

# **CHAPTER-1**

## **INTRODUCTION**

### **1.1 MOTIVATION**

Hospitals are the essential part of our lives, providing best medical facilities to people suffering from various ailments. Which may be due to change in climatic conditions, increased work-load emotional stress etc. Provide day-to-day updates of the hospital through web application. It helps user to find the best hospital and specialist doctors in different places for medical emergencies.

### **1.2 PROBLEM DEFINITION**

Common People are troubling to choose a hospital for required treatment during medical emergencies. This is because of lack of transparency of facilities in the hospitals. Especially in small towns and rural areas people does not know whether the doctor available in the hospital or not. There is no direct communication between hospital and patient. There is no online appointment booking process especially in rural areas.

### **1.3 OBJECTIVE OF THE PROJECT**

We are directly collecting the information from different hospitals in that locality and broadcast it to all the people through a web application. Scheduling the appointment of patient with doctors to make it convenient for both. A login will be given to all hospitals and data will be updated by hospital staff every day. Scheduling the services of specialized doctors and emergency properly so that facilities provided by hospital are fully utilized in effective and efficient manner. The information of the patient should be kept up to date and there record should be kept in the system for historical purposes.

### **1.4 LIMITATIONS OF PROJECT**

Some of the limitations are

- The system is does not provide online payment for appointment booking.
- There is no centralized database maintenance.
- Appointment booking is limited to some hospitals.

## **1.5 ORGANISATION OF DOCUMENTATION**

### **1.5.1 Feasibility Study**

After doing the project Hospital Finder, study and analyzing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project.

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

- Technical Feasibility
- Operation Feasibility
- Economic Feasibility

#### **Technical Feasibility**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment's have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

#### **Operation Feasibility**

The operational feasibility includes User friendly, reliability, security, portability, availability and maintainability of the software used in the project.

#### **Economic Feasibility**

Analysis of a project costs and revenue in an effort to determine whether or not it is logical and possible to complete.

## **CHAPTER-2**

### **LITERATURE SURVEY**

#### **2.1 EXISTING SYSTEM**

Existing system refers to the system that is being followed till now. Presently all hospital functionalities are done manually. That is if a patient wants to consult a doctor, he can visit there till his chance called. This is very difficult for patients. It is time consuming process. And we don't have proper updates of doctor availability. Especially in rural areas there is no online appointment booking process.

#### **2.2 DISADVANTAGES OF EXISTING SYSTEM**

- Applications available today do not provide status of doctor's availability in the hospital.
- Limited to major cities and towns.
- No day-to-day updates
- Limited to branded and multispecialty hospitals.

#### **2.3 PROPOSED SYSTEM**

In the proposed system, we can check the availability of the doctor and also displays number of appointments booked on that particular day. After booking the appointment user will get the message as your appointment is booked.

#### **2.4 ADVANTAGES OF EXISTING SYSTEM**

- Provides day-to-day updates.
- User friendly.
- Time saving
- Provides hospital details like location, number of beds available and etc.
- Data will be directly updated by the hospital staff.

## **CHAPTER-3**

### **ANALYSIS**

#### **3.1 SOFTWARE REQUIREMENT SPECIFICATION**

- **FRONT END:** HTML, CSS
- **BACK END:** PHP, JAVA SCRIPT

#### **3.2 HARDWARE REQUIREMENTS**

- Processor
- Keyboard
- Desktop
- Mouse

#### **3.3 REQUIRED THINGS AND RELATIONSHIPS**

- **Structural things:** Class
- **Grouping things:** Package
- **Annotation things:** Note

**Relationships:** Association, Generalization, Realization, Dependency.

**Identified classes:** User, Admin, Hospital, Hospital Staff, Doctor.

# CHAPTER-4

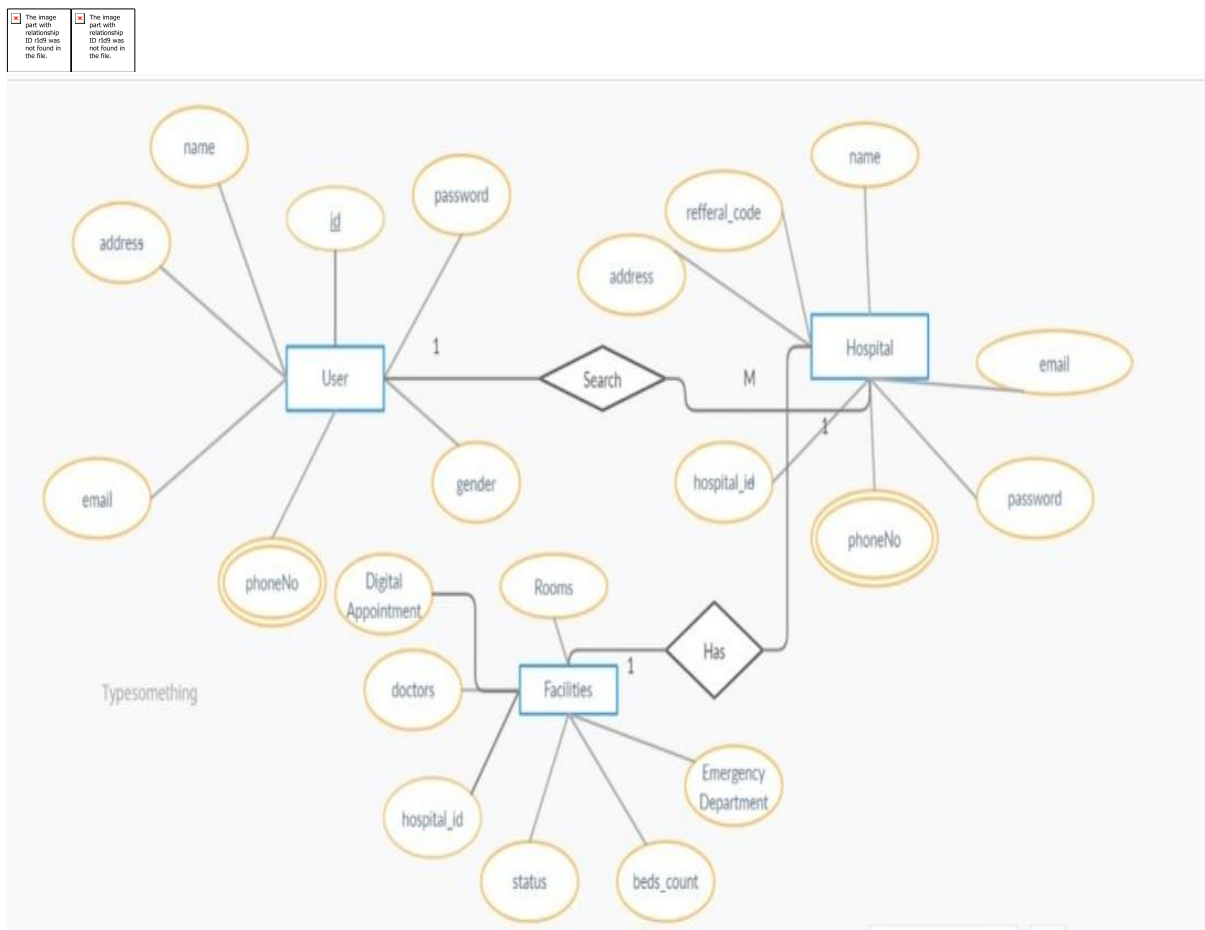
## SYSTEM DESIGN

### 4.1 INTRODUCTION

**Model:** A model is nothing but simplified representation of thing/product.

**UML:** UML is a language for Visualizing, Specifying, Constructing, and Documenting artifacts of a software projects.

### 4.2 Module Design and Organization



**E-R Diagram of Hospital System** An entity-relationship diagram (ERD) is an abstract and conceptual representation of data. Entity relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion.

### 4.3 UML DIAGRAMS:

**Diagram:** “The group of things and relationship is known as diagrams”. To better understand to a model (or) a system there are different in UML each diagram provides different information about system. Every software contains structural aspects as well as behavioural aspects to represent this the diagrams.

Diagrams are categorized into two parts.

- Static/structural diagram
- Dynamic/behavioural diagram

#### STATIC DIAGRAMS:

- 1) Class diagram

#### DYNAMIC DIAGRAMS:

- 2) Use Case diagram
- 3) Sequence diagram
- 4) Activity diagram

→**Things** are divided into four types

- Structural things
- Behavioural thing
- Grouping thing
- Annotation thing

→**Relationships** are divided into

- Association relationship
- Generalization relationship
- Realization relationship
- Dependency relationship

## STATIC DIAGRAMS

**CLASS:** Class is a collection of similar objects (or) class is a collection of attributes, operation, behaviour and relationship.

**CLASS DIAGRAM:** It is used to represent the static behavior of a system or it is used to model the structure of a software.

#### REQUIRED CLASSES:

User, Admin, Hospital, HospitalStaff, Doctor.

## REQUIRED INTERFACES:

System.

## DESCRIPTION:

User goes through our website and if he is new user of our website then he need to register into the website, if he already registered he will login.

--

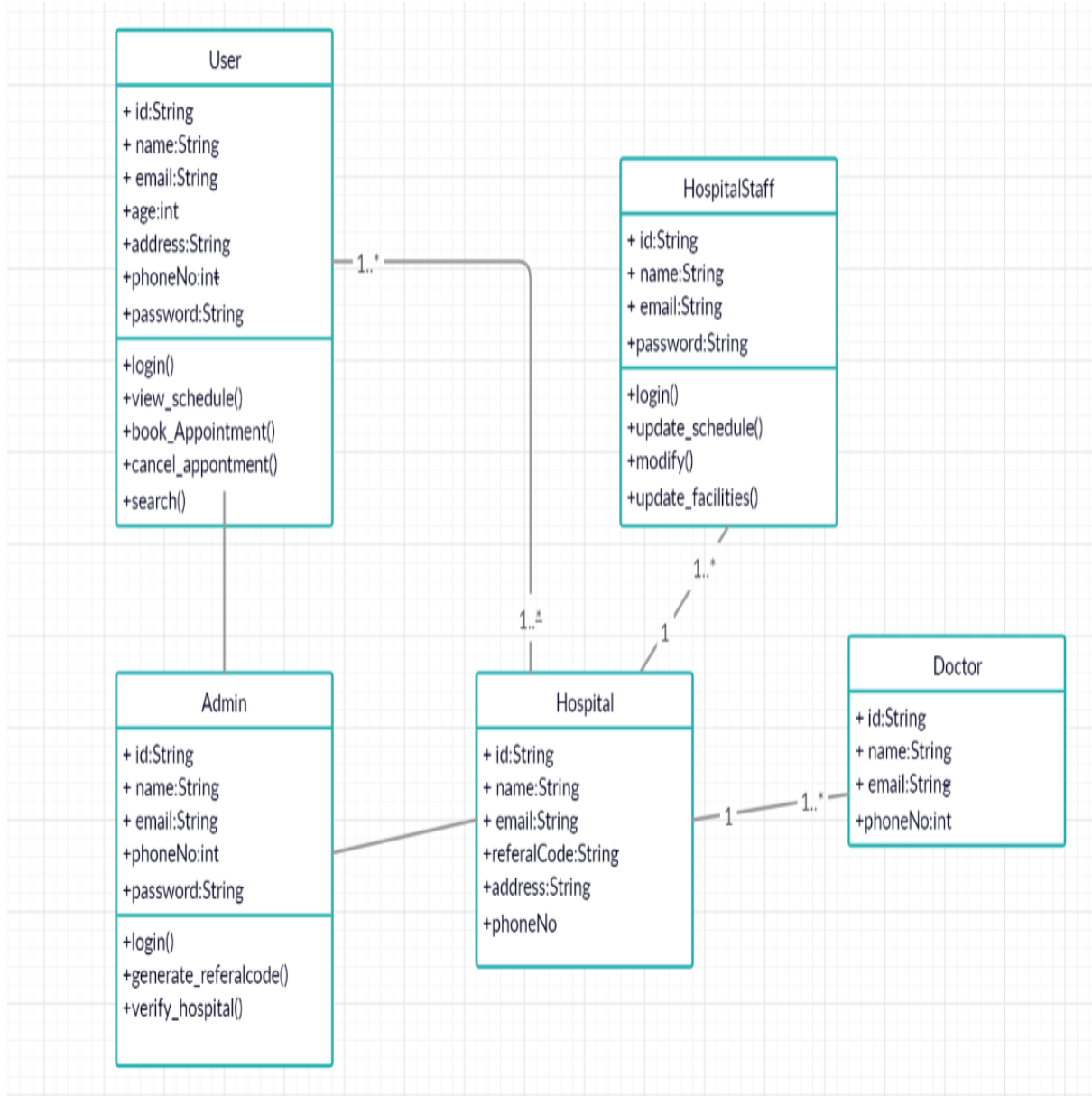


Figure 4.3.1: Class Diagram

## DYNAMIC DIAGRAMS

### SEQUENCE DIAGRAM:

Sequence diagram is one of the two interaction diagrams. It emphasizes on time ordering of messages.

→ In sequence diagram the objects that are participated in interaction are arranged top long the x-axis.

→The first object always initializes the interaction.

### REQUIREMENTS:

User, Website, Admin.

### DESCRIPTION:

The User will be able to login website then admin will verified it after user will get notification as login is successful. user can search for the best hospital in that locality then book the appointment based on doctor schedule. After booking the appointment logout from the website.

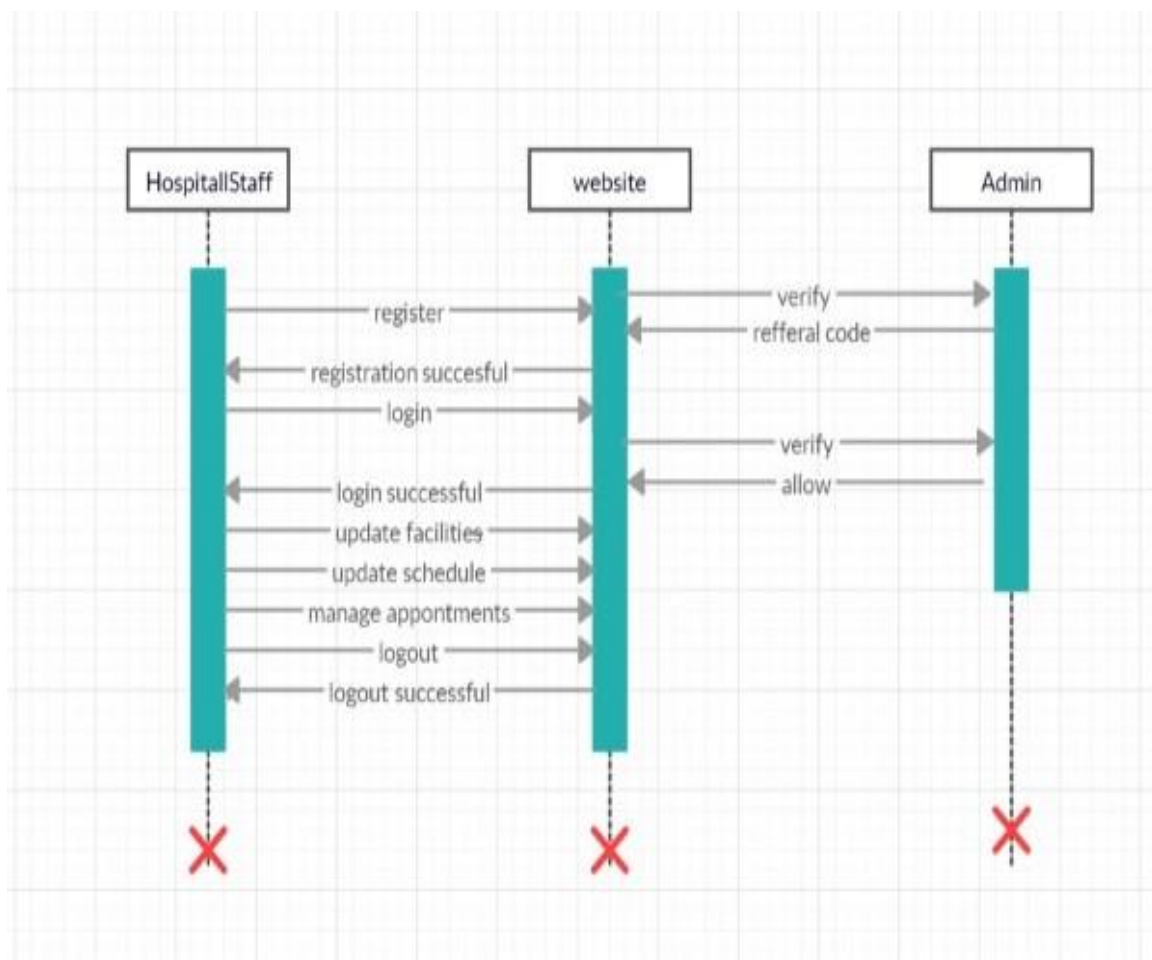


Figure 4.3.2: Sequence diagram



## USECASE DIAGRAM:

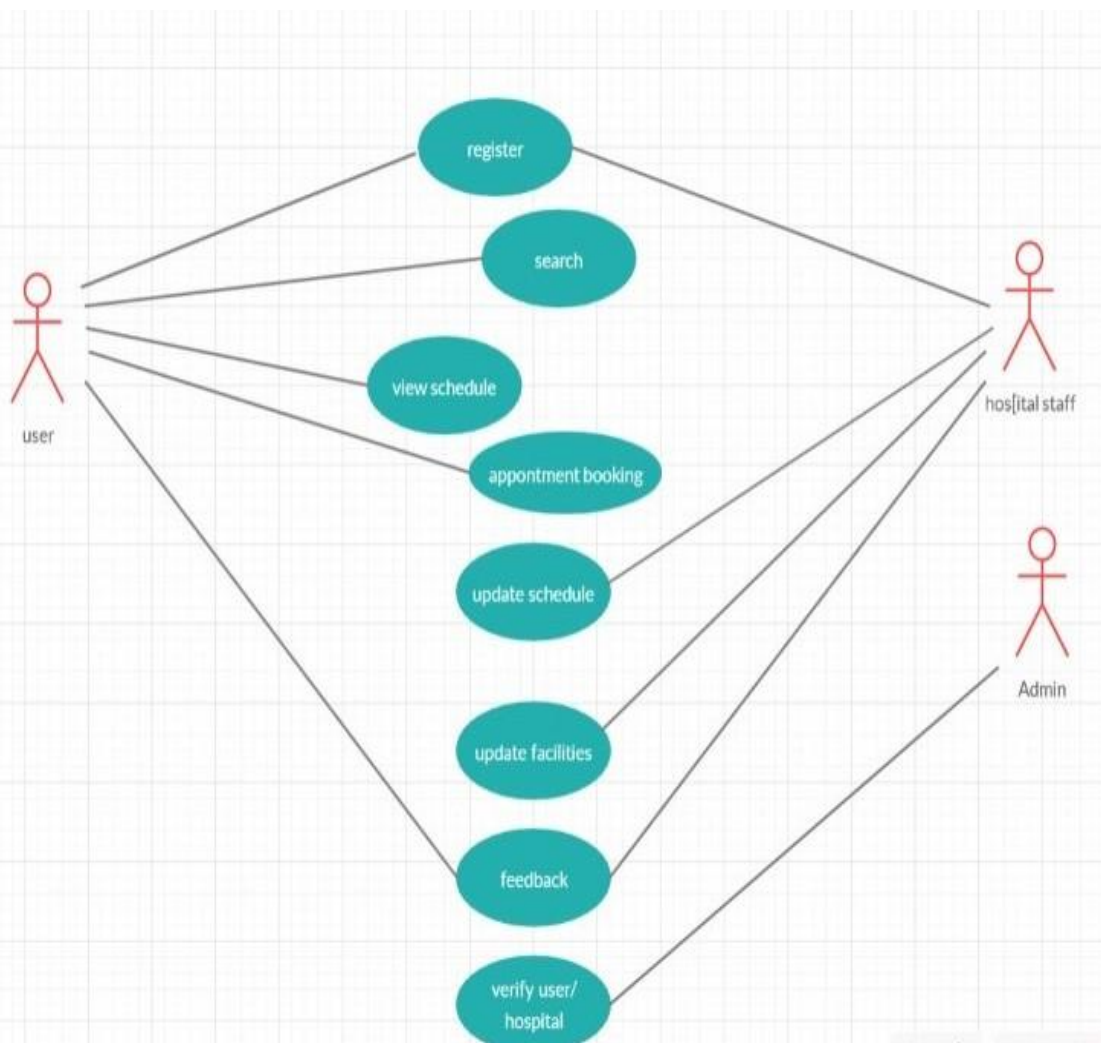
A use case diagram describes a set of interactions between an actor and the system in order to achieve a particular goal. Prompted by some kind of problem for an actor. A use case diagram contains a set of use cases and can developed from by telling others of how a system will be from differing goals.

## REQUIREMENTS:

Register, search, view schedule, appointment booking, update schedule, update facilities, feedback.

## DESCRIPTION:

The user will be able to register in this account using his/her credentials, After th at hospital staff verify login details of the user. then user search for the best hospital then book the appointment based on schedule. hospital staff update the facilities and user details. Final user will give the feedback.



**Figure 4.3.3: Usecase diagram**

## ACTIVITY DIAGRAM:

Activity diagram is used to represent the flow of execution from one activity to another activity. Activity diagram is a type of diagram which illustrate the business and operational step by step workflow of component within a system and shows the overflow of control.

## REQUIREMENTS:

Login page, Register, Make an appointment, Make a selection from available appointments, logout.

## DESCRIPTION:

Activity diagram of hospital finder which shows the flow between the activity of costumer

The admin will be able to login in this account using his/her credentials, After the login, Admin can manage all the operation like add the user details, delete the user details based on time. User can register and login and book the appointment.

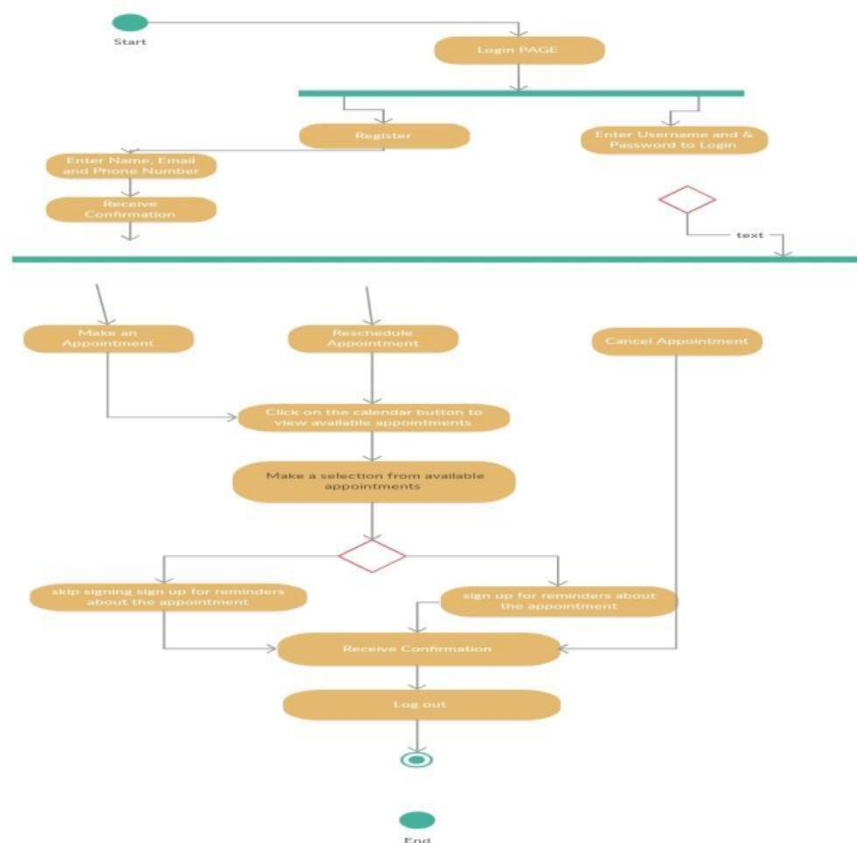


Figure 4.3.4:Activity diagram

### **4.3 CONCLUSION**

Based on all the diagrams we are able to design the required functionalities and the flow of data that is to be maintained between each of them. By doing all this we are able to maintain the application without any bugs and errors. All the diagrams that are developed show us the functionalities of the website.

## CHAPTER-5

### IMPLEMENTATION AND RESULTS

#### 5.1 INTRODUCTION

The Hospital finder can be accessed using username and password. User and the hospital staff will be provided logins. Only authorized hospital staff can login and update the details of the hospital in the website. User can search for hospitals based on speciality , location, specialized doctors available in the hospital. A user can fill the registration form and then they can book appointment. So that he/she can plan their visit to hospital based on the schedule. This web application provides details of the hospital like location, number of beds available, displays number of appointments already booked on that particular day and doctors availability.

#### 5.2 IMPLEMENTATION OF KEY FUNCTIONS

Our project mainly have two modules one for the user and the hospital. The use can register in the website and search for hospitals, view day-to-day updates and doctors availability. The hospital staff can update the details by using a code given to the hospital.

#### 5.3 METHOD OF IMPLEMENTATION

Index.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <link rel="stylesheet" href="index.css">
    <meta charset="utf-8">
    <title>Home Page</title>
  </head>
  <body>
    <div class="A">
      <marquee behavior="alternate" direction="right" scrollamount="2">
        
        
        
      </marquee>
    </div>
    <div class="B">
      <a href="index.html">Home</a>
```

```

<a href="about.html">about</a>
<a href="contact.html">contact</a>
<a href="hospital.html">Hospitals</a>
<a href="login.html">login</a>
</div>
<div class="C">
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
  <div class="x fade">
    
  </div>
</div>

```

```

<br>
<div class="y">
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
  <span class="dot"></span>
</div>
<script type="text/javascript" src="index.js"></script>
</body>
<footer>
  <p>All rights reserved</p>
  <p>email us at abcd@1234.gmail.com</p>
</footer>
</html>

```

Index.css

```

body{
}
.A{
  height: 150px;
}
.B{
  height: 50px;
  display: flex;
  background-color: green;
  text-align: center;
  justify-content: center;
  justify-content: space-between;
}
.A img{
  width: 550px;
  height: 150px;
}
.x img{

```

```

height: 600px;
width: 100%;
vertical-align: middle;
}
.y
{
    text-align: center;;
}
.x{
    display: none;
}
a{
    padding: 15px;
    margin: 5px;
    font-size: 25px;
}
a:link {
    color: white;
    background-color: transparent;
    text-decoration: none;
}
a:visited {
    color: white;
}
a:hover {
    color: yellow;
    background-color: transparent;
    text-decoration: underline;
}
.slideshow-container {
    position: relative;
    margin: auto;
}

.dot {
    height: 15px;
    width: 15px;
    margin: 0 2px;
    background-color: #bbb;
    border-radius: 50%;
    display: inline-block;
    transition: background-color 0.6s ease;
}

```

```

.active {
  background-color: #717171;
}
.fade {
  -webkit-animation-name: fade;
  -webkit-animation-duration: 1.5s;
  animation-name: fade;
  animation-duration: 1.5s;
}
footer{
  text-align: center;
  background-color: #ECFEED;
}

```

## Index.js

```

var slideIndex = 0;
showSlides();
function showSlides() {
  var i;
  var slides = document.getElementsByClassName("x");
  var dots = document.getElementsByClassName("dot");
  for (i = 0; i < slides.length; i++) {
    slides[i].style.display = "none";
  }
  slideIndex++;
  if (slideIndex > slides.length) {slideIndex = 1}
  for (i = 0; i < dots.length; i++) {
    dots[i].className = dots[i].className.replace(" active", "");
  }
  slides[slideIndex-1].style.display = "block";
  dots[slideIndex-1].className += " active";
  setTimeout(showSlides, 3000);
}

```

## Contact.html

```

<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
  <link rel="stylesheet" href="contact.css">
  <meta charset="utf-8">
  <title></title>

```



```

</head>
<body>
  <div class="y">
    <h1>Contact Us</h1>
    <div class="x">
      <h3>email-id:</h3>
      <p>hithaishiyarragudi11@gmail.com</p>
      <p>greeshma11@gmail.com</p>
      <h3>contact numbers:</h3>
      <p>999998888</p>
      <p>888889999</p>
      <p>777775555</p>
    </div>
  </div>
</body>
</html>

```

contact.css

```

body{
  background-image: url('images/9.png');
  background-size: cover;
  text-align: center;
}
.x{
  background-color:black;
  opacity: 0.6;
  filter: alpha(opacity=60);
  top: 15%;
  left: 35%;
  position:absolute;
  width:30%;
  height: 80%;
  position:absolute;
  text-align: center;
}
h1{
  color: #465E23;
  font-weight: bold;
  font-size: 50px;
  text-shadow: 0 0 3px black;
}
h3{

```

```

    color:#F90BB0;
    font-weight: bold;
    font-size: 40px;
}
p{
    color: white;
    font-size: 25px;
    font-weight: bold;
}

```

login.php

```

<?php
use Phppot\Member;

if (! empty($_POST["login-btn"])) {
    require_once __DIR__ . '/Model/Member.php';
    $member = new Member();
    $loginResult = $member->loginMember();
}
?>
<HTML>
<HEAD>
<TITLE>Login</TITLE>
<link href="assets/css/phppot-style.css" type="text/css"
    rel="stylesheet" />
<link href="assets/css/user-registration.css" type="text/css"
    rel="stylesheet" />
<script src="vendor/jquery/jquery-3.3.1.js" type="text/javascript"></script>
</HEAD>
<BODY>
    <div class="phppot-container">
        <div class="sign-up-container">
            <div class="login-signup">
                <a href="user-registration.php">Sign up</a>
            </div>
            <div class="signup-align">
                <form name="login" action="" method="post"
                    onsubmit="return loginValidation()">
                    <div class="signup-heading">Login</div>
                    <?php if(!empty($loginResult)){ ?>
                    <div class="error-msg"><?php echo
$loginResult;?></div>

```

```

        <?php }?>
        <div class="row">
            <div class="inline-block">
                <div class="form-label">
                    Username<span
class="required error" id="username-info"></span>
                </div>
                <input class="input-box-330"
type="text" name="username"
                    id="username">
            </div>
        </div>
        <div class="row">
            <div class="inline-block">
                <div class="form-label">
                    Password<span
class="required error" id="login-password-info"></span>
                </div>
                <input class="input-box-330"
type="password"
                    name="login-password"
id="login-password">
            </div>
        </div>
        <div class="row">
            <input class="btn" type="submit"
name="login-btn"
                    id="login-btn" value="Login">
        </div>
    </form>
</div>
</div>
</div>
</div>
<script>
function loginValidation() {
    var valid = true;
    $("#username").removeClass("error-field");
    $("#password").removeClass("error-field");

    var UserName = $("#username").val();
    var Password = $('#login-password').val();

```

```

$("#username-info").html("").hide();

if (UserName.trim() == "") {
    $("#username-info").html("required.").css("color",
"#ee0000").show();
    $("#username").addClass("error-field");
    valid = false;
}
if (Password.trim() == "") {
    $("#login-password-info").html("required.").css("color",
"#ee0000").show();
    $("#login-password").addClass("error-field");
    valid = false;
}
if (valid == false) {
    $('.error-field').first().focus();
    valid = false;
}
return valid;
}
</script>
</BODY>
</HTML>

```

user-registration.php

```

<?php
use Phppot\Member;
if (! empty($_POST["signup-btn"])) {
    require_once './Model/Member.php';
    $member = new Member();
    $registrationResponse = $member->registerMember();
}
?>
<HTML>
<HEAD>
<TITLE>User Registration</TITLE>
<link href="assets/css/phppot-style.css" type="text/css"
    rel="stylesheet" />
<link href="assets/css/user-registration.css" type="text/css"
    rel="stylesheet" />
<script src="vendor/jquery/jquery-3.3.1.js" type="text/javascript"></script>
</HEAD>

```

```

<BODY>
    <div class="phppot-container">
        <div class="sign-up-container">
            <div class="login-signup">
                <a href="index.php">Login</a>
            </div>
            <div class="">
                <form name="sign-up" action="" method="post"
                    onsubmit="return signupValidation()">
                    <div class="signup-
heading">Registration</div>
                        <?php
if (! empty($registrationResponse["status"])) {
    ?>
        <?php
if ($registrationResponse["status"] == "error") {
    ?>
        <div class="server-response error-msg"><?php
echo $registrationResponse["message"]; ?></div>
            <?php
        } else if ($registrationResponse["status"] == "success") {
            ?>
                <div class="server-response success-msg"><?php echo
$registrationResponse["message"]; ?></div>
                    <?php
                }
            ?>
                <?php
            }
        ?>
            <div class="error-msg" id="error-msg"></div>
            <div class="row">
                <div class="inline-block">
                    <div class="form-label">
                        Username<span
class="required error" id="username-info"></span>
                    </div>
                    <input class="input-box-330"
type="text" name="username"
                        id="username">
                    </div>
                </div>
            <div class="row">

```

```

                                <div class="inline-block">
                                    <div class="form-label">
                                        Email<span class="required
error" id="email-info"></span>
                                </div>
                                <input class="input-box-330"
type="email" name="email" id="email">
                                </div>
                            </div>
                            <div class="row">
                                <div class="inline-block">
                                    <div class="form-label">
                                        Password<span
class="required error" id="signup-password-info"></span>
                                    </div>
                                    <input class="input-box-330"
type="password"
                                        name="signup-password"
id="signup-password">
                                </div>
                            </div>
                            <div class="row">
                                <div class="inline-block">
                                    <div class="form-label">
                                        Confirm Password<span
class="required error"
                                        id="confirm-
password-info"></span>
                                    </div>
                                    <input class="input-box-330"
type="password"
                                        name="confirm-password"
id="confirm-password">
                                </div>
                            </div>
                            <div class="row">
                                <input class="btn" type="submit"
name="signup-btn"
                                id="signup-btn" value="Sign up">
                            </div>
                        </form>
                    </div>
                </div>
            </div>

```

</div>

<script>

```
function signupValidation() {
    var valid = true;

    $("#username").removeClass("error-field");
    $("#email").removeClass("error-field");
    $("#password").removeClass("error-field");
    $("#confirm-password").removeClass("error-field");

    var UserName = $("#username").val();
    var email = $("#email").val();
    var Password = $('#signup-password').val();
    var ConfirmPassword = $('#confirm-password').val();
    var emailRegex = /^[a-zA-Z0-9.!#$%&'*/+=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$/;

    $("#username-info").html("").hide();
    $("#email-info").html("").hide();

    if (UserName.trim() == "") {
        $("#username-info").html("required.").css("color",
"#ee0000").show();
        $("#username").addClass("error-field");
        valid = false;
    }
    if (email == "") {
        $("#email-info").html("required").css("color", "#ee0000").show();
        $("#email").addClass("error-field");
        valid = false;
    } else if (email.trim() == "") {
        $("#email-info").html("Invalid email address.").css("color",
"#ee0000").show();
        $("#email").addClass("error-field");
        valid = false;
    } else if (!emailRegex.test(email)) {
        $("#email-info").html("Invalid email address.").css("color",
"#ee0000")
            .show();
        $("#email").addClass("error-field");
        valid = false;
    }
}
```

```

    }
    if (Password.trim() == "") {
        $("#signup-password-info").html("required.").css("color",
"#ee0000").show();
        $("#signup-password").addClass("error-field");
        valid = false;
    }
    if (ConfirmPassword.trim() == "") {
        $("#confirm-password-info").html("required.").css("color",
"#ee0000").show();
        $("#confirm-password").addClass("error-field");
        valid = false;
    }
    if(Password != ConfirmPassword){
        $("#error-msg").html("Both passwords must be same.").show();
        valid=false;
    }
    if (valid == false) {
        $('.error-field').first().focus();
        valid = false;
    }
    return valid;
}
</script>
</BODY>
</HTML>

```

user-registration.css

```

.sign-up-container {
    border: 1px solid;
    border-color: #9a9a9a;
    background: #fff;
    border-radius: 4px;
    padding: 10px;
    width: 350px;
    margin: 50px auto;
}

.page-header {
    float: right;
}

```



```
.login-signup {  
    margin: 10px;  
    text-decoration: none;  
    float: right;  
}
```

```
.login-signup a {  
    text-decoration: none;  
    font-weight: 700;  
}
```

```
.signup-heading {  
    font-size: 2em;  
    font-weight: bold;  
    padding-top: 60px;  
    text-align: center;  
}
```

```
.inline-block {  
    display: inline-block;  
}
```

```
.row {  
    margin: 15px 0px;  
    text-align: center;  
}
```

```
.form-label {  
    margin-bottom: 5px;  
    text-align: left;  
}
```

```
input.input-box-330 {  
    width: 250px;  
}
```

```
.sign-up-container .error {  
    color: #ee0000;  
    padding: 0px;  
    background: none;  
    border: #ee0000;  
}
```

```
.sign-up-container .error-field {  
    border: 1px solid #d96557;  
}
```

```
.sign-up-container .error:before {  
    content: '*';  
    padding: 0 3px;  
    color: #D8000C;  
}
```

```
.error-msg {  
    padding-top: 10px;  
    color: #D8000C;  
    text-align: center;  
}
```

```
.success-msg {  
    padding-top: 10px;  
    color: #176701;  
    text-align: center;  
}
```

```
input.btn {  
    width: 250px  
}
```

```
.signup-align {  
    margin: 0 auto;  
}
```

```
.page-content {  
    font-weight: bold;  
    padding-top: 60px;  
    text-align: center;  
}
```

member.php

```
<?php  
namespace Phppot;
```

```
class Member
```

```

{

private $ds;

function __construct()
{
    require_once __DIR__ . '/../lib/DataSource.php';
    $this->ds = new DataSource();
}
public function isUsernameExists($username)
{
    $query = 'SELECT * FROM tbl_member where username = ?';
    $paramType = 's';
    $paramValue = array(
        $username
    );
    $resultArray = $this->ds->select($query, $paramType, $paramValue);
    $count = 0;
    if (is_array($resultArray)) {
        $count = count($resultArray);
    }
    if ($count > 0) {
        $result = true;
    } else {
        $result = false;
    }
    return $result;
}

public function isEmailExists($email)
{
    $query = 'SELECT * FROM tbl_member where email = ?';
    $paramType = 's';
    $paramValue = array(
        $email
    );
    $resultArray = $this->ds->select($query, $paramType, $paramValue);
    $count = 0;
    if (is_array($resultArray)) {
        $count = count($resultArray);
    }
    if ($count > 0) {
        $result = true;
    }
}
}

```

```

    } else {
        $result = false;
    }
    return $result;
}

public function registerMember()
{
    $isUsernameExists = $this->isUsernameExists($_POST["username"]);
    $isEmailExists = $this->isEmailExists($_POST["email"]);
    if ($isUsernameExists) {
        $response = array(
            "status" => "error",
            "message" => "Username already exists."
        );
    } else if ($isEmailExists) {
        $response = array(
            "status" => "error",
            "message" => "Email already exists."
        );
    } else {
        if (! empty($_POST["signup-password"])) {

            // PHP's password_hash is the best choice to use to store passwords
            // do not attempt to do your own encryption, it is not safe
            $hashedPassword = password_hash($_POST["signup-password"],
PASSWORD_DEFAULT);
        }
        $query = 'INSERT INTO tbl_member (username, password, email)
VALUES (?, ?, ?)';
        $paramType = 'sss';
        $paramValue = array(
            $_POST["username"],
            $hashedPassword,
            $_POST["email"]
        );
        $memberId = $this->ds->insert($query, $paramType, $paramValue);
        if (! empty($memberId)) {
            $response = array(
                "status" => "success",
                "message" => "You have registered successfully."
            );
        }
    }
}

```

```

    }
    return $response;
}

public function getMember($username)
{
    $query = 'SELECT * FROM tbl_member where username = ?';
    $paramType = 's';
    $paramValue = array(
        $username
    );
    $memberRecord = $this->ds->select($query, $paramType, $paramValue);
    return $memberRecord;
}

public function loginMember()
{
    $memberRecord = $this->getMember($_POST["username"]);
    $loginPassword = 0;
    if (! empty($memberRecord)) {
        if (! empty($_POST["login-password"])) {
            $password = $_POST["login-password"];
        }
        $hashedPassword = $memberRecord[0]["password"];
        $loginPassword = 0;
        if (password_verify($password, $hashedPassword)) {
            $loginPassword = 1;
        }
    } else {
        $loginPassword = 0;
    }
    if ($loginPassword == 1) {
        // login success so store the member's username in
        // the session
        session_start();
        $_SESSION["username"] = $memberRecord[0]["username"];
        session_write_close();
        $url = "./home.php";
        header("Location: $url");
    } else if ($loginPassword == 0) {
        $loginStatus = "Invalid username or password.";
        return $loginStatus;
    }
}

```

```
}  
}
```

type.html

```
<!DOCTYPE html>  
<html lang="en" dir="ltr">  
  <head>  
    <meta charset="utf-8">  
    <title></title>  
    <link rel="stylesheet" href="type.css">  
  </head>  
  <body>  
    <div class="A">  
      <h3>Successfully Booked</h3>  
      <a href="index.html">go home</a>  
    </div>  
  </body>  
</html>
```

type.css

```
body{  
  background-image: url('images/c.png');  
  background-size: cover;  
}  
.A{  
  width:25%;  
  height: 50%;  
  background:white ;  
  border-radius: 3%;  
  top: 45%;  
  left: 45%;  
  display: grid;  
  position:absolute;  
  transform: translate(-40%,-40%);  
  text-align: center;  
  justify-content: between;  
}
```

update.html

```

<!DOCTYPE html>
<html>
<head>
<style>
body{
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}
* {box-sizing: border-box}
input[type=text], input[type=password] {
    width: 100%;
    font-size: 28px;
    padding: 15px;
    margin: 5px 0 22px 0;
    display: inline-block;
    border: none;
    background: #f1f1f1;
}
label{
    font-size: 15px;
}
input[type=text]:focus, input[type=password]:focus {
    background-color: #ddd;
    outline: none;
}
hr {
    border: 1px solid #f1f1f1;
    margin-bottom: 25px;
}
a {
    font-size: 18px;
    font-weight: bold;
    background-color: rgb(10, 119, 13);
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 100%;
    opacity: 0.9;
}
a:hover {
    opacity:1;
}

```

```

.cancel {
  padding: 14px 20px;
  background-color: #ff3d2f;
}
.formContainer {
  padding: 16px;
}
.formContainer p{
  font-size: 28px;
}
</style>
<body>
<form>
<div class="formContainer">
<h1>Hospital details</h1>
<hr>
<label for="email"><b>Hospital code</b></label>
<input type="text" placeholder="Enter code" name="email" required>
<label for="email"><b>docor availability</b></label>
<input type="text" placeholder="YES/NO" name="email" required>
<label for="password"><b>Appointments booked</b></label>
<input type="password" placeholder="Enter count" name="password"
required>
<label for="repeatPassword"><b>Beds availability</b></label>
<input type="password" placeholder="count" name="repeatPassword"
required>
<div>
<a href="index.html">Cancel</a>
<a href="s.html">Update</a>
</div>
</div>
</form>
</body>
</html>

```

contact.html

```

<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
  <link rel="stylesheet" href="contact.css">
  <meta charset="utf-8">

```



```

<title></title>
</head>
<body>
  <div class="y">
    <h1>Contact Us</h1>
    <div class="x">
      <h3>email-id:</h3>
      <p>hithaishiyarragudi11@gmail.com</p>
      <p>greeshma11@gmail.com</p>
      <h3>contact numbers:</h3>
      <p>999998888</p>
      <p>888889999</p>
      <p>777775555</p>
    </div>
  </div>
</body>
</html>

```

contact.css

```

body{
  background-image: url('images/9.png');
  background-size: cover;
  text-align: center;
}
.x{
  background-color:black;
  opacity: 0.6;
  filter: alpha(opacity=60);
  top: 15%;
  left: 35%;
  position:absolute;
  width:30%;
  height: 80%;
  position:absolute;
  text-align: center;
}
h1{
  color: #465E23;
  font-weight: bold;
  font-size: 50px;
}

```

```

    text-shadow: 0 0 3px black;
}
h3{
    color:#F90BB0;
    font-weight: bold;
    font-size: 40px;
}
p{
    color: white;
    font-size: 25px;
    font-weight: bold;
}

```

data.html

```

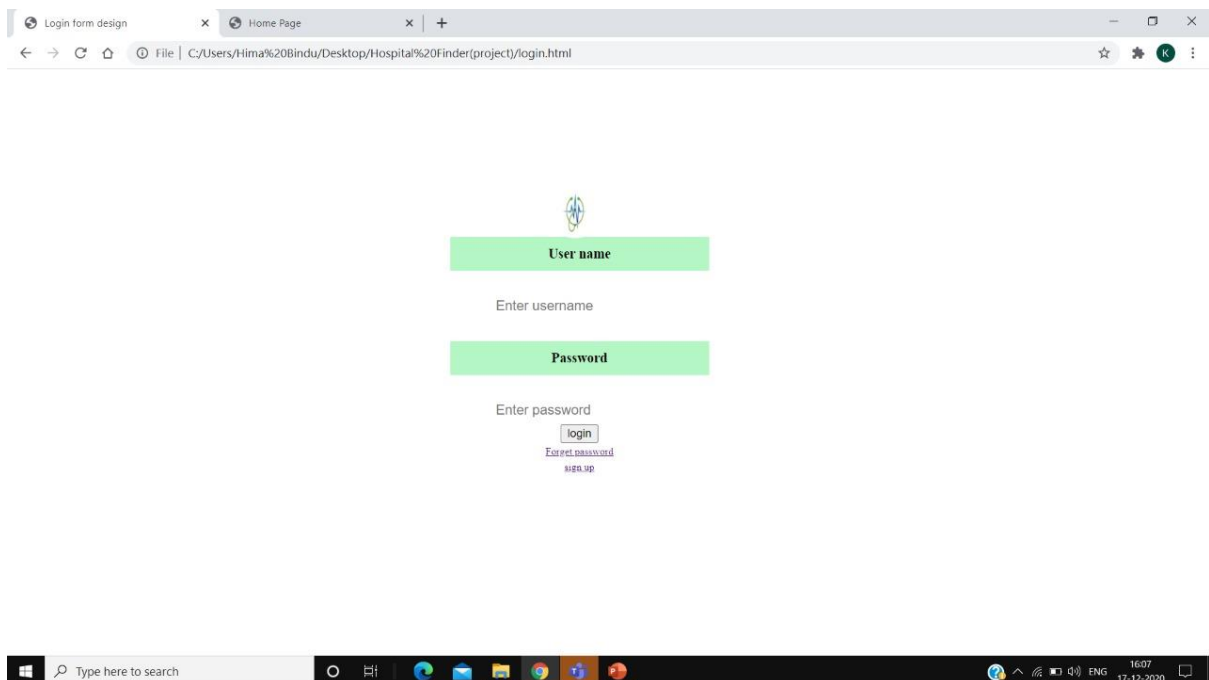
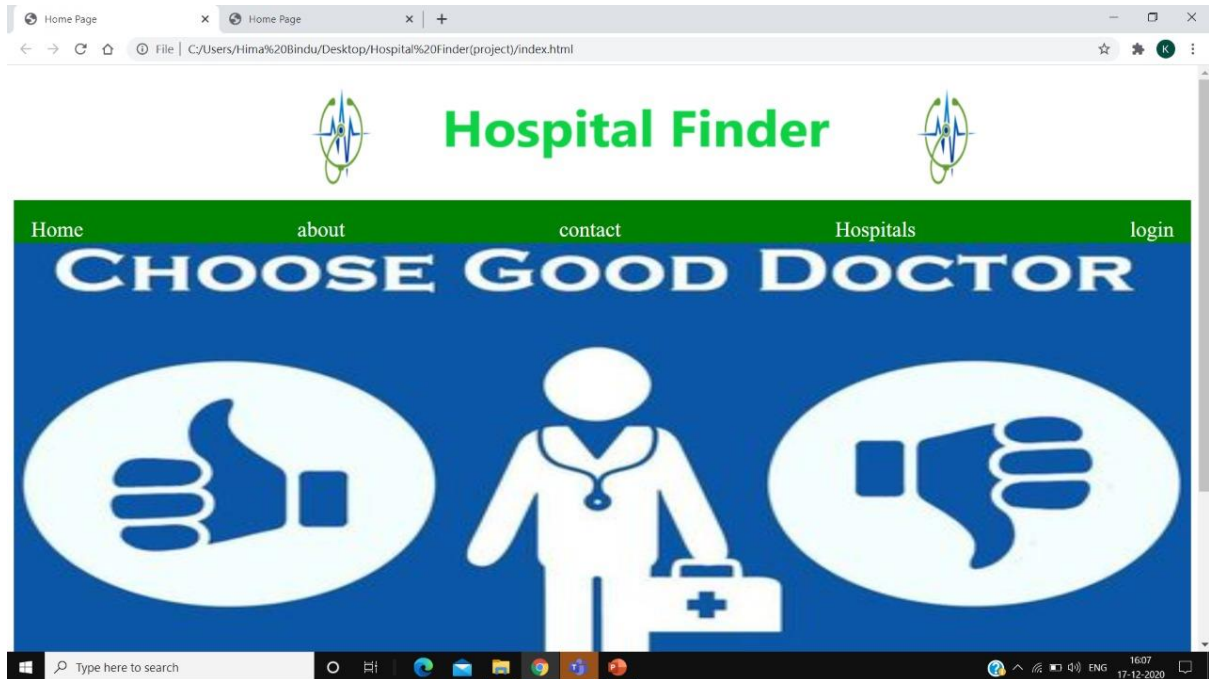
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <link rel="stylesheet" href="data.css">
    <meta charset="utf-8">
    <title>Home Page</title>
  </head>
  <body>
    <div class="A">
      <marquee behavior="alternate" direction="right" scrollamount="2">
        
        
        
      </marquee>
    </div>
    <div class="B">
      <a href="appointment.html">Book Appointment</a>
      <a href="update.html">update</a>
    </div>
    <div class="C">
      <h1>Santharam Hospital</h1>
      <h2>Doctor Availibility:YES</h2>
      <h2>Appointments count:15</h2>
      <h2>Beds availabe:5</h2>
    </div>
  </html>

```

data.css

```
body{
}
.A{
  height: 150px;
}
.B{
  height: 50px;
  display: flex;
  background-color: green;
  text-align: center;
  justify-content: center;
  justify-content: space-between;
}
.A img{
  width: 550px;
  height: 150px;
}
a{
  padding: 15px;
  margin: 5px;
  font-size: 25px;
}
a:link {
  color: white;
  background-color: transparent;
  text-decoration: none;
}
a:visited {
  color: white;
}
a:hover {
  color: yellow;
  background-color: transparent;
  text-decoration: underline;
}
.C {
  text-align: center;
}
h1{
  color: blue;
}
```

## 5.4 OUTPUT SCREENS AND RESULT ANALYSIS:



form.html

File | C:/Users/Hima%20Bindu/Desktop/Hospital%20Finder(project)/form.html

## Sign Up Form

Email

absghs23@gmail.com

Password

Enter Password

Repeat Password

Repeat Password

☒ Remember me

By creating an account you agree to our [Terms & Privacy](#).

Cancel

Sign Up

Type here to search


16:14 17-12-2020


hospital.html

File | C:/Users/Hima%20Bindu/Desktop/Hospital%20Finder(project)/hospital.html

## Search for Hospitals

search here

  
*Santharam Hospital*

  
Guptha Hospital

File:///C:/Users/Hima Bindu/Desktop/Hospital Finder(project)/hospital.html

Type here to search

16:12 17-12-2020

appointment.html

File | C:/Users/Hima%20Bindu/Desktop/Hospital%20Finder(project)/appointment.html

Hithaishi

Email

Phone

Special notes, concerns, or requirements

What is the best way to reach you?

☐ Phone  
☐ Email

Days of the week you are available for appointment:

☐ Monday  
☐ Tuesday  
☐ Wednesday  
☐ Thursday  
☐ Friday

Best time of day for your appointment:

☐ Morning  
☐ Afternoon

[Book](#)

Type here to search

16:15  
17-12-2020

Hospital details

Hospital code

Enter code

docor availability

YES/NO

Appointments booked

Enter count

Beds availability

count

Cancel Update

Type here to search

11:28  
31-01-2021

## 5.4 CONCLUSION

Above are the proposed Hospital Finder system implementation along with results we got by implementing the source code. With this type of system, we can easily get the clarity about the Hospital Finder system which we are doing and it looks Smart during the Usage.

# **CHAPTER-6**

## **TESTING AND VALIDATION**

### **6.1 INTRODUCTION**

#### **INTRODUCTION TO TESTING**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development

#### **UNIT TESTING**

The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of software. Unit Testing is a level of software testing where individual units/components of a software are tested. Typically the unit test will establish some sort of artificial environment and then invoke methods in the unit being tested. It then checks the results returned against some known value. When the units are assembled we can use the same tests to test the system as a whole. It usually has one or a few inputs and usually a single output.

#### **FUNCTIONAL TESTING**

Functional Testing is a testing technique that is used to test the features /functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases. For example: Testing of backend storage and Testing of navigation transmission.

#### **INTEGRATION TESTING**

Upon completion of unit testing, the units or modules are to be integrated which gives rise to integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated. Similarly, every unit is integrated after the testing of every single unit is done individually.

#### **SYSTEM TESTING**

System testing of software or hardware is the testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. The hardware and the software units are tested separately and then tested together to check if the desired results are obtained.

### **Black Box testing**

Black Box Testing is a software Testing method in which testers evaluate the functionality of the software under test without looking at the internal code structure. This can be applied to every level of software testing such as Unit, Integration, System and Acceptance Testing.

It is also called as Behavioral/Specification-Based/Input-Output Testing.

Black Box Testing techniques:

1. Equivalence partitioning
2. Boundary Value Analysis
3. Decision Table
4. State Transition

### **White Box testing**

White Box Testing is based on applications internal code structure. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. This testing usually done at the unit level.

It is also called as Glass Box, Clear Box, Structural Testing.

White Box Testing Techniques:

1. Statement Coverage
2. Branch Coverage
3. Path Coverage

## **PERFORMANCE TESTING**

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.



## **6.2 VALIDATION**

The following test case scenarios were used in the integrated system testing to prove the working of the developed system.

- Appointment booking details checking and server communication checking.
- User and admin valid credentials checking.
- Database storing at backend validation.
- Automatic updating of appointment details.

## **6.3 Conclusion**

All above validations are verified and executed successfully.

# **CHAPTER-7**

## **CONCLUSION**

### **7.1 CONCLUSION**

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
- Any further changes can be easily adaptable.
- Based on the future security issues, security can be improved using emerging technologies.
- Online payment module can be added.

## **CHAPTER 8**

### **REFERENCES**

#### **REFERENCES**

[1]

[https://www.academia.edu/36406675/Hospital\\_Management\\_System\\_Project\\_report](https://www.academia.edu/36406675/Hospital_Management_System_Project_report).

[2]

[https://www.academia.edu/7149341/HOSPITAL\\_MANAGEMENT\\_SYSTEM\\_A\\_PROJECT\\_REPORT\\_Submitted\\_in\\_Partial\\_Fulfillment\\_of\\_the\\_requirements\\_for\\_the\\_Award\\_of\\_the](https://www.academia.edu/7149341/HOSPITAL_MANAGEMENT_SYSTEM_A_PROJECT_REPORT_Submitted_in_Partial_Fulfillment_of_the_requirements_for_the_Award_of_the)

[3]

[https://www.google.com/search?q=html+and+php+database+connections&rlz=1C1SQJL\\_enIN820IN820&oq=html+and+php+database+connections&aqs=chrome..69i57j33i22i29i30l3.2l978j1j7&sourceid=chrome&ie=UTF-8](https://www.google.com/search?q=html+and+php+database+connections&rlz=1C1SQJL_enIN820IN820&oq=html+and+php+database+connections&aqs=chrome..69i57j33i22i29i30l3.2l978j1j7&sourceid=chrome&ie=UTF-8)