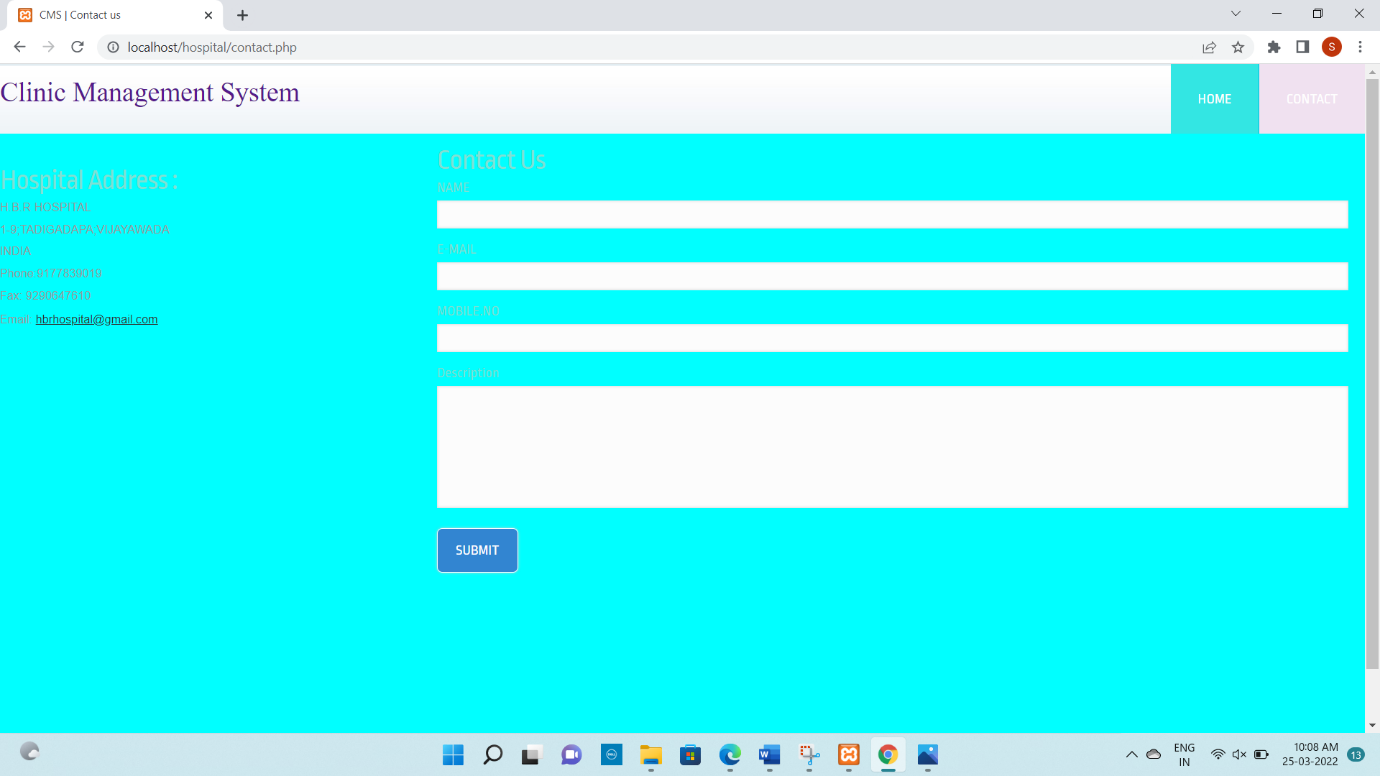
****

**4.3 OUTPUT SCREEN:**

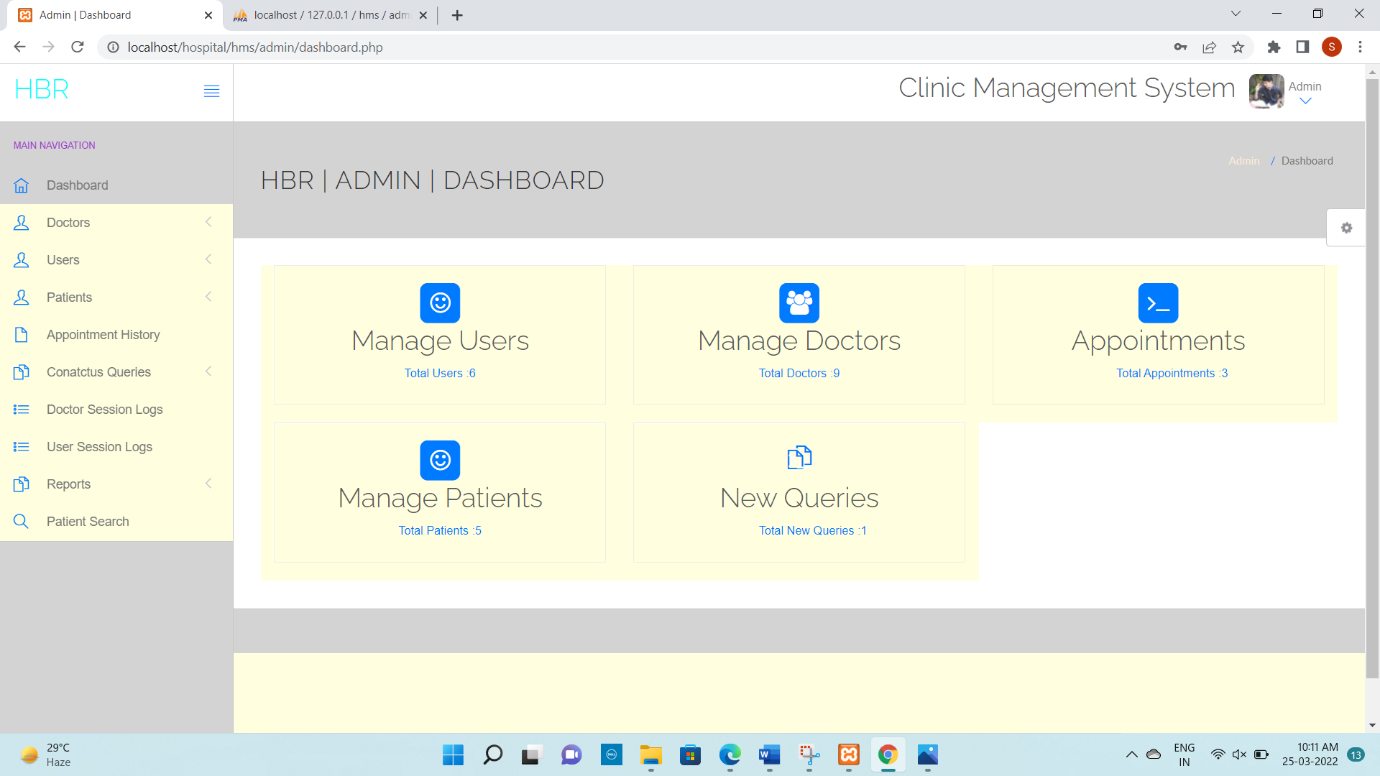
**HOME SCREEN**



CONTACT:



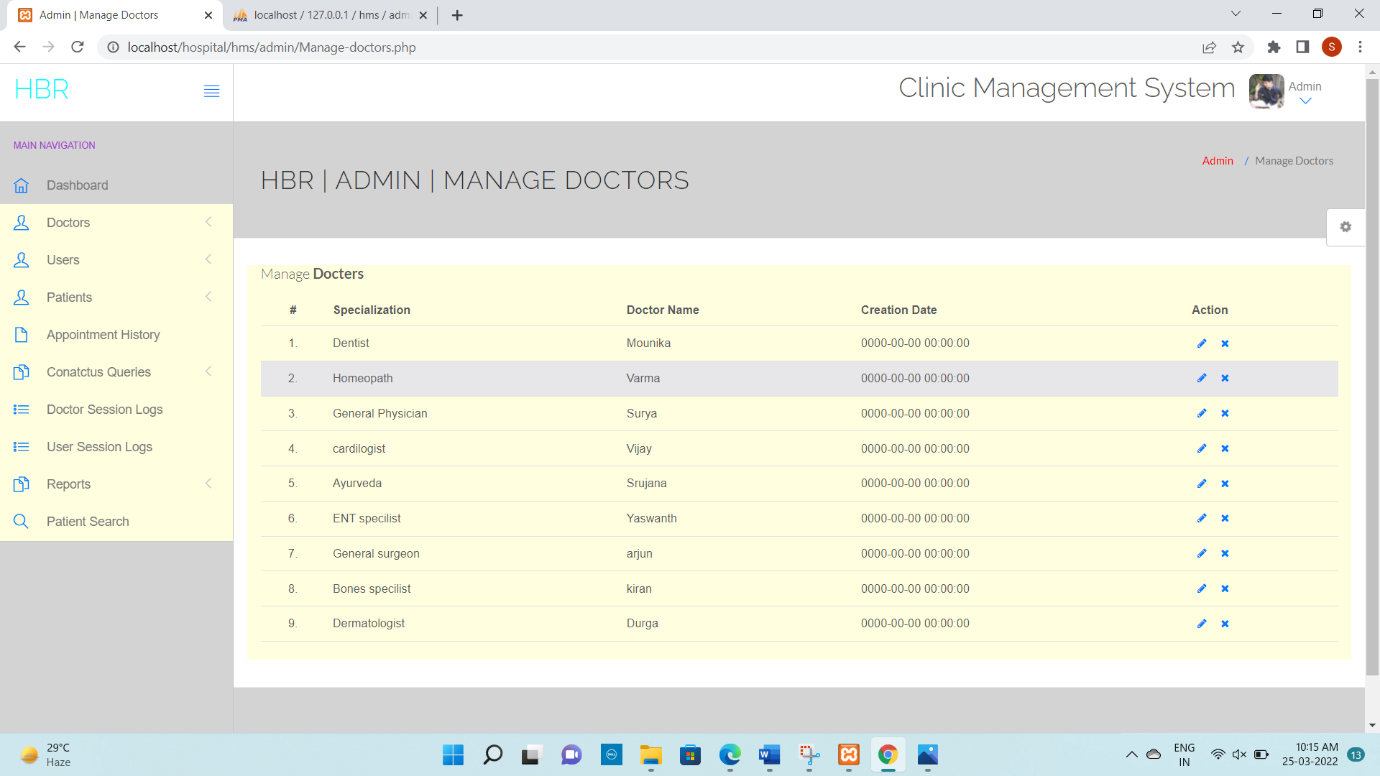
**ADMIN:**



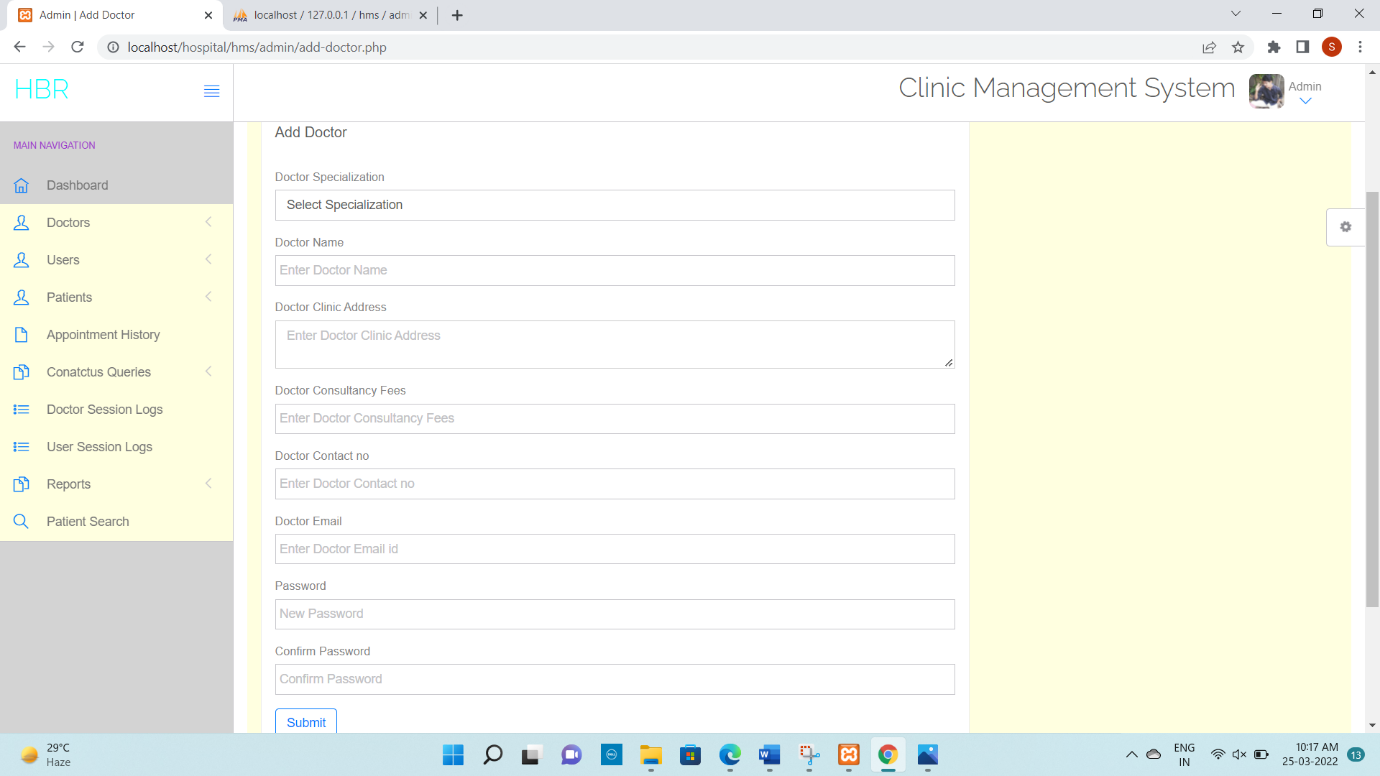
**Manage Patients (Admin):**



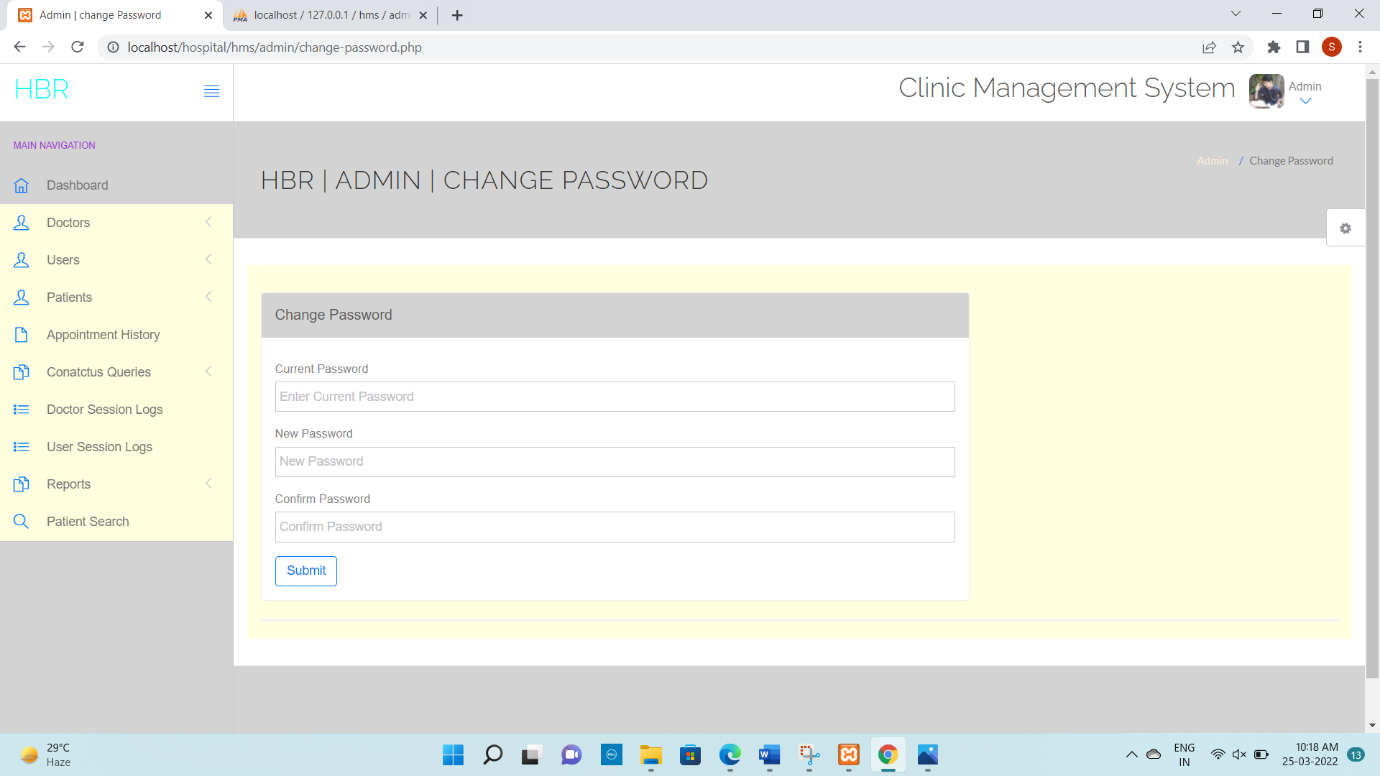
**Manage Doctors (Admin):**



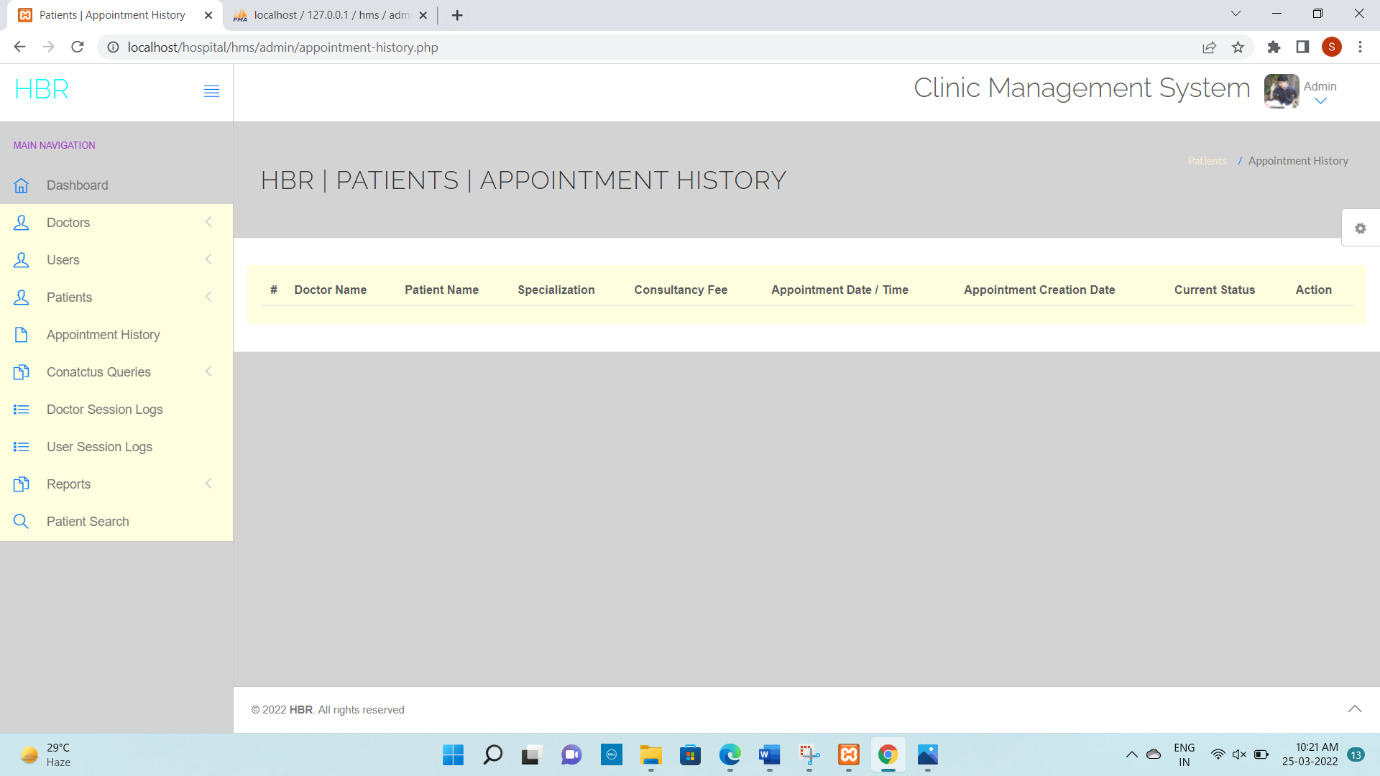
Adding Doctor:



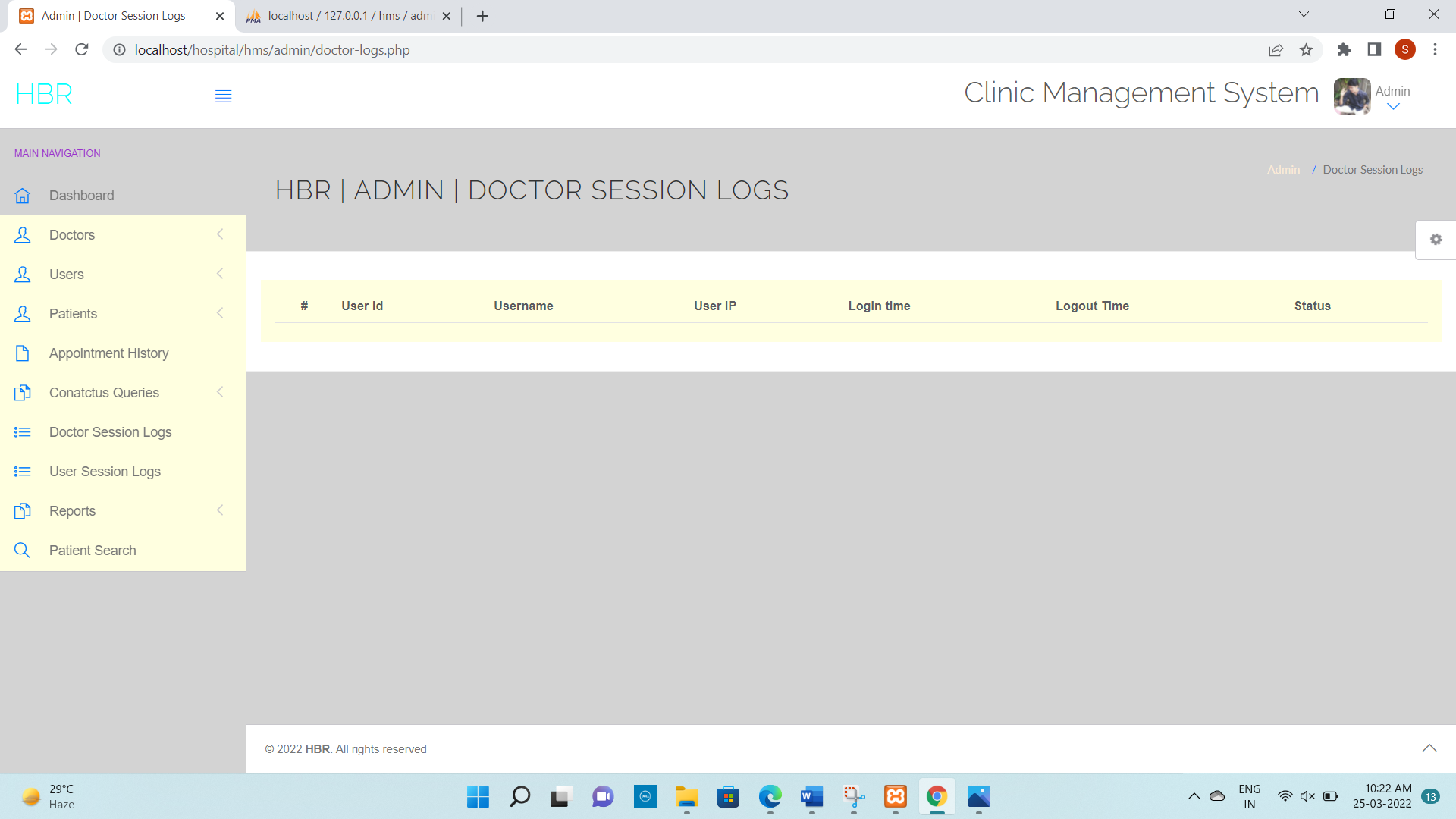
**Edit Admin Details:**



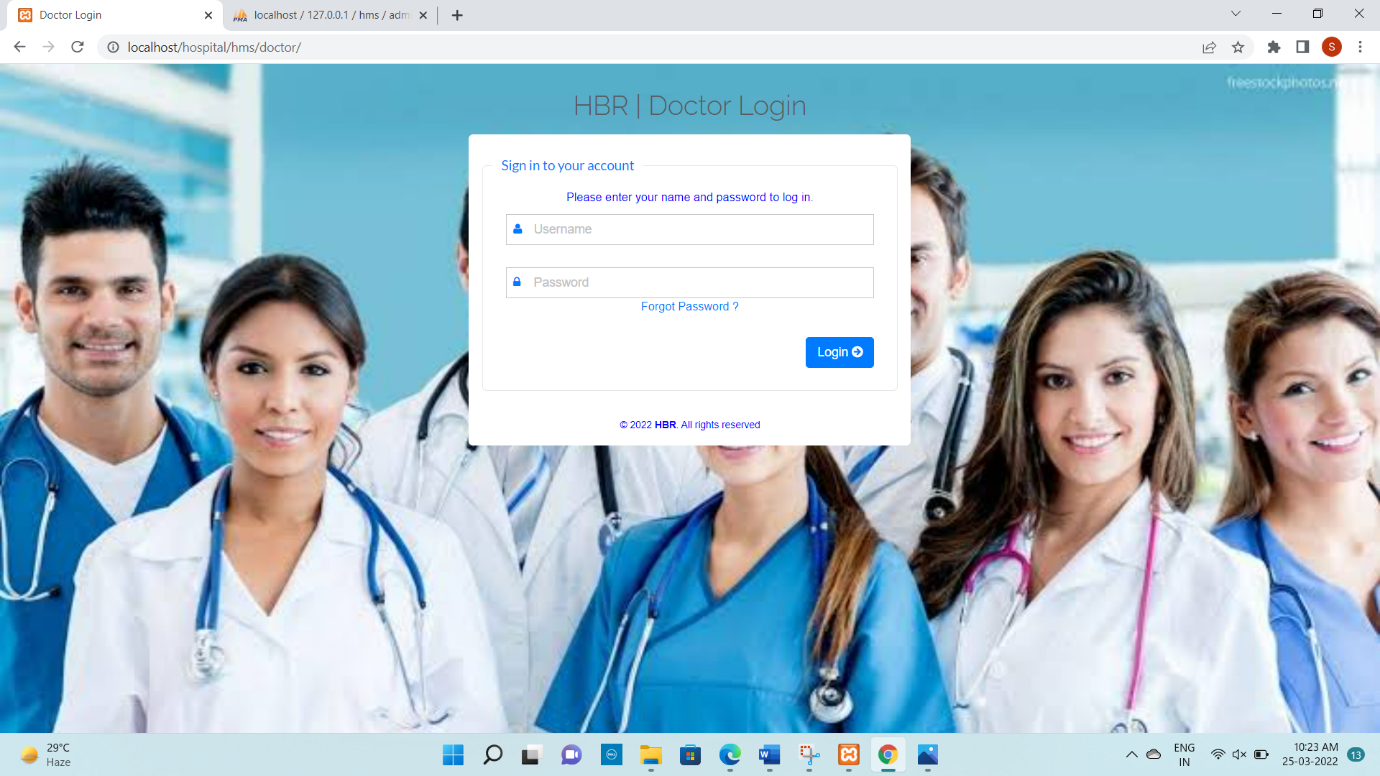
**Appointment History:**



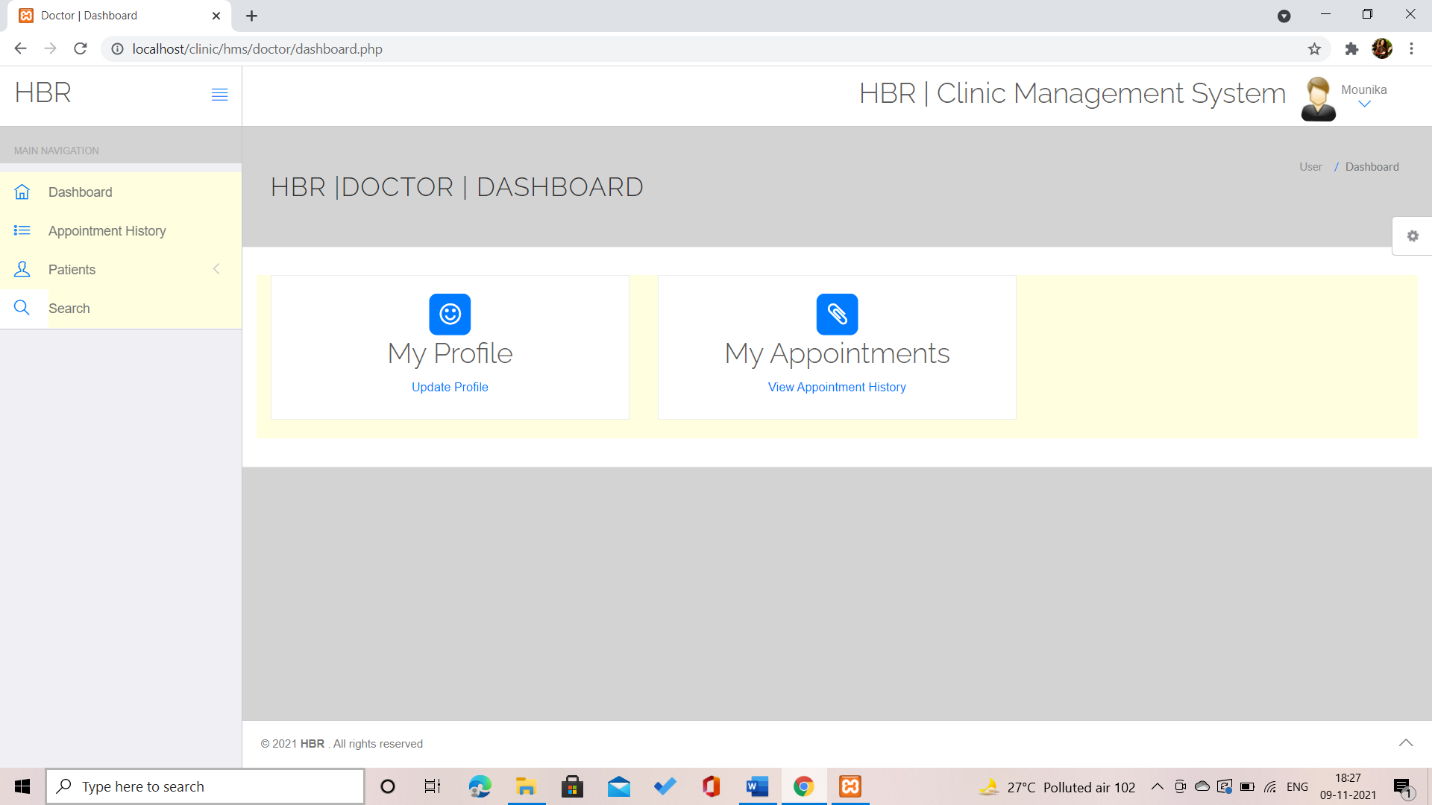
**Doctor Session Login:**



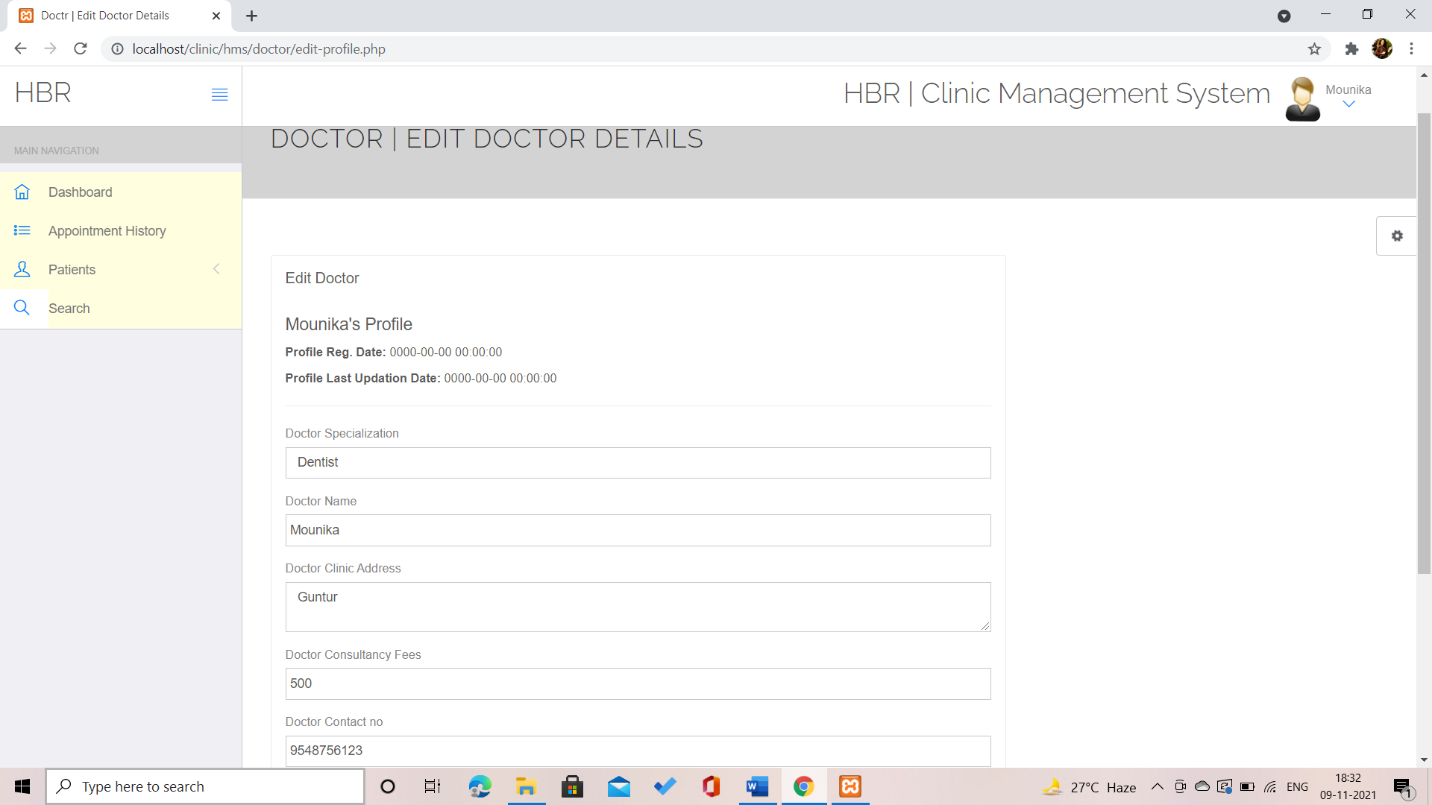
**Doctor login:**



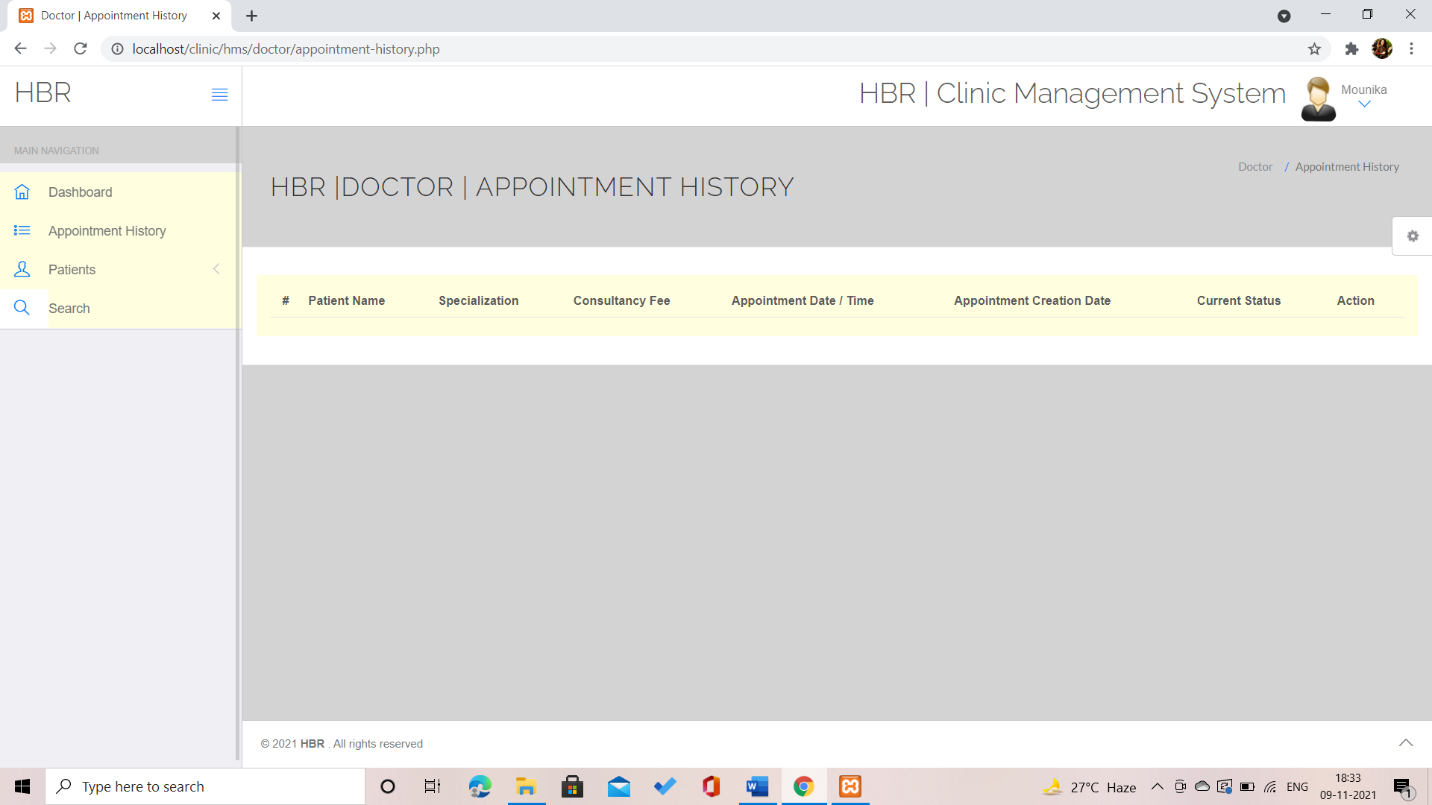
Doctor dashboard:



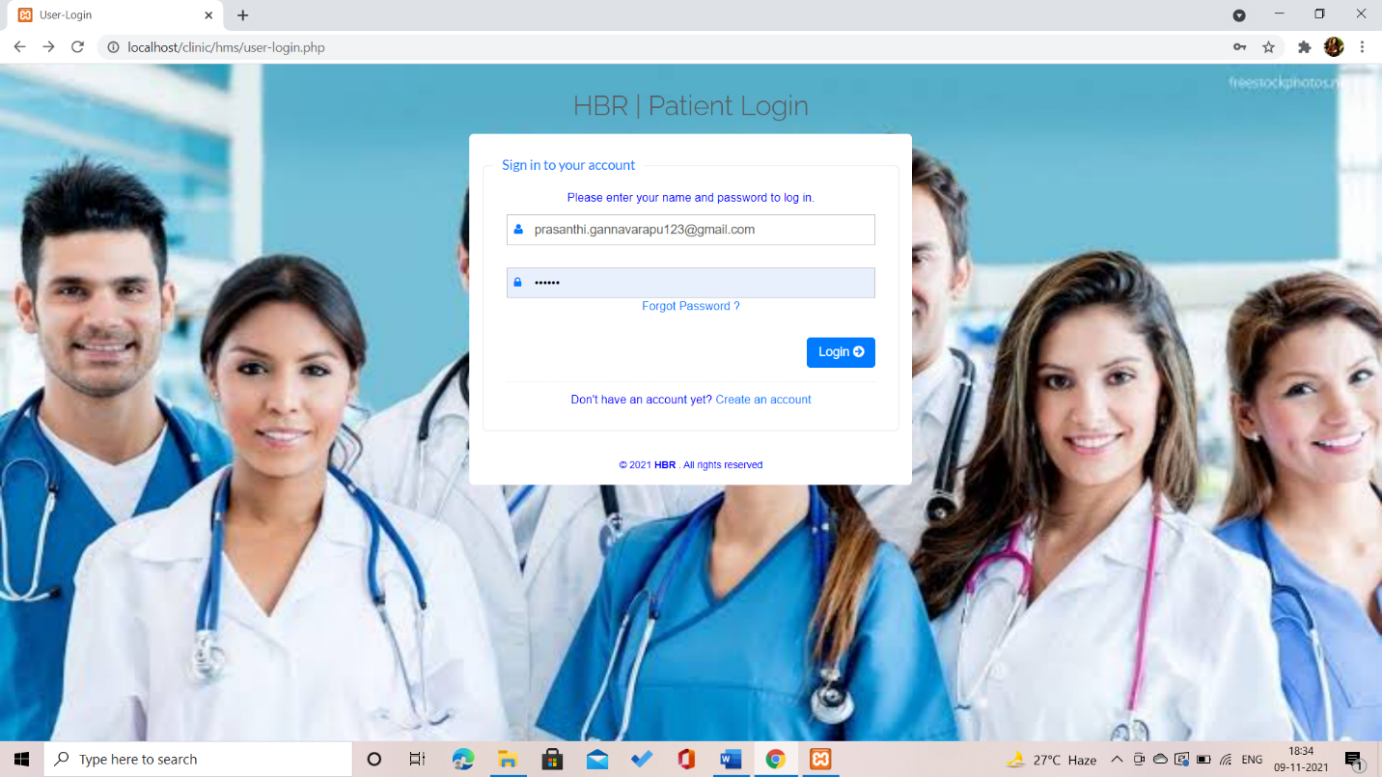
Manage doctor details:



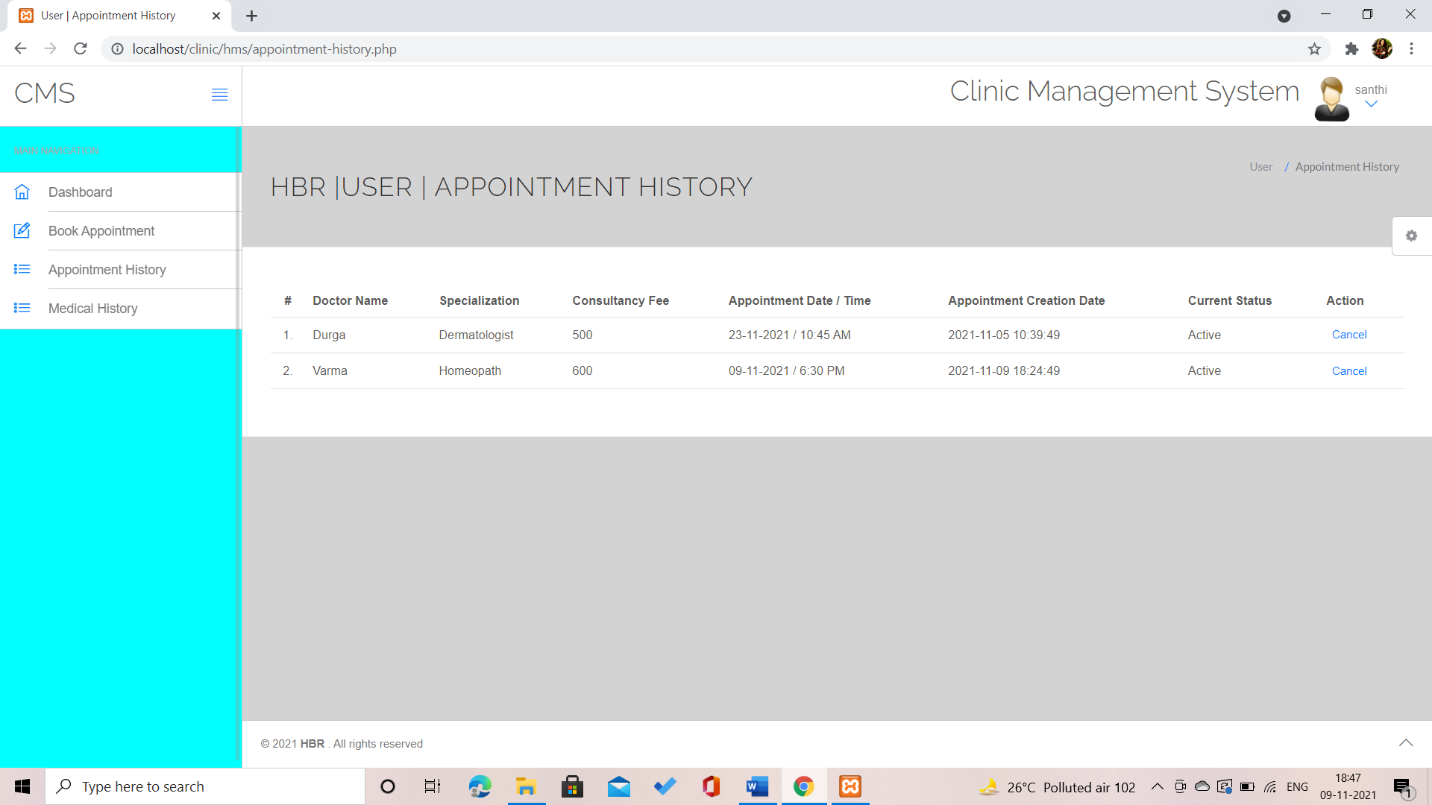
View appointment history:



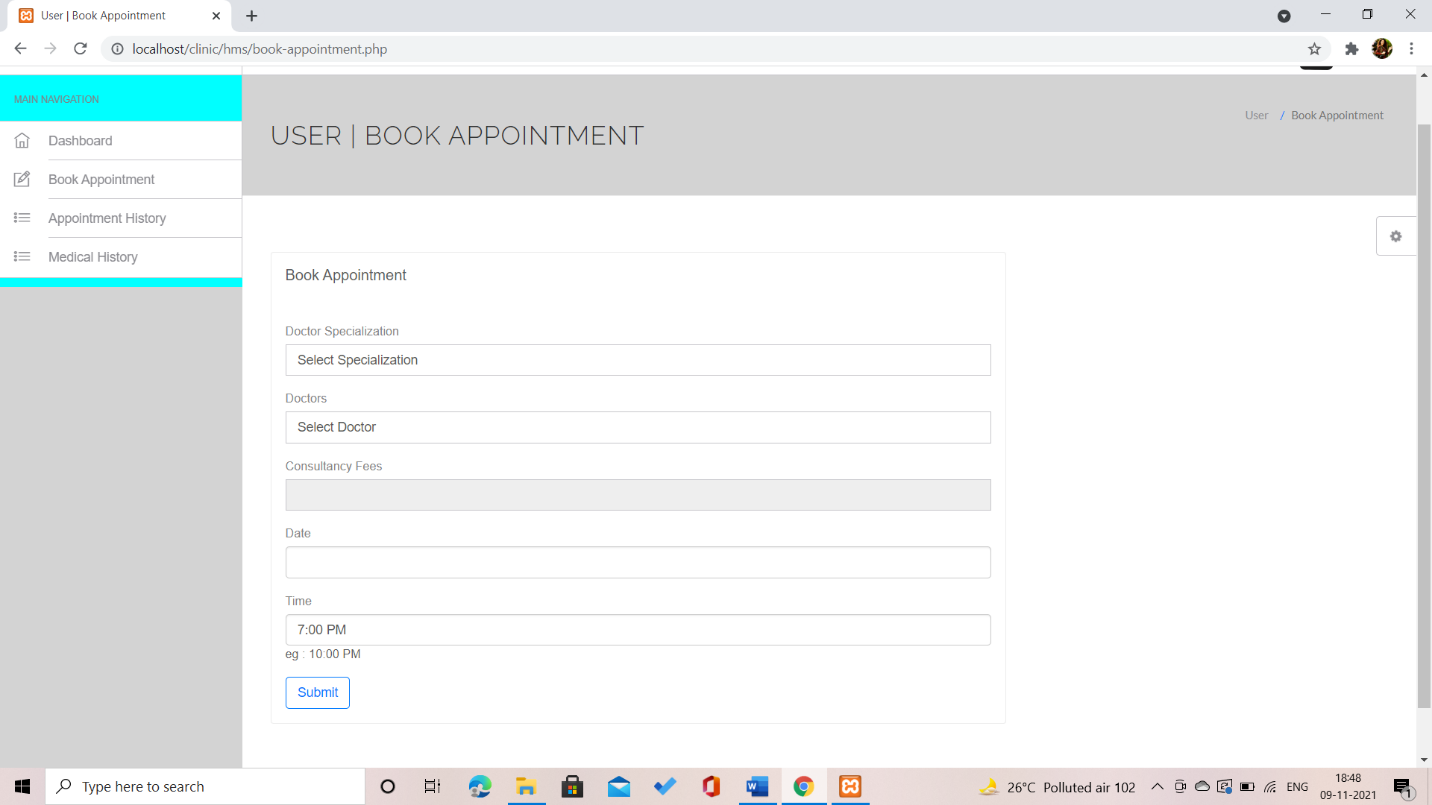
**Patient login:**



Medical History :



**Book Appointment:**



**CHAPTER 5**

**CONCLUSION**

The Hospital management System allows the patients to schedule appointment through online and report their problem. The system has to maintain the details of patients. Patients have to login through valid username and password and book appointment .

The Administrator has full authority over user accounts. He maintains the patients and doctors information. He maintains the list of all patients and their status information. He can change the status of a patient’s appointment if it is resolved. After checking all the information, he logs out from his homepage.

The Hospital Management System has many advantages in following ways:

* Better monitoring of Patients Appointment.
* Better understanding of nature of problems.
* Resolves issues in a timely and cost effective way.
* Helps in improving the services offered to the Patients.

The new system can be improved by adding few more features like:

* Allowing Patient to cancel the Approved Appointment.
* Maintaining the Medicine stock.
* Maintaining patient’s medical records.

**CHAPTER 6**

**REFERENCES**

1. Object Oriented Analysis and Design 2005 by Mike O’docherty.
2. Visual Modeling with Rational Rose 2002 by Teny Quatran.
3. Applying UML and patterns: An Introduction to Object Oriented Analysis and Design, Third Edition 2005 by Craig Larman.
4. Using UML: Software Engineering with objects and components, Second Edition 2006 by Perdita Stevens, R. J. Pooley.
5. The elements of UML style by Scott W. Ambler.
6. The Unified Modelling Language User Guide by Grady Booch, James Rumbaugh, Ivar Jacobson.

**Web Links:**

1. Hospital Management System – [<https://www.wix.com/>]
2. [www.w3schools.com/php](http://www.w3schools.com/php)
3. <https://stackoverflow.com>