

# **"BOOK MY APPOINTMENT"**

*A MINI PROJECT REPORT SUBMITTED TO*

**THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU**

(An Autonomous Institution under VTU)



**In partial fulfillment of the requirements for the fifth semester of the Database  
Management Systems laboratory of  
Bachelor of Engineering  
in  
Computer Science and Engineering**

*Submitted by:*

<b>Rohan Karnik SR</b>	<b>:</b>	<b>4NI15CS078</b>
<b>Saransh Sachdeva</b>	<b>:</b>	<b>4NI15CS087</b>
<b>Raj Abhishek</b>	<b>:</b>	<b>4NI15CS073</b>

Under the Guidance of

**Mr. Narender.M**

Assistant Professor

**Smt. M. R. Rashmi**

Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**THE NATIONAL INSTITUTE OF ENGINEERING**

(Autonomous under VTU)

Manandavadi Road, Mysore-570 008

2017-2018

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING THE  
NATIONAL INSTITUTE OF ENGINEERING**



**CERTIFICATE**

This is to certify that the project work entitled “**Book My Appointment**” is a work carried out by **ROHAN KARNIK SR(4NI15CS078), SARANSH SACHDEVA(4NI15CS087) and RAJ ABHISHEK(4NI15CS073)** for the fifth semester Database Laboratory project work in Computer Science & Engineering, The National Institute of Engineering (Autonomous Institution under Vishveshwaraya Technological University, Belgaum) during the academic year 2017-2018. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the work prescribed for the semester.

Signature of Staff:

1.

.....  
Mr. Narender.M  
(Assistant Professor)

.....  
Dr. H. D. Phaneendra  
(Head of the Department)

2.

.....  
Smt. M. R. Rashmi  
(Assistant Professor)

Department of Computer Science and Engineering  
NIE, Mysuru.

## ACKNOWLEDGEMENTS

I would like to take this opportunity to express out profound gratitude to all those people who were directly or indirectly involved in the completion of this project. I thank each and everyone who encouraged me in every possible way.

I would like to thank our Principal **Dr. G.Ravi** for letting me to be the part of this prestigious institution and letting me to explore my abilities to the fullest.

I would also like to extend my sincere gratitude to **Dr. H. D. Phaneendra**, our H.O.D for being a source of inspiration and instilling an enthusiastic spirit in me throughout the process of project making.

I would like to express my heartfelt gratitude towards my project guides **Mr. Narender.M** and **Smt. M. R. Rashmi** for their constant guidance, valuable knowledge and experience.

I am thankful to my parents, family and friends for their constant support, inspiration and encouragement, for their helping hand and also to all those who supported me directly or indirectly in my academic process.

Thank you.

**-Rohan Karnik SR**  
**-Saransh Sachdeva**  
**-Raj Abhishek**

# **TABLE OF CONTENTS**

<b>1. INTRODUCTION</b>	
1.1    Back end	1
1.2    Database Management System	2
1.3    Front end	2
<b>2. SYSTEM ANALYSIS</b>	
2.1    Existing system	3
2.2    Proposed system	4
<b>3. SYSTEM DESIGN</b>	
3.1    Entity description and attribute details	5
3.2    ER Diagram	9
3.3    Screen shots of description tables	10
<b>4. HARDWARE AND SOFTWARE REQUIREMENTS</b>	
4.1    Software requirements	13
4.2    Hardware requirements	13
<b>5. SYSTEM IMPLEMENTATION</b>	14
<b>6. CODE SNIPPET</b>	
<b>7. TESTING</b>	
<b>8. SCREENSHOTS</b>	
<b>9. CONCLUSION AND FUTURE ENHANCEMENTS</b>	
<b>10. REFERENCES</b>	

## CHAPTER 1

### INTRODUCTION

The “Book My Appointment” has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. Moreover, this project is developed to carry out operations in a smoother and effective manner.

This project is reduced as much as possible to avoid errors while entering data. No formal knowledge is needed for user to use this system. Thus by this all, it proves it is user-friendly.

#### **1.1 BACK END**

The backend development is the part of the application that is never visible to the user. It is built with use of server side language and database. In *simple* words, front end code interacts with a user in real time while back end code interacts with a server to return user ready results. A back-end database stores data but does not include end-user application elements such as stored queries, forms, macros or reports

The backend used here is MySQL on PHP. MySQL is the world’s most widely used open source relational database management system (RDBMS) that runs a server providing a multi-user access to a number of databases. The SQL phrase stands for Structured Query Language.

## **1.2 DATABASE MANAGEMENT SYSTEM**

Database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc.

The software which is used to manage database is called Database Management System (DBMS) that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.

A database is a collection of information that is organized so that it can be easily accessed, managed and updated. Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.

Typically, a database manager provides users with the ability to control read/write access, specify report generation and analyze usage. Some databases offer ACID (atomicity, consistency, isolation and durability) compliance to guarantee that data is consistent and that transactions are complete.

A database is not usually portable across different DBMS, but different DBMS can inter operate by using standards such as MySQL to allow a single application to work with more than one DBMS.

## **1.3 Front end**

Front end typically means the parts of the project a user interacts with-such as the graphical user interface or command line. The front end used here is HTML, PHP and CSS languages.

# **SYSTEM ANALYSIS**

## **2.1 EXISTING SYSTEM**

The existing system of booking appointments with doctors is usually carried out manually which leads to many problems, such as data loss, difficulty in maintaining appointments, difficulty in retrieving of patient's data, data inconsistency, data insecurity, wastage of time in maintaining record manually. The previous history of patient needs to be looked up in hospital's paper record and for that, hospital or clinic providing the service must be contacted in order to gain knowledge about the patient's medical history. One of the major drawbacks is that all the appointments cannot be booked on the phone and even if a person goes to the hospital to book his appointment, it usually takes a lot of time and sometimes it might get cancelled due to some unavoidable circumstances.

Thus, we can conclude that the traditional method of booking appointments is not efficient and thus there is a need of automation of the entire system.

## CHAPTER 2

### 2.2 PROPOSED SYSTEM

This project aims at the automation of doctor appointment system by providing users with a working structure to book their doctor's appointment with user friendly environment.

The database keeps track of the various data required.

This project is divided into the following modules:

1. **Doctor Details:** This module holds the records of the enlisted doctors along with their work experience, qualification, fee structure, phone number and the amount of slots available to get their appointments
2. **Hospital details:** This module holds the record of available hospitals and the slots available along with the specialist doctor in that particular hospital or clinic.
3. **Booking module:** This is the booking module of the system which keeps tracks of the patients and doctors who have already registered to book their appointment or the doctor who is ready to take appointments.  
This module holds the information of the registered user.
4. **Login module:** This module authenticates the user to login to their respective profile using their login ids and password to prevent MySQL injection.



### **SYSTEM DESIGN**

#### **3.1. ENTITY DESCRIPTION AND ATTRIBUTE DETAILS**

##### **person:**

It includes the person details like person name, gender, date of birth, phone number and register date.

person id which is a primary key.

- personID
- firstName
- lastName
- gender
- dob
- phoneNumber
- registerDate

##### **personLogin:**

It includes patient login details like person id (foreign key), email (primary key), password, question, answer and last login.

The password and the security answer are encrypted.

- personID
- email
- password
- question
- answer
- lastLogin

## CHAPTER 3

### **doctor:**

It is a table that holds the basic information of doctor along with necessary information added for doctor profile such as qualification, experience, fees, department, etc.

departmentID and buildingID are the foreign keys.

- doctorID
- firstName
- lastName
- gender
- dob
- phoneNumber
- registerDate
- qualification
- experience
- fee
- departmentID
- buildingID

### **doctorLogin:**

It is a table which contains the login details for the user doctor. It holds the credentials needed to authorize access to the registered user.

doctorID is the foreign key.

- doctorID
- email
- password
- question
- answer
- lastLogin

## CHAPTER 3

### **department:**

It consists of the details of the respective departments to which a registered doctor belongs to.

It has a primary key departmentID.

- name
- departmentID

### **building:**

It holds the details of all the hospitals and clinics that comes under the patients interest of visit.

buildingID is the primary key.

- buildingID
- name
- addressLine 1, 2, 3
- city
- pin
- longitude
- latitude

### **booking:**

This includes records of booking details of a patient who wants to book an appointment with specified doctor.

bookingID is the primary key.

personID and doctorID are the foreign key.

- bookingID
- bookingDate
- appointmentDate
- personID

## CHAPTER 3

- doctorID
- slot
- bookingStatus

### **history:**

This table holds the records of past bookings that the patient or doctor had.

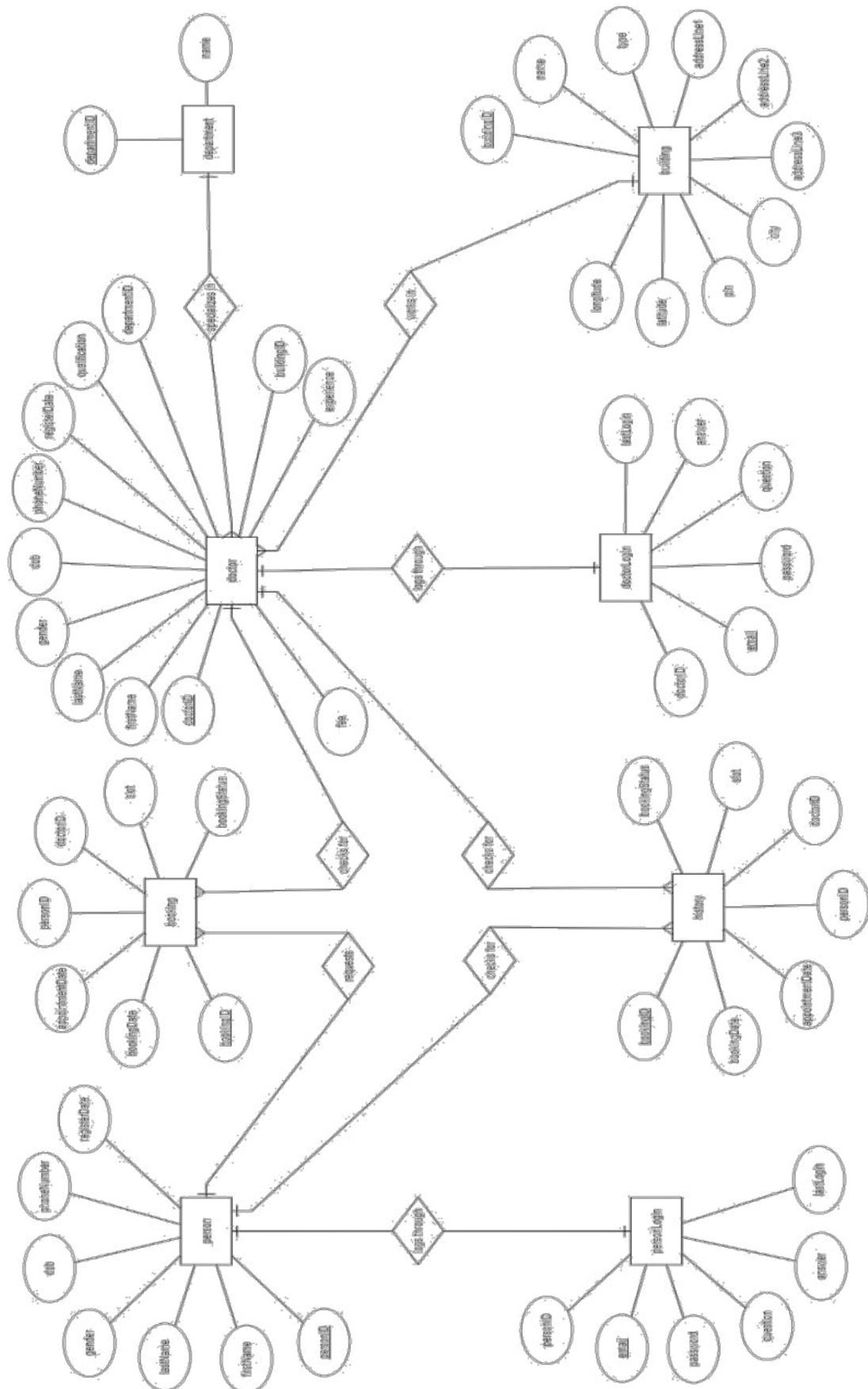
bookingID is the primary key.

personID and doctorID are the foreign key.

- bookingID
- bookingDate
- appointmentDate
- personID
- doctorID
- slot
- bookingStatus

## CHAPTER 3

### 3.2. ER DIAGRAM



## CHAPTER 3

### 3.3 SCREEN SHOTS OF DESCRIPTION TABLES

```
mysql> desc person;
```

Field	Type	Null	Key	Default	Extra
personID	int(11)	NO	PRI	NULL	auto_increment
firstName	varchar(50)	NO		NULL	
lastName	varchar(50)	YES		NULL	
gender	varchar(6)	NO		NULL	
dob	date	NO		NULL	
phoneNumber	bigint(10)	NO		NULL	
registerDate	date	NO		NULL	

7 rows in set (0.00 sec)

```
mysql> desc personLogin;
```

Field	Type	Null	Key	Default	Extra
personID	int(11)	NO	MUL	NULL	
email	varchar(100)	NO	PRI	NULL	
password	varchar(100)	NO		NULL	
question	varchar(100)	NO		NULL	
answer	varchar(100)	NO		NULL	
lastLogin	datetime	NO		NULL	

6 rows in set (0.00 sec)

```
mysql> desc doctor;
```

Field	Type	Null	Key	Default	Extra
doctorID	int(11)	NO	PRI	NULL	auto_increment
firstName	varchar(50)	NO		NULL	
lastName	varchar(50)	YES		NULL	
gender	varchar(6)	NO		NULL	
dob	date	NO		NULL	
phoneNumber	bigint(10)	NO		NULL	
registerDate	date	NO		NULL	
qualification	varchar(100)	NO		NULL	
departmentID	int(11)	NO	MUL	NULL	
buildingID	int(11)	NO	MUL	NULL	
experience	int(2)	YES		NULL	
fee	int(4)	NO		NULL	

12 rows in set (0.00 sec)

## CHAPTER 3

```
mysql> desc doctorLogin;
```

Field	Type	Null	Key	Default	Extra
doctorID	int(11)	NO	MUL	NULL	
email	varchar(100)	NO	PRI	NULL	
password	varchar(100)	NO		NULL	
question	varchar(100)	NO		NULL	
answer	varchar(100)	NO		NULL	
lastLogin	datetime	NO		NULL	

```
6 rows in set (0.00 sec)
```

```
mysql> desc department;
```

Field	Type	Null	Key	Default	Extra
departmentID	int(11)	NO	PRI	NULL	auto_increment
name	varchar(100)	NO		NULL	

```
2 rows in set (0.00 sec)
```

```
mysql> desc building;
```

Field	Type	Null	Key	Default	Extra
buildingID	int(11)	NO	PRI	NULL	auto_increment
name	varchar(100)	NO		NULL	
type	varchar(8)	NO		NULL	
addressLine1	varchar(100)	NO		NULL	
addressLine2	varchar(100)	NO		NULL	
addressLine3	varchar(100)	NO		NULL	
city	varchar(50)	NO		NULL	
pin	int(6)	NO		NULL	
latitude	double	NO		NULL	
longitude	double	NO		NULL	

```
10 rows in set (0.00 sec)
```

## CHAPTER 3

```
mysql> desc booking;
```

Field	Type	Null	Key	Default	Extra
bookingID	int(11)	NO	PRI	NULL	auto_increment
bookingDate	date	NO		NULL	
appointmentDate	date	NO		NULL	
personID	int(11)	NO	MUL	NULL	
doctorID	int(11)	NO	MUL	NULL	
slot	time	NO		NULL	
bookingStatus	varchar(9)	NO		NULL	

7 rows in set (0.00 sec)

```
mysql> desc history;
```

Field	Type	Null	Key	Default	Extra
bookingID	int(11)	NO	PRI	NULL	
bookingDate	date	NO		NULL	
appointmentDate	date	NO		NULL	
personID	int(11)	NO	MUL	NULL	
doctorID	int(11)	NO	MUL	NULL	
slot	time	NO		NULL	
bookingStatus	varchar(9)	NO		NULL	

7 rows in set (0.00 sec)



## **CHAPTER 4**

### **HARDWARE AND SOFTWARE REQUIREMENTS**

#### **4.1 SOFTWARE REQUIREMENTS**

Front end:	HTML, CSS, Bootstrap, JavaScript
Back end:	MySQL
Scripting Language:	PHP
Environment:	XamppServer

#### **4.2 HARDWARE REQUIREMENTS**

OS:	Windows/Linux
Memory:	1 GB RAM
Disk:	2 GB HDD Space
Network:	512 Kbps or faster

A web browser with cookies and JavaScript enabled.

### SYSTEM IMPLEMENTATION

- The database software used: MySQL.
- Table used in this project: Doctor, doctorLogin, person, personLogin, booking, history, department, building.
- **Doctor** and **doctorLogin** tables are used store information of doctors.
- **Person** and **personLogin** are used to store information of patients.
- **Booking** and **History** tables are used to store information of the future bookings and past bookings respectively.
- **Department** table stores department name of a particular doctor.
- **building** table stores the name of the hospital/clinic in which a particular doctor is working.

All the records are maintained/edited by writing queries in **MySQL** through **PHP**.

Patient should first register and then login to book a doctor's slot. All booking information will be present in the profile of the patient.

Doctor should login to see all the bookings and cancel it if necessary from their profile page.

## Chapter 6

### CODE SNIPPET

```
1 <?php
2
3 $servername = "localhost";
4 $username = "root";
5 $password = "";
6 $dbname = "BMA";
7
8 //create connection
9 $conn = new mysqli($servername, $username, $password, $dbname);
10
11 //check connection
12 if ($conn->connect_error) {
13     die("Connection failed: " . $conn->connect_error);
14 }
15
```

```
//start transaction
$conn->autocommit(FALSE);

//insert data into doctor table
$sql = "insert into doctor (firstName, lastName, gender, dob, phoneNumber, registerDate, qualification, departmentID, buildingID, experience, fee) values ('$first', '$last', '$gender', '$dob', '$phone', curdate(), '$qualification', '$department', '$building', '$experience', '$fee')";

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

    //echo "Error: " . $conn->error;
    $conn->rollback();
    header("Location: ../doctor/signup.php?signup=error");
    exit();
}

//insert data into doctorLogin table
$sql = "insert into doctorLogin (doctorID, email, password, question, answer, lastLogin) values (last_insert_id(), '$email', '$hashedPassword', '$question', '$hashedAnswer', now())";

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

    //echo "Error: " . $conn->error;
    $conn->rollback();
    header("Location: ../doctor/signup.php?signup=error");
    exit();
}

//commit transaction
$conn->commit();
```

## Chapter 6

### CODE SNIPPET

```
$sql = "select * from doctorLogin where email = '$email'";
$result = $conn->query($sql);

if ($result->num_rows < 1) {

    $_SESSION['doctor_invalid_login'] = 1;

    header("Location: ../doctor/login.php?login=invalid_login");
    exit();
}

if ($row = $result->fetch_assoc()) {

    //hashing the password
    $hashedPassword = md5($password);

    //verify password
    if ($hashedPassword !== $row['password']) {

        $_SESSION['doctor_invalid_login'] = 1;

        header("Location: ../doctor/login.php?login=invalid_login");
        exit();
    }

    $did = $row['doctorID'];
    $sql = "update doctorLogin set lastLogin = now() where doctorID = '$did'";
    $result = $conn->query($sql);

    $_SESSION['doctorID'] = $row['doctorID'];
    $_SESSION['email'] = $row['email'];
    $_SESSION['lastLogin'] = $row['lastLogin'];
    $_SESSION['doctorLogin'] = 1;

    header("Location: ../doctor.php?login=success");
    exit();
}
```

```
//start transaction
$conn->autocommit(FALSE);

//insert data into person table
$sql = "insert into person (firstName, lastName, gender, dob, phoneNumber, registerDate) values ('$first', '$last', '$gender', '$dob', '$phone', curdate())";

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

    //echo "Error: " . $conn->error;
    $conn->rollback();
    header("Location: ../person/signup.php?signup=error");
    exit();
}

//insert data into personLogin table
$sql = "insert into personLogin (personID, email, password, question, answer, lastLogin) values (last_insert_id(), '$email', '$hashedPassword', '$question', '$hashedAnswer', now())";

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

    //echo "Error: " . $conn->error;
    $conn->rollback();
    header("Location: ../person/signup.php?signup=error");
    exit();
}

//commit transaction
$conn->commit();
```

```

$sql = "select * from personLogin where email = '$email'";
$result = $conn->query($sql);

if ($result->num_rows < 1) {

    $_SESSION['invalid_login'] = 1;

    header("Location: ../person/login.php?login=invalid_login");
    exit();
}

if ($row = $result->fetch_assoc()) {

    //hashing the password
    $hashedPassword = md5($password);

    //verify password
    if ($hashedPassword !== $row['password']) {

        $_SESSION['invalid_login'] = 1;

        header("Location: ../person/login.php?login=invalid_login");
        exit();
    }

    $pid = $row['personID'];
    $sql = "update personLogin set lastLogin = now() where personID = '$pid'";
    $result = $conn->query($sql);

    $_SESSION['personID'] = $row['personID'];
    $_SESSION['email'] = $row['email'];
    $_SESSION['lastLogin'] = $row['lastLogin'];
    $_SESSION['login'] = 1;

    header("Location: ../index.php?login=success");
    exit();
}

```

```
k?php
```

```

session_start();

require_once "../includes/connect.php";

if (!isset($_SESSION['doctorID'])) {

    header("Location: ../doctor.php");
    exit();
}

$sql_doctor = $_SESSION['doctorID'];
$sql = "select * from doctor where doctorID = '$sql_doctor'";
$result = $conn->query($sql);
$row = $result->fetch_assoc();
$first = $row['firstName'];
$last = $row['lastName'];
$phone = $row['phoneNumber'];

$sql2 = "select * from booking where doctorID = '$sql_doctor'";
$result2 = $conn->query($sql2);

$sql3="select * from history where doctorID = '$sql_doctor'";
$result3 = $conn->query($sql3);

?>

```

## CODE SNIPPET

```
<?php

session_start();

require_once "../includes/connect.php";

if (!isset($_SESSION['personID'])) {

    header("Location: ../index.php");
    exit();
}

$sql_person = $_SESSION['personID'];
$sql = "select * from person where personID = '$sql_person'";
$result = $conn->query($sql);
$row = $result->fetch_assoc();
$first = $row['firstName'];
$last = $row['lastName'];
$phone = $row['phoneNumber'];

$sql2 = "select * from booking where personID = '$sql_person'";
$result2 = $conn->query($sql2);

$sql3="select * from history where personID = '$sql_person'";
$result3 = $conn->query($sql3);
?>
```

```
<?php

$x = $_GET["searchText"];

$sql1 = "select departmentID, name from department where name like '$x'";
$result1 = $conn->query($sql1);

if($result1->num_rows > 0) {
    while($row = $result1->fetch_assoc()) {
        $departmentID = $row['departmentID'];
        $departmentName = $row['name'];

        $latitude_loc = $_GET['latitude'];
        $longitude_loc = $_GET['longitude'];

        $sql2 = "
        SELECT doctor.*, building.*, department.*,
        sqrt((building.latitude-$latitude_loc)*(building.latitude-$latitude_loc)+(building.longitude-$longitude_loc)*(building.longitude-$
        longitude_loc))*100 AS distance
        FROM doctor, building, department
        WHERE doctor.buildingID = building.buildingID
        AND doctor.departmentID = department.departmentID
        AND department.departmentID = '$departmentID'
        ORDER BY distance;
        ";
        $result2 = mysqli_query($conn,$sql2);

        //$result = mysqli_query("select * from doctor where department = '$x'");

        //$sql = "select * from doctor where department = '$x'";

        //$result = mysqli_query($conn, $sql);

        if ($result2->num_rows > 0) {
            // output data of each row
            while($row = $result2->fetch_assoc()) {
                $firstName = $row["firstName"];
                $lastName = $row["lastName"];
                $gender = $row["gender"];
                $experience = $row["experience"];
                $qualification = $row["qualification"];
                $fee = $row["fee"];
                $doctorID = $row["doctorID"];
                $distance = $row["distance"];
                $distance = floor($distance*10)/10;
            }
        }
    }
}
?>
```



```

//start transaction
$conn->autocommit(FALSE);

$sql = "insert into booking(bookingDate, appointmentDate, personID, doctorID, slot, bookingStatus)
      values(curdate(), '$appointmentdate', '$personID', '$doctorID', '$appointmenttime', 'Booked')";

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

    $conn->rollback();
    header("Location: ../index.php?booking=error");
    exit();
}

//commit transaction
$conn->commit();

//enable autocommit
$conn->autocommit(TRUE);

//start transaction
$conn->autocommit(FALSE);

$_SESSION['appointmentdate'] = $appointmentdate;
$_SESSION['appointmenttime'] = $appointmenttime;

$sql2 = "select bookingID from booking where appointmentDate = '$appointmentdate' and personID =
        '$personID' and doctorID = '$doctorID' and slot = '$appointmenttime'";
$result = $conn->query($sql2);

if ($row = $result->fetch_assoc()) {

    $_SESSION['bookingID'] = $row['bookingID'];
}

//commit transaction
$conn->commit();

```

```

<?php

session_start();
require_once "connect.php";
$bookingID = $_GET['bookingID'];

$sql = "update booking set bookingStatus = 'Cancelled' where bookingID = $bookingID;";
if ($conn->query($sql) !== TRUE) {

    header("Location: ../doctor.php?cancel=error");
    exit();
}

header("Location: ../doctor.php?cancel=success");
exit();
?>

```

# Testing

Testing is the set of activities that can be planned in advance and conducted systematically. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding.

Preparation of the test data plays a vital role in the system testing. After preparation the test data, the system under study is tested those test data. Errors were found and corrected by using the following testing steps and corrections are recorded for future references. Thus, series of testing is performed on the system before it is already for implementation.

There are many strategies that can be used to test conventional software. Testing can be done once the entire software is complete. However, this results in buggy software and is simply not effective.

Alternate approach would be to test the software on a daily basis. Whenever a new part of software is developed, it is sent for testing first. This approach is effective as bugs are eliminated as the software is constructed and leads to more efficient software once all the modules are integrated.

Quality assurance is the review of software products and related documentation for completeness, correctness, reliability and maintainability. And of course it includes assurances that the system meets the specification and the requirements for its intended use and performance. The various levels of quality assurance are described in the following sub sections.



## **CHAPTER 7**

### **7.1 SYSTEM TESTING**

Software testing is a critical element of software quality assurance and represents the ultimate review of specifications, design and coding. The testing phase involves the testing of system using various test data; Preparation of test data plays a vital role in the system testing. After preparation the test data, the system under study is tested.

Those test data, errors were found and corrected by following testing steps and corrections are recorded for future references. Thus a series testing is performed on the system before it is ready for implementation.

**The various types of testing on the system are:**

- Unit testing
- Integrated testing
- Validation testing
- Output testing
- User acceptance testing

### **7.2 Unit testing**

It focuses on the smallest unit of the software design. Smallest unit include the module which are integrated to produce the final project. The unit testing focuses on the internal logic and data structures within the boundaries of the component. Test considerations can be the data structures, boundary conditions, independent paths, error handling paths, etc. Unit testing was done on verifying the email and password for accessing the database. The following results were obtained:

## CHAPTER 7

### Unit testing of each modules:

#### 7.2.1 Test case for Login:

Test Cases	Email-id	Password	Test Result
TC1	Correct Email id	Correct Password	Successful login, Main page is displayed
TC2	Correct Email id	Incorrect Password	Prompt saying “Wrong Password”
TC3	Incorrect Email id	Correct Password	Prompt saying “Email id is not yet registered”
TC4	Incorrect Email id	Incorrect Password	Prompt saying “Email id is not yet registered”

**Fig 1:** Test results for successful and unsuccessful login.

#### 7.2.2 Test Cases for Registration :

Test Cases	E-Mail Id	Test Result
TC1	Unregistered Email Id	Successful Registration, Login page is displayed
TC2	Registered Email Id	Prompt saying that “Email already exists!!!”.

**Fig 2:** Test results for successful and unsuccessful registration.

## **CHAPTER 7**

### **7.3 INTEGRATION TESTING**

Data can be across an interface one module can have an adverse effect on another's sub function, when combined may not produce the desired major function; global data structures can present problems. Integration testing is a symmetric technique for constructing tests to uncover errors associated with the interface. All modules are combined in this testing step. Then the entire program was tested as a whole.

### **7.4 VALIDATION TESTING**

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the consumer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- The function or performance characteristics confirm to specification that are accepted.
- A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

### **7.5 OUTPUT TESTING**

After performing the validation testing, the next step is output testing of the proposed system, since a system is useful if it does not produce the required output in the specific format required by them tests the output generator displayed on the system under consideration. Here the output is considered in two ways: one is onscreen and the other

## **CHAPTER 7**

is printed format. The output format on the screen is found to be correct as the format was designed in the system design phase according to the user needs.

As far as hardcopies are considered it goes in terms with the user requirement. Hence output testing does not result any correction in the system.

### **7.6 BLACK-BOX TESTING**

Black- Box testing refers to tests that are conducted at the software interface. A Black-box test examines some fundamental aspects of a system with little regard for the internal logical structure of the software.

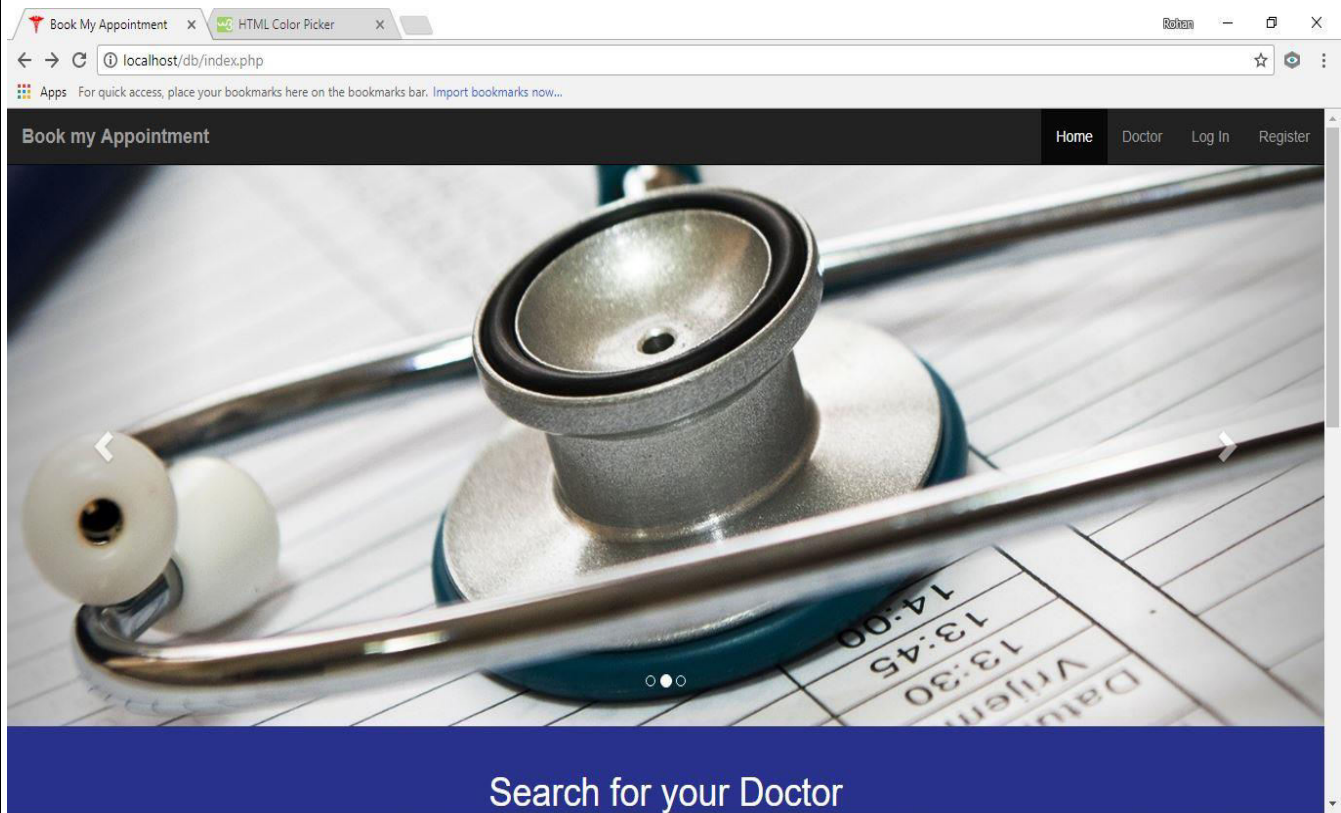
The software developed was subjected to black-box testing to test the functionality of the Graphical User Interface without checking the internal logic of the program. For example, the use of back button resulting in the appearance of right frames etc.

### **7.7 USER ACCEPTANCE TESTING**


User acceptance of the system is a key factor for success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required.

## CHAPTER 8

### SCREEN SHOTS



## Patient's Login



**Email:**

**Password:**

[Not a member? Sign Up](#)

[Forgot Password?](#)

[Back to Home page](#)

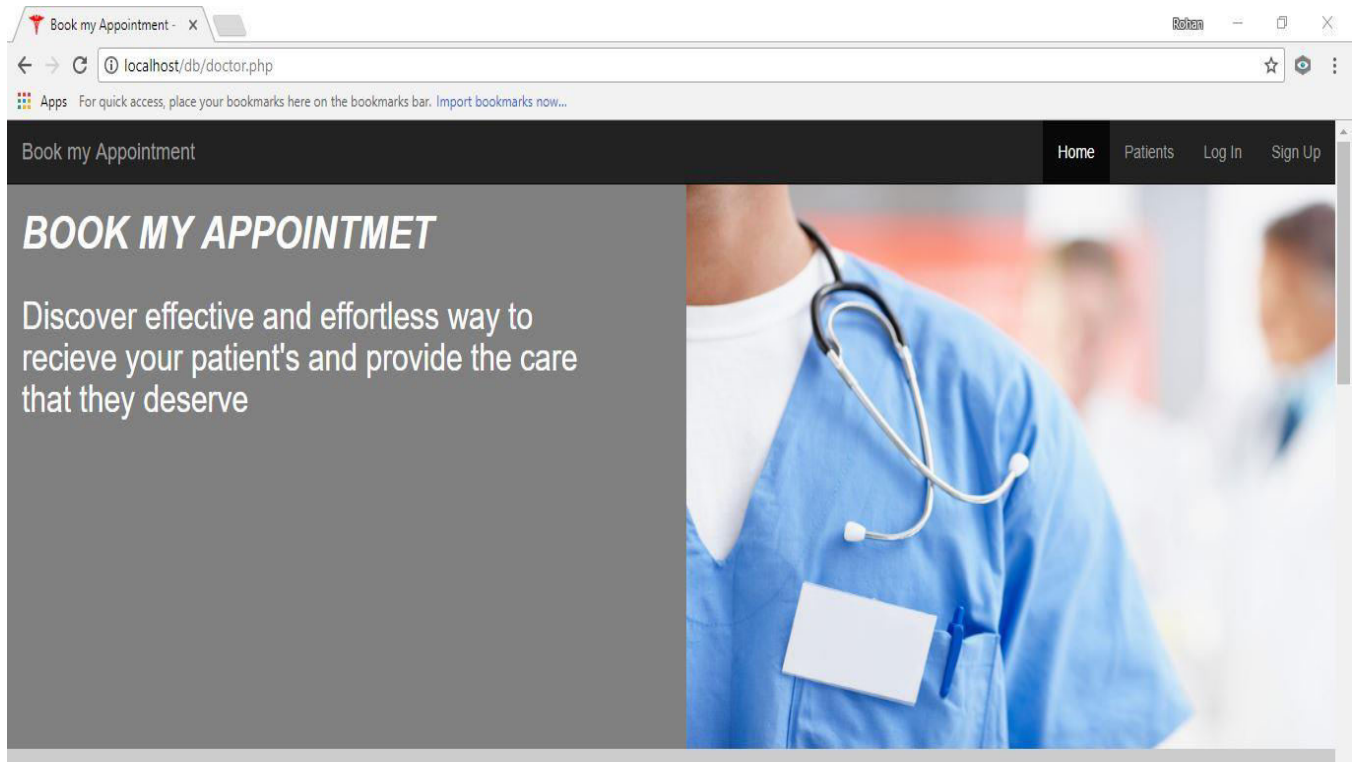
## CHAPTER 8

### SCREEN SHOTS

### Patient register

First Name:	<input type="text" value="Enter First Name"/>
Last Name:	<input type="text" value="Enter Last Name"/>
Gender:	<input type="text" value="Female"/>
Date of Birth:	<input type="text" value="24-11-2017"/>
Phone Number:	<input type="text" value="Enter Phone Number"/>
Email:	<input type="text" value="Enter Email"/>
Password:	<input type="text" value="Create a Password"/>
Security Question:	<input type="text" value="In what city or town was your mother born?"/>
Security Answer:	<input type="text" value="Enter Security Answer"/>

[Already a member? Log In](#)  
[Back to Home](#)




## CHAPTER 8

### SCREEN SHOTS



### Doctor's Login



**Email:**

**Password:**

[Not a member? Register](#)

[Forgot Password?](#)

[Back to Home](#)

### Doctor's Registration

**First Name:**

**Last Name:**

**Gender:**

**Date of Birth:**

**Phone Number:**

**Qualifications:**

**Department:**

**Hospital / Clinic:**

**Experience:**

**Fee:**

**Email:**

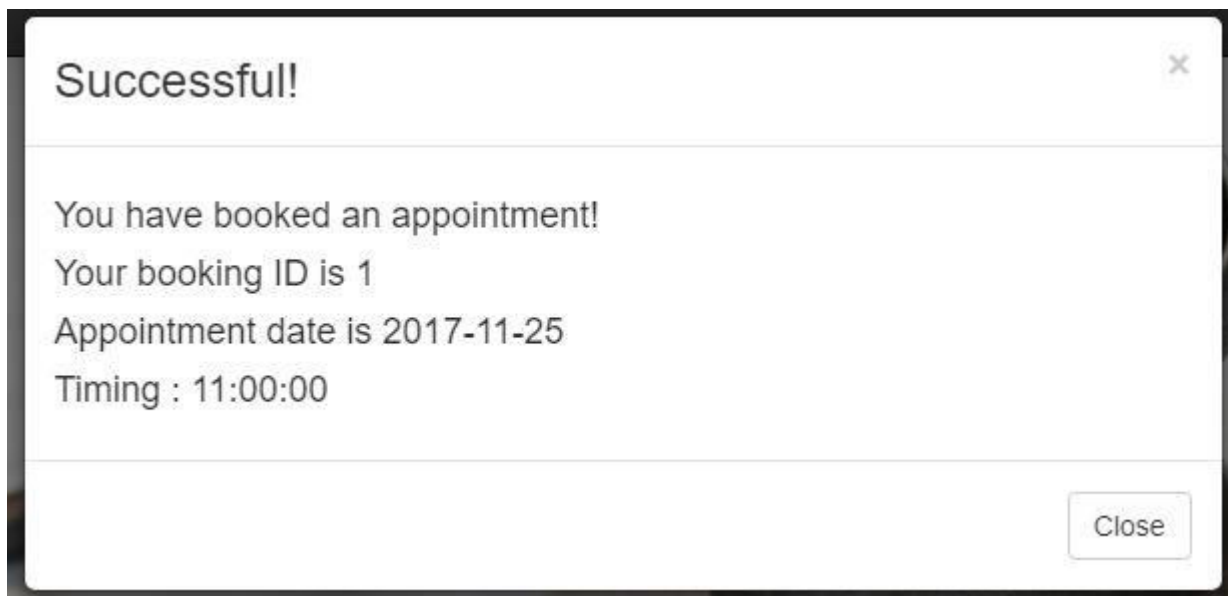
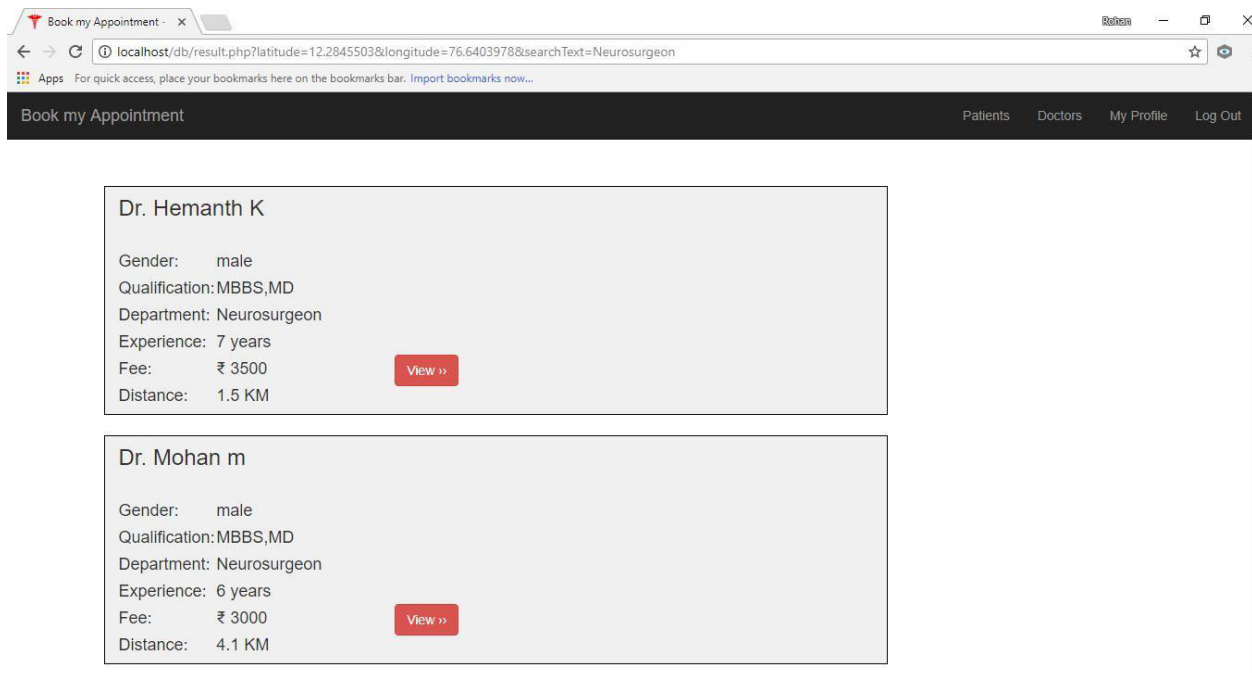
**Password:**

**Security Question:**

**Confirm Password:**

## CHAPTER 8

### SCREEN SHOTS





## CHAPTER 8

### SCREEN SHOTS

The screenshot shows a web browser window with the address bar displaying `localhost/db/person/profile.php`. The page title is "Book My Appointment". The user is logged in as "Rohan". The page content includes a "Your Profile" section with a blue header. It displays a welcome message "Welcome Rohan K", the last login time "2017-11-24 11:49:21", and contact information: Email: `sr.rohanro@gmail.com` and Phone Number: `9483478805`. Below this is a section titled "Your Upcoming Bookings" with a table showing one booking. At the bottom, it states "No Past Bookings!".

Booking ID	Booking Date	Appointment Date	Time	Doctor Name	Status
1	2017-11-24	2017-11-25	11:00:00	Dr. Mohan m	Booked

The screenshot shows a web browser window with the address bar displaying `localhost/db/doctor/profile.php`. The page title is "Book my Appointment". The user is logged in as "Rohan". The page content includes a "Your Profile" section with a grey header. It displays a welcome message "Welcome Dr. Mohan m", the last login time "2017-11-24 11:31:12", and contact information: Email: `moh@g.com` and Phone Number: `4563982178`. Below this is a section titled "Your Upcoming Appointments" with a table showing one appointment. At the bottom, it states "No Past Appointments!".

Booking ID	Appointment Date	Time	Patient Name	Status	Manage
1	2017-11-25	11:00:00	Rohan K	Booked	<a href="#">Cancel</a>

### CONCLUSION AND FUTURE ENHANCEMENTS

#### CONCLUSION:

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also been adopted.

This application provides the security of data and ensures data's accuracy.

This application will minimize the manual data entry and will reduce the unwanted time taken to manually book appointment to a specific doctor.

Doctor can check their profile to see the timings he has to treat a patient.

Moreover, he/she can also cancel the appointment if he/she is too busy to treat at the specific time.

Due to the encryption by PHP, the application cannot make use of user's private data.

The data in the database is backed up frequently to recover the data during system failure using transactions.

This application can provide better service and can improve efficiency of the doctor appointment system.

#### FUTURE ENHANCEMENTS:

This application provides only the basic prototype of the project which is applicable for a single constituency. We can develop fully functional web portal using MySQL and other required applications to provide efficient way of booking doctor's appointment.

This application can be further extended by adding more features that will make it more portable and convenient for user.

## CHAPTER 9

Some enhancements that can be introduced in the project are:

- Review system can be added where each patient can rate their experience with the doctor.
- We can host the platform on online servers to make it accessible worldwide.
- Create the master and slave databases structure to reduce the overload of the database queries.
- Implement the backup mechanism for taking backup of codebase and database on regular basis on different servers.
- We intend to add emergency button such that in emergency case, an ambulance can be deployed instantaneously.

## CHAPTER 10

### REFERENCES

#### WEBSITES:

<https://www.udemy.com/>

<https://stackoverflow.com/>

<http://www.php.net/>

<https://www.apachefriends.org/>

<https://www.tutorialspoint.com/>

<https://www.mysql.com/>

#### TEXTBOOKS:

***“DATABASE SYSTEMS” – Ramez Elmasri, Shamkant B. Navathe.***

***“Database System Concepts” - Henry Korth, S.Sudarshan and Abraham Silberschatz***

***“THE DATABASE BOOK – PRINCIPLE AND PRACTICE USING  
MYSQL” – Narain Gehani.***