

## CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION TO SQL

SQL which is an abbreviation for **Structured Query Language** is a language to request data from a database, to add, update, or remove data within a database, or to manipulate the metadata of the database.

Sometimes SQL is characterized as *non-procedural* because procedural languages generally require the details of the operations to be specified, such as opening and closing tables, loading and searching indexes, or flushing buffers and writing data to file systems. Therefore, SQL is designed at a higher conceptual level of operation than procedural languages.

Commonly used statements are grouped into the following categories

**Data Query Language (DQL):** The commands are used to retrieve data from the database

SELECT - Used to retrieve certain records from one or more tables.

**Data Manipulation Language (DML):** The commands are used to insert data, modify data and delete data from the database.

INSERT - Used to create a record

UPDATE - Used to change certain records.

DELETE - Used to delete certain records.

**Data Definition Language (DDL):** The commands are used to create database objects, alter the structure and delete database objects

CREATE - Used to create a new table, a view of a table, or other object in database.

ALTER - Used to modify an existing database object, such as a table.

DROP - Used to delete an entire table, a view of a table or other object in the database.

**Data Control Language (DCL):** The commands are used to control the access to data store in the database.

GRANT - Used to give a privilege to someone

REVOKE - Used to take back privileges granted to someone.

## 1.2 INTRODUCTION TO FRONT END SOFTWARE

The “front end languages” live in the browser. After you type in an address in the address bar at the top and hit Enter, your browser will receive at least an HTML file from the web server.

Each of these languages performs a separate but very important function but they work harmoniously together to determine how the web page is **STRUCTURED(HTML)**, how it **LOOKS(CSS)**, and how its **FUNCTIONS (PHP)**.

Front end web development is NOT design (You won’t be playing around in Photoshop or anything), but a front-end developer does apply the work of designers to the web page by translating their well-designed layouts into real code. The front-end developer stands between the designer on one end and the back-end developer on the other, translating the design into code and plugging the data from the back-end developer into the right spots.

**PHP** is a server-side scripting language designed primarily for web development but also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Development Team.

PHP code may be embedded into HTML or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server software combines the result of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and it can be used to implement stand-alone graphical applications.

The standard PHP interpreter, powered by the Zend Engine, is free to use software released under the PHP License. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as Common Gateway Interface(CGI) executable. PHP has been widely ported on web servers on almost every operating system and platform, free of charge.

## 1.3 PROJECT REPORT OUTLINE

The report is arranged in the following way:

**Chapter 1: Introduction**

In this chapter, an introduction to SQL and an introduction to front end is given

**Chapter 2: Requirement Specification**

In this chapter the software requirements and hardware requirements required for the project is specified.

**Chapter 3: Objective of the Project**

The objective of the project gives us details about the project that is being done.

**Chapter 4: Implementation**

The required ER diagram and the mapping of the ER diagram to the schema is done

**Chapter 5: FRONT END DESIGN**

This chapter deals with the front end code and php code required for the project.

**Chapter 6: Testing**

This chapter gives the outline of all the testing methods that are carried out to get a bug free system.

**Chapter 7: Results**

Displays the results obtained

## CHAPTER 2

### REQUIREMENTS SPECIFICATION

#### 2.1 SOFTWARE REQUIREMENTS

Operating System	: 64bit WINDOWS Operating System, X64-based processor
Database	: MYSQL
Scripting Language	: HTML5, CSS3, PHP
Server	: WAMP

#### 2.2 HARDWARE REQUIREMENTS

Processor	: Intel Celeron CPU N3060 @1.60GHz or Above
RAM	: 4.00 GB or Above
Hard Disk	: 1 TB
Compact Disk	: CD-ROM, CD-R, CD-RW
Input device	: Keyboard

## CHAPTER 3

### OBJECTIVE OF THE PROJECT

**The main objective of Physiotherapy database management system is:**

- As modernizing is taking over all the systems and digitalizing helps them improve in so many ways. This Physiotherapy database management system will help the administration in speeding up the tasks and reduce the complexity.
- The objective of this system is to digitalize and create a system where a user is able to book an online appointment for physiotherapy .
- This system will enable an user to search the physiotherapist and gives the feedback about the website.
- This system will enable an user to delete their appointment and eliminate their details from the database .
- This system will also enable the user to share their contact details in order to get the more information about the physiotherapy management system.
- This system also contains the information about physiotherapy and about the physiotherapists. This system also helps the users in reducing carbon footprint as amount of paper used in company reduces.
- This will improve the transparency between the users and the system which is a good quality in a system. It will also give the layer of security to the users that only authorize users can access by their credentials.
- One of the most important objective of this project is that we learnt different web scripting languages like HTML, CSS, PHP and also gained knowledge about SQL.

## CHAPTER 4

# IMPLEMENTATION

### 4.1 ER DIAGRAM OF PHYSIOTHERAPY DATABASE

1. An **entity-relationship model (ER Model)** describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.
2. An entity may be defined as a thing capable of an independent existence that can be uniquely identified. An entity is an abstraction from the complexities of a domain.
3. Attributes are drawn as ovals and are connected with a line to exactly one entity or relationship set.
4. An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.
5. Cardinality constraints are expressed as follows:
  - a. A double line indicates a participation constraint, totality or subjectivity: all entities in the entity set must participate in at least one relationship in the relationship set.
  - b. An arrow from entity set to relationship set indicates a key constraint, i.e. injectivity: each entity of the entity set can participate in at most one relationship in the relationship set.
  - c. A thick line indicates both, i.e. bijectivity: each entity in the entity set is involved in exactly one relationship.
  - d. An underlined name of an attribute indicates that it is a key: two different entities or relationships with this attribute always have different values for this attribute.

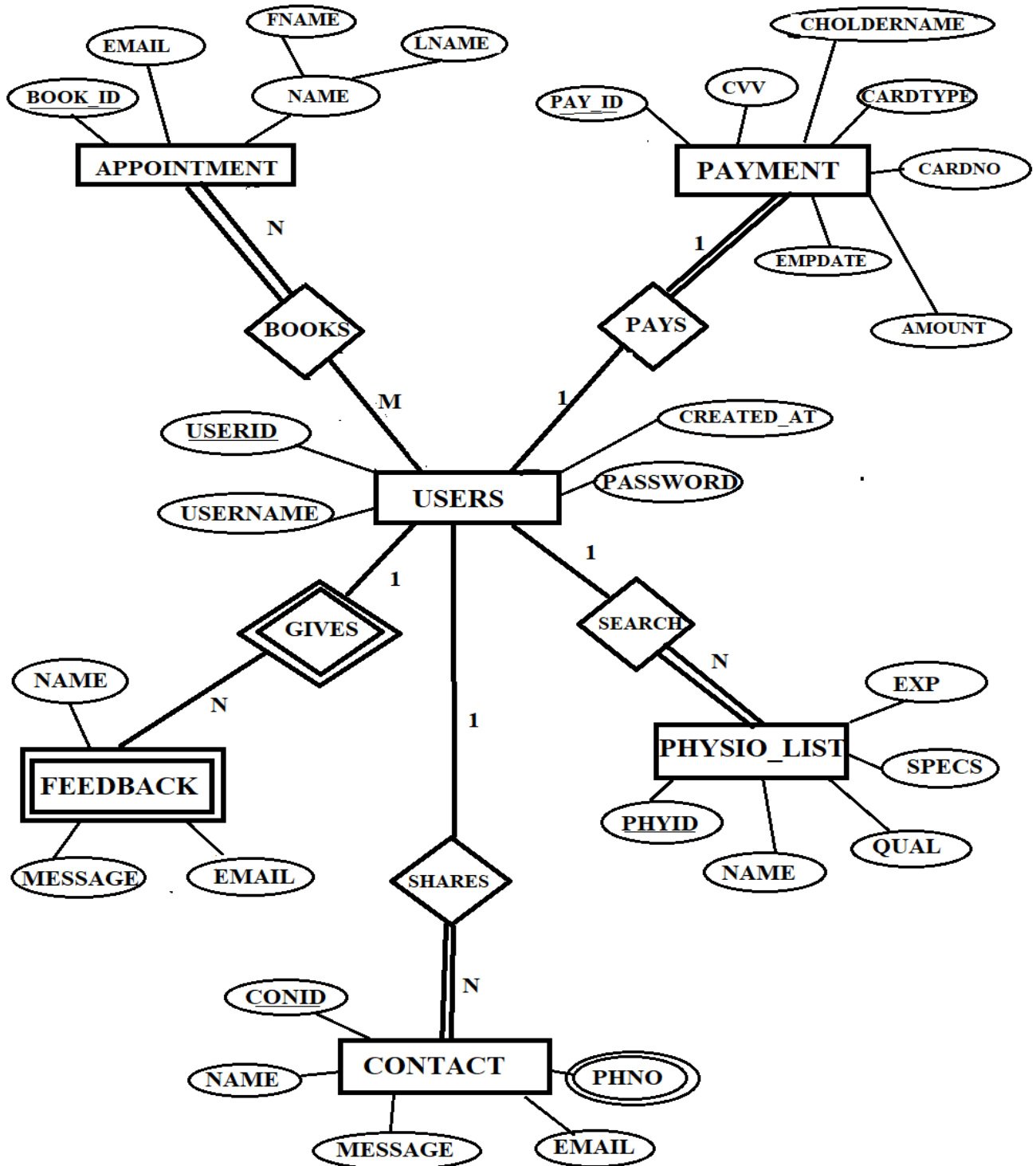


Figure 4.1: ER Diagram of Physiotherapy Database

## 4.2 MAPPING OF ER DIAGRAM TO RELATIONS

### STEP 1: Mapping of Regular Entities

For each regular entity type E in the ER schema, create relation R that includes all simple attributes of E.

#### USERS

<u>USERID</u>	USERNAME	PASSWORD	CREATED_AT
---------------	----------	----------	------------

#### APPOINTMENT

<u>BOOKID</u>	FIRSTNAME	LASTNAME	EMAIL
---------------	-----------	----------	-------

#### PAYMENT

<u>PAYID</u>	CARDTYPE	CHOLDERNAME	CARDNO	EXPDATE	CVV	AMOUNT
--------------	----------	-------------	--------	---------	-----	--------

#### CONTACT

<u>CONID</u>	NAME	MESSAGE	EMAIL	PHNO
--------------	------	---------	-------	------

### STEP 2 : Mapping of Weak Entity Types

- For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes.
- The primary key of R is the combination of the primary key(s) of the owner and the partial key of the weak entity

#### USERS

<u>USERID</u>	USERNAME	PASSWORD	CREATED_AT
---------------	----------	----------	------------

#### FEEDBACK

<u>USERID</u>	name	Email	message
---------------	------	-------	---------

FK

### STEP 3: Mapping of 1:1 Relationship

- Identify the relation S that represents the participating entity type at the 1-side of the relationship type.



- Include as foreign key in S the primary key of the relations T that represents the other entity type participating in R.
- For each binary 1:1 relationship type R in ER schema, identify the relations S and T that correspond to the entity types participating in R if any.

#### USERS

<u>USERID</u>	USERNAME	PASSWORD	CREATED_AT
---------------	----------	----------	------------

#### PAYMENT

<u>PAYID</u>	CARDTYPE	CHOLDERNAME	CARDNO	EXPDATE	CVV	AMOUNT
--------------	----------	-------------	--------	---------	-----	--------

FK

### STEP 4 : Mapping of 1:N Relationship

- For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
- Include any simple attributes of the 1:N relation type as attributes of S

#### a) USERS AND CONTACT

##### USERS

<u>USERID</u>	USERNAME	PASSWORD	CREATED_AT
---------------	----------	----------	------------

##### CONTACT

<u>CONID</u>	NAME	MESSAGE	EMAIL	PHNO
--------------	------	---------	-------	------

FK

#### b)USERS AND FEEDBACK

##### USERS

<u>USERID</u>	USERNAME	PASSWORD	CREATED_AT
---------------	----------	----------	------------

##### FEEDBACK

<u>USERID</u>	NAME	EMAIL	MESSAGE
---------------	------	-------	---------

FK

### STEP 5 : Mapping of M:N Relationship

Create a new relation S to represent R. Include as foreign key attributes in S the primary key of the relations that represents the participating entity types their combination will form the primary key of S. Also, include any simple attributes of the M:N relationship type as attributes of S.

**USERS**

<u>USERID</u>	USERNAME	PASSWORD	CREATED_AT
---------------	----------	----------	------------

**BOOKS**

<u>USERID</u>	<u>BOOKID</u>
FK	FK

**APPOINTMENT**

<u>BOOKID</u>	FIRSTNAME	LASTNAME	EMAIL
---------------	-----------	----------	-------

**STEP 6: Mapping of Multi-Valued Attributes**

For each multivalued attributes A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.

The Primary Key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

**CONTACT**

<u>CONID</u>	NAME	EMAIL	PHNO	MESSAGE
--------------	------	-------	------	---------

**DETAIL**

<u>CONID</u>	PHNO
FK	

**STEP 7: Mapping of N-ARY Relationship Types**

For each n-ary relationship type R, where  $n > 2$  create a new relationship S to represent R.  $\lambda$  include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.

$\lambda$  also includes any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S. There are **no** n-ary relationship types.

### 4.3 SCHEMA DIAGRAM OF PHYSIOTHERAPY MANAGEMENT

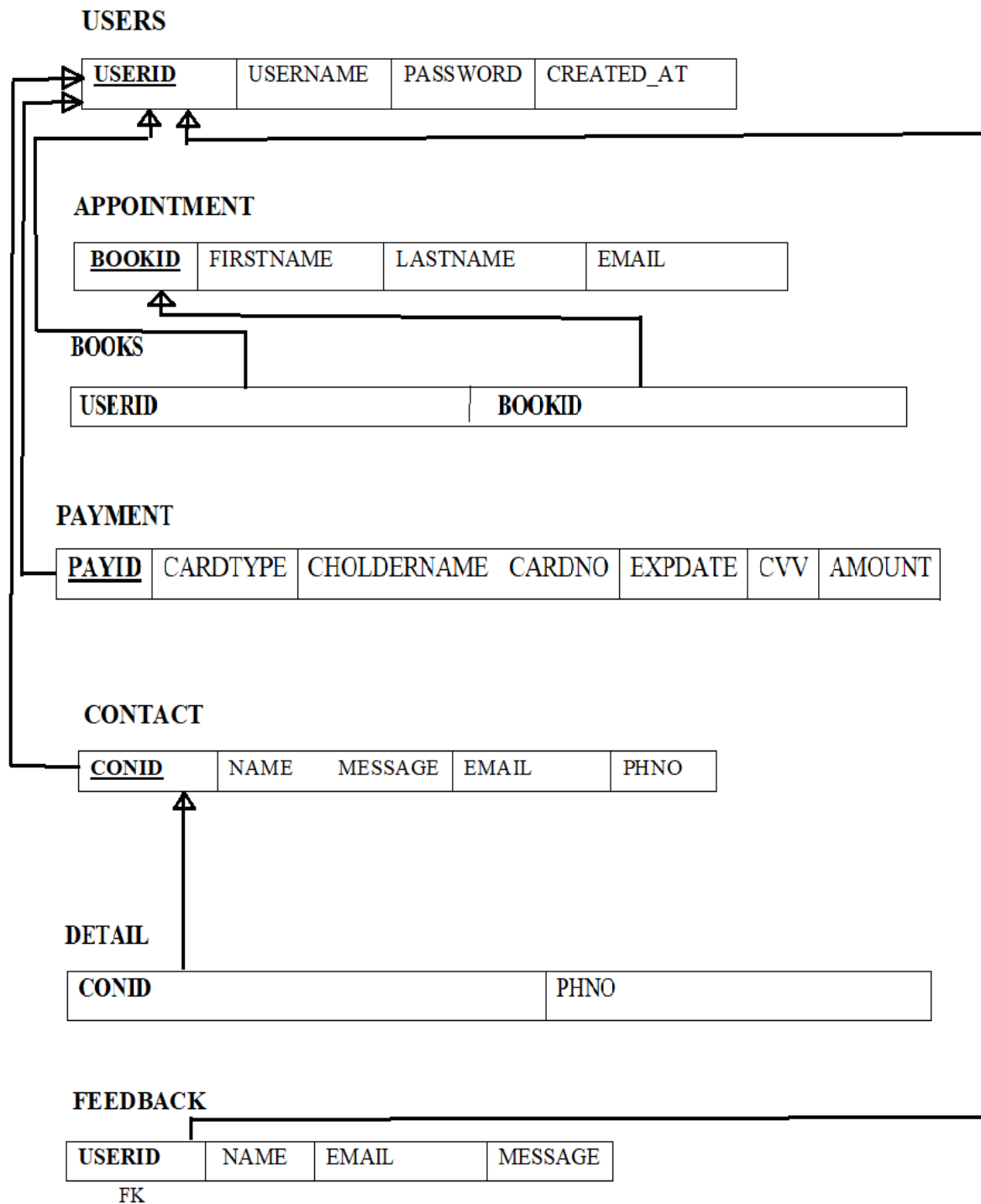


Figure 4.2: Schema Diagram

## 4.3 NORMALIZE THE RELATIONS

Database normalization, or simply normalization, is the process of organizing the columns(attributes) and tables(relations) of a relational database to reduce data redundancy and improve data integrity. Normalization involves arranging attributes in relations based on dependencies between attributes.

### 1. First Normal Form

As per First normal form, no two rows of data must contain repeating group of information. Each set of columns must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that will distinguish it as unique.

**Example:**

**USERS**

<u>USERID</u>	USERNAME	PASSWORD	CREATED_AT
---------------	----------	----------	------------

All the tables in the database are normalized to 1NF as all the attributes are atomic.


### 2. Second Normal Form (2NF)

A table is in 2NF if it is in 1NF and if all non-key attributes are fully functionally dependent on all of the key.

**Example:**

**APPOINTMENT**

<u>BOOKID</u>	FIRSTNAME	LASTNAME	EMAIL
---------------	-----------	----------	-------



### 3. Third Normal Form(3NF)

A table is in 3NF if it is in 2NF and if it has no transitive dependency.  $X \rightarrow Y$ ,  $Y \rightarrow Z$ ,  $X \rightarrow Z$

According to CODD's definition a relation schema R is in 3NF. It satisfies 2NF and no non-prime attribute of R is transitively dependent on the primary key. All tables of database satisfies upto 3NF.

## 4.5 CREATION OF TABLES

### 1. CREATING USERS TABLE

```
CREATE TABLE `physio`.`users` ( `userid` INT(11) NOT NULL AUTO_INCREMENT ,
`username` VARCHAR(11) NOT NULL, `password` VARCHAR(50) NOT NULL
,`created-at` DATETIME NOT NULL, PRIMARY KEY (`userid`), UNIQUE (`userid`));
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id</b>	int(11)			No	None		AUTO_INCREMENT	<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
2	<b>username</b>	varchar(50)	latin1_swedish_ci		No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
3	<b>password</b>	varchar(255)	latin1_swedish_ci		No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
4	<b>created_at</b>	datetime			Yes	CURRENT_TIMESTAMP			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>

### 2. CREATE APPOINTMENT TABLE

```
CREATE TABLE `physio`.`bookapt` ( `bookid` INT(11) NOT NULL
AUTO_INCREMENT , `firstname` VARCHAR(11) NOT NULL, `lastname`
VARCHAR(50) NOT NULL ,`dob` NOT NULL, `email` VARCHAR(50) NOT NULL,
PRIMARY KEY (`bookid`), UNIQUE (`bookid`));
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>bookid</b>	int(100)			No	None		AUTO_INCREMENT	<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
2	<b>firstname</b>	varchar(100)	latin1_swedish_ci		No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
3	<b>lastname</b>	varchar(100)	latin1_swedish_ci		No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
4	<b>dob</b>	date			No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
5	<b>email</b>	varchar(100)	latin1_swedish_ci		No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>

### 3. CREATE PAYMENT TABLE

```
CREATE TABLE `physio`.`payment` ( `payid` INT(11) NOT NULL AUTO_INCREMENT
, `cardtype` VARCHAR(11) NOT NULL, `choldername` VARCHAR(50) NOT NULL
,`cardno` INT(11) NOT NULL ,`expirydate` DATE,`cvv` INT(11),`amount` INT(11) ,
PRIMARY KEY (`payid`), UNIQUE (`payid`));
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>payid</b>	int(100)			No	None		AUTO_INCREMENT	<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
2	<b>cardtype</b>	varchar(100)	latin1_swedish_ci		No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
3	<b>choldername</b>	varchar(100)	latin1_swedish_ci		No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
4	<b>cardno</b>	int(100)			No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
5	<b>expirydate</b>	date			No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
6	<b>cvv</b>	int(20)			No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
7	<b>amount</b>	int(100)			No	None			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>

#### 4. CREATE FEEDBACK TABLE

```
CREATE TABLE `physio`.`feedback` ( `NAME` VARCHAR(50) NOT NULL ,`EMAIL` VARCHAR NOT NULL, UNIQUE (`userid`));
```

```
ALTER TABLE `bookapt` ADD FOREIGN KEY (`userid`) REFERENCES `users`(`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	<b>userid</b>	int(11)		No	None			Change  Drop  More
<input type="checkbox"/>	2	<b>name</b>	varchar(100) latin1_swedish_ci		No	None			Change  Drop  More
<input type="checkbox"/>	3	<b>email</b>	varchar(100) latin1_swedish_ci		No	None			Change  Drop  More
<input type="checkbox"/>	4	<b>message</b>	varchar(100) latin1_swedish_ci		No	None			Change  Drop  More

#### 5. CREATE CONTACT TABLE

```
CREATE TABLE `physio`.`ctact` ( `conid` INT(11) NOT NULL AUTO_INCREMENT , `name` VARCHAR(11) NOT NULL, `email` VARCHAR(50) NOT NULL ,`phno` INT(11) NOT NULL, PRIMARY KEY (`conid`), UNIQUE (`conid`));
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	<b>conid</b>	int(100)		No	None		AUTO_INCREMENT	Change  Drop  More
<input type="checkbox"/>	2	<b>name</b>	varchar(100) latin1_swedish_ci		No	None			Change  Drop  More
<input type="checkbox"/>	3	<b>email</b>	varchar(100) latin1_swedish_ci		No	None			Change  Drop  More
<input type="checkbox"/>	4	<b>phno</b>	int(100)		No	None			Change  Drop  More
<input type="checkbox"/>	5	<b>message</b>	varchar(100) latin1_swedish_ci		No	None			Change  Drop  More

#### 6. CREATE PHYSIOTHERAPIST TABLE

```
CREATE TABLE `physio`.`physiotherapist` ( `physioid` INT(11) NOT NULL AUTO_INCREMENT , `fullname` VARCHAR(11) NOT NULL, `qual` VARCHAR(50) NOT NULL ,`specs` VARCHAR(11) NOT NULL,`exp` INT(11) NOT NULL, PRIMARY KEY (`physioid`), UNIQUE (`physioid`));
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	<b>physioid</b>	int(11)		No	None			Change  Drop  More
<input type="checkbox"/>	2	<b>fullname</b>	varchar(50) latin1_swedish_ci		No	None			Change  Drop  More
<input type="checkbox"/>	3	<b>qual</b>	varchar(100) latin1_swedish_ci		No	None			Change  Drop  More
<input type="checkbox"/>	4	<b>specs</b>	varchar(100) latin1_swedish_ci		No	None			Change  Drop  More
<input type="checkbox"/>	5	<b>exp</b>	int(20)		No	None			Change  Drop  More

## 4.6 INSERTION OF TUPLES

### 1. INSERTION OF USERS TABLE
















```
INSERT INTO `users` (`id`, `username`, `password`, `created-at`) VALUES ('12', 'Shaashwata Audichya', '1234567', '2019-11-11 00:57:43');
```

```
INSERT INTO `users` (`id`, `username`, `password`, `created-at`) VALUES ('13', 'Cristiano Ronaldo', '777777', '2019-11-11 00:58:45');
```

```
INSERT INTO `users` (`id`, `username`, `password`, `created-at`) VALUES ('14', 'Rahul Dravid', '191919', '2019-11-11 18:05:59');
```

```
INSERT INTO `users` (`id`, `username`, `password`, `created-at`) VALUES ('15', 'Neha Ramesh', 'topper', '2019-11-11 18:11:32');
```

```
INSERT INTO `users` (`id`, `username`, `password`, `created-at`) VALUES ('16', 'Shilpashree Gr', 'abababa', '2019-11-11 18:13:01');
```

	id	username	password	created_at
  	12	Shaashwata Audichya	\$2y\$10\$308rDT0NqhQ8qPz658bJiO0gcs6xG3sEezEdrean/oU...	2019-11-11 00:57:43
  	13	Cristiano Ronaldo	\$2y\$10\$6tcQkLkjHqf4mDyXna9.5ewLTuwjHyyGfbu/NE20cJd...	2019-11-11 00:58:45
  	14	Rahul Dravid	\$2y\$10\$9guH1FgN0SfJ9vooM.QrSOBBXCXfui5r8T4hzFH5srE...	2019-11-11 18:05:59
  	15	Neha Ramesh	\$2y\$10\$/Q9A6jqtxSLirxW8ffH8PqurlQBebrZ8EI12HII0StBE...	2019-11-11 18:11:32
  	16	Shilpashree Gr	\$2y\$10\$MFPBTtqt4MUW0oliYzBtLuYFUWKY9N/oEhBv9/eBaV...	2019-11-11 18:13:01

### 2. INSERTION OF APPOINTMENT TABLE

```
INSERT INTO `bookapt` (`bookid`, `firstname`, `lastname`, `dob`, `email`) VALUES ('7', 'CRISTIANO', 'RONALDO', '1975-02-04', 'criscr7@gmail.com');
```

```
INSERT INTO `bookapt` (`bookid`, `firstname`, `lastname`, `dob`, `email`) VALUES ('11', 'PRANAV', 'RAO', '1998-10-10', 'prao@gmail.com');
```

```
INSERT INTO `bookapt` (`bookid`, `firstname`, `lastname`, `dob`, `email`) VALUES ('12', 'PRAJWAL', 'YB', '1997-05-13', 'praj@yahoo.com');
```

```
INSERT INTO `bookapt` (`bookid`, `firstname`, `lastname`, `dob`, `email`) VALUES ('13', 'PRANAV', 'RAO', '1999-04-01', 'neha@gmail.com');
```

```
INSERT INTO `bookapt` (`bookid`, `firstname`, `lastname`, `dob`, `email`) VALUES ('14', 'SHILPASHREE', 'GR', '1999-01-20', 'ssdababa@gmail.com');
```

		bookid	firstname	lastname	dob	email
<input type="checkbox"/>	Edit  Copy  Delete	7	CRISTIANO	RONALDO	1975-02-04	criscr7@gmail.com
<input type="checkbox"/>	Edit  Copy  Delete	11	PRANAV	RAO	1998-10-10	prao@gmail.com
<input type="checkbox"/>	Edit  Copy  Delete	12	PRAJWAL	YB	1997-05-13	praj@yahoo.com
<input type="checkbox"/>	Edit  Copy  Delete	13	NEHA	RAMESH	1999-04-01	neha@gmail.com
<input type="checkbox"/>	Edit  Copy  Delete	14	SHILPASHREE	GR	1999-01-20	ssdababa@gmail.com

### 3. INSERTION OF PAYMENT TABLE

INSERT INTO `payment` (`payid`, `cardtype`, `choldername`, `cardno`,  
`expirydate`, `cvv`, `amount`) VALUES ('25', 'rupay', 'Neha', '41257896', '2019-11-29', '896',  
'100');

INSERT INTO `payment` (`payid`, `cardtype`, `choldername`, `cardno`,  
`expirydate`, `cvv`, `amount`) VALUES ('26', 'visa', 'Mahi', '18318338', '2020-03-29', '896',  
'100');

INSERT INTO `payment` (`payid`, `cardtype`, `choldername`, `cardno`,  
`expirydate`, `cvv`, `amount`) VALUES ('27', 'mastercard', 'Suresh', '74125896', '2020-02-  
21', '116', '100');

INSERT INTO `payment` (`payid`, `cardtype`, `choldername`, `cardno`,  
`expirydate`, `cvv`, `amount`) VALUES ('28', 'visa', 'Harry', '25896314', '2020-05-23', '333',  
'100');

INSERT INTO `payment` (`payid`, `cardtype`, `choldername`, `cardno`,  
`expirydate`, `cvv`, `amount`) VALUES ('29', 'rupay', 'Vijay', '22211133', '2025-11-29',  
'232', '100');

		payid	cardtype	choldername	cardno	expirydate	cvv	amount
<input type="checkbox"/>	Edit  Copy  Delete	25	rupay	Neha	41257896	2019-11-29	896	100
<input type="checkbox"/>	Edit  Copy  Delete	26	visa	Mahi	18318338	2020-03-29	201	100
<input type="checkbox"/>	Edit  Copy  Delete	27	mastercard	Suresh	74125896	2020-02-21	116	100
<input type="checkbox"/>	Edit  Copy  Delete	28	visa	Harry	25896314	2020-05-23	333	100
<input type="checkbox"/>	Edit  Copy  Delete	29	rupay	Vijay	22211133	2025-11-29	232	100

### 4. INSERTION OF FEEDBACK TABLE

INSERT INTO `feedback` (`userid`, `name`, `email`, `message`) VALUES ('1', 'mastercard',  
'amateuraudichya1172@gmail.com', 'Hala Madrid');

INSERT INTO `feedback` (`userid`, `name`, `email`, `message`) VALUES ('2', 'Rodriquez',  
'jrodriquez@gmail.com', 'Vamos !! Hala Madrid');

INSERT INTO `feedback` (`userid`, `name`, `email`, `message`) VALUES ('4', 'Prajwal',



```
'praj@yahoo.com', 'good');
```

```
INSERT INTO `feedback` (`userid`, `name`, `email`, `message`) VALUES ('5', 'Neha Ramesh', 'neha@gmail.com', 'Awesome');
```

userid	name	email	message
1	mastercard	amateuraudichya1172@gmail.com	Hala Madrid
2	Rodriquez	jrodriquez@gmail.com	Vamos !! Hala Madrid
3	Shaashwata Audichya	audi1172@gmail.com	kya yaar !
4	prajwal	praj@yahoo.com	Good
5	Neha Ramesh	neha@gmail.com	Awesome

## 5. INSERTION OF CONTACT TABLE

```
INSERT INTO `ctact` (`conid`, `name`, `email`, `phno`, `message`) VALUES ('12', 'ashutosh', 'abc@gmail.com', '1122334455', 'Nice');
```

```
INSERT INTO `ctact` (`conid`, `name`, `email`, `phno`, `message`) VALUES ('13', 'Yuvraj', 'yuvi@gmail.com', '1122334458', 'nice');
```

```
INSERT INTO `ctact` (`conid`, `name`, `email`, `phno`, `message`) VALUES ('14', 'Krunal', 'krp@gmail.com', '1122336655', 'very nice');
```

```
INSERT INTO `ctact` (`conid`, `name`, `email`, `phno`, `message`) VALUES ('15', 'Shreyas', 'siyer@gmail.com', '1144778855', 'tremendous');
```

		conid	name	email	phno	message
<input type="checkbox"/>	Edit  Copy  Delete	12	ashutosh	abc@gmail.com	1122334455	Nice
<input type="checkbox"/>	Edit  Copy  Delete	13	Yuvraj	yuvi@gmail.com	1122334458	nice
<input type="checkbox"/>	Edit  Copy  Delete	14	Krunal	krp@yahoo.com	1122336655	very nice
<input type="checkbox"/>	Edit  Copy  Delete	15	Shreyas	siyer@gmail.com	1144778855	tremendous
<input type="checkbox"/>	Edit  Copy  Delete	16	Rajesh	rajesh@gmail.com	1122336655	tremendous

## 6. INSERTION OF PHYSIOTHERAPIST TABLE

```
INSERT INTO `physiotherapist` (`physioid`, `fullname`, `qual`, `specs`, `exp`) VALUES ('2', 'Dr.Prajwal YB', 'B.PTh, CCCE', 'Cardiopulmonary', '15');
```

```
INSERT INTO `physiotherapist` (`physioid`, `fullname`, `qual`, `specs`, `exp`) VALUES ('3', 'Dr.Neha R', 'BPT,MPT(Neuro)', 'Neurology', '26');
```

```
INSERT INTO `physiotherapist` (`physioid`, `fullname`, `qual`, `specs`, `exp`) VALUES ('1', 'Dr.Shilpa Gr', 'Sports Medicine (ABPMR)', 'Sports physiotherapy', '19');
```

<div><div><div><div></div><div></div><div></div></div></div><div></div></div>			physioid	fullname	qual	specs	exp	
<div><div><div></div></div></div>	<div><div><div></div></div><div>Edit</div></div>	<div><div><div></div></div><div>Copy</div></div>	<div><div><div></div></div><div>Delete</div></div>	1	Dr.Pranav Rao	Bachelor of Physiotherapy	Cardiorespiratory	20
<div><div><div></div></div></div>	<div><div><div></div></div><div>Edit</div></div>	<div><div><div></div></div><div>Copy</div></div>	<div><div><div></div></div><div>Delete</div></div>	2	Dr.Prajwal YB	B.PTh, CCCE	Cardiopulmonary	15
<div><div><div></div></div></div>	<div><div><div></div></div><div>Edit</div></div>	<div><div><div></div></div><div>Copy</div></div>	<div><div><div></div></div><div>Delete</div></div>	3	Dr.Neha R	BPT, MPT (Neuro)	Neurology	26
<div><div><div></div></div></div>	<div><div><div></div></div><div>Edit</div></div>	<div><div><div></div></div><div>Copy</div></div>	<div><div><div></div></div><div>Delete</div></div>	4	Dr.Yogesh Karan	Bachelor of Physiotherapy	Cardiorespiratory	29
<div><div><div></div></div></div>	<div><div><div></div></div><div>Edit</div></div>	<div><div><div></div></div><div>Copy</div></div>	<div><div><div></div></div><div>Delete</div></div>	5	Dr.Shilpa Gr	Sports Medicine (ABPMR)	Sports Physiotherapy	19

## 4.7 CREATION OF TRIGGERS

The trigger is made such that when a new record is inserted into a APPOINTMENT table, it automatically changes the lowercase name into uppercase in the backend.

### TRIGGER ON APPOINTMENT TABLE TO CHANGING NAME TO UPPERCASE

```

DELIMITER $$
CREATE TRIGGER UPPERCASE
BEFORE INSERT on bookapt
FOR EACH ROW
BEGIN
SET NEW.firstname=UPPER(NEW.firstname);
SET NEW.lastname=UPPER(NEW.lastname);
END$

```

Edit trigger

Details

Trigger name
UPPERCASE

Table
bookapt

Time
BEFORE

Event
INSERT

Definition

```

1 BEGIN
2 SET NEW.firstname=UPPER(NEW.firstname);
3 SET NEW.lastname=UPPER(NEW.lastname);
4 END

```

Definer
root@localhost

## STORED PROCEDURE // TRIGGER // APPOINTMENT TABLE

Here is the DISPLAY of.....

Stored Procedure & Trigger....

Procedure & Trigger Created Successfully.

Calling Stored Procedure & Trigger(UpperCase)!!!

Book ID	First Name	Last Name	Date of Birth	Age	Email
7	CRISTIANO	RONALDO	1975-02-04	44	criscr7@gmail.com
11	PRANAV	RAO	1998-10-10	21	prao@gmail.com
12	PRAJWAL	YB	1997-05-13	22	praj@yahoo.com
13	NEHA	RAMESH	1999-04-01	20	neha@gmail.com

## 4.8 CREATION OF STORED PROCEDURES

This stored procedure is used to find the age of the given customer using date of birth and current date.

### STORED PROCEDURE ON CUSTOMER TABLE TO FIND AGE

```
DELIMITER $$

CREATE PROCEDURE GetAge()

BEGIN

SELECT *, year(CURRENT_DATE())-year(DOB) as age from bookapt;

END$$
```

Edit routine

Details

Routine name	GetAge				
Type	PROCEDURE ▼				
Parameters	Direction	Name	Type	Length/Values	Options
Add parameter					
Definition	<pre> 1 BEGIN 2 SELECT *,year(CURRENT_DATE())-year(dob) as age from bookapt; 3 END </pre>				
Is deterministic	<input type="checkbox"/>				
Adjust privileges	<input checked="" type="checkbox"/>				
Definer	`root`@`localhost`				
Security type	DEFINER ▼				
SQL data access	CONTAINS SQL ▼				
Comment					

## CHAPTER 5

### FRONT END DESIGN

#### 5.1 SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. System design could see it as the application of systems theory to product development. There is some overlap with the disciplines of system analysis, system architecture and systems engineering. If the border topic of product development “blends the perspective of marketing, design, and manufacturing into a single approach to product development,” then design is the act of taking the marketing information and creating the design of the product to be manufactured. System design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

Until the 1990’s systems design had a crucial and respected role in the data processing industry. In 1990’s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.

Object-oriented analysis and design methods are becoming the most widely used methods for computer system design. The UML has become the standard language in object-oriented analysis and design. It is widely used for modelling software systems and organizations.

System design is one of the most important phases of software development process. The purpose of the design is to plan the solution of a problem specified by the requirement documentation. In other words, the first step in the solution to the problem is the design of the project.

## 5.2 FRONT END CODE

### 5.2.1 CREATING FRONT END PAGE TO LINK ALL TABLE

```
<!DOCTYPE html>

<html>

<head>

<title>DBMS</title>

<link rel="stylesheet" type="text/css" href="fstpg.css">

</head>

<body>

<font size="9"><p style="text-align:center;">Physiotherapy Database </font></p><br>

<a href="search.php">Search </a></i><br>

<a href="delete.php">Delete </a></i><br><ul>

<li><a href="pysiothe.html">What is Physiotherapy?</a></li>

<li><a href="physiolist.php">Our Physiotherapists</a></li>

<!--<li><a href="FormSolution.html"><h2>Register with us!</h2></a></li>-->

<li><a href="appointment.php">Book an appointment</a></li>

<li><a href="viewappt.php">View your appointment</a></li>

<li><a href="fbback.php">Patient Feedback</a></li>

<li><a href="Ct.php">Contact us</a></li>

<li><a href="welcome.php">Sign Out</a></li> </ul><br><br><br><br>

</body>

</html>
```

### 5.2.2 CREATION OF LOGIN PAGE

```
<?php

session_start();

if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){

header("location: login.php"); exit;}

?>

<!DOCTYPE html>

<html lang="en">
```

```
<head> <meta charset="UTF-8">

<title>Welcome</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">

<style type="text/css">

body{ background-color: rgba(255,0,255,0.2);

background: radial-gradient(circle, rgba(238,174,202,1) 0%, rgba(148,187,233,1) 100%);  }

</style></head>

<body>

<div class="page-header">

    <marquee behavior="scroll" direction="right" scrollamount="12"><h1>Hi, <b><?php echo
($_SESSION["username"]);?></b>

    <a href="firstpage.html"><h2><b>Click here to access the Main Page!!</b></h2></a><br><br>

    <a href="logout.php" class="btn btn-danger">Sign Out of Your Account</a> </p>

</body>

</html>
```

### **5.2.3 INSERTION IN FRONT END CODE**

```
<?php

if(isset($_POST['firstname']) && isset($_POST['lastname']) && isset($_POST['email'])):

    $firstname = $_POST['firstname'];

    $lastname = $_POST['lastname'];

    $dateofbirth= $_POST['dob'];

    $email = $_POST['email'];

    $link = new mysqli('localhost','root','','physio');

    if($link->connect_error)

        die('connection error: '.$link->connect_error);

    $sql = "INSERT INTO bookapt (firstname, lastname, dob, email) VALUES
('".$firstname."', '".$lastname."', '".$dateofbirth."', '".$email."')";

    $result = $link->query($sql);

    if($result > 0):

        include('paymentnow.html');
```

```

else:
    echo 'ERROR: Could not able to execute $sql.';
endif;

$link->close();

die();

endif;

?>

<!DOCTYPE html>

<html lang="en"><html>


<head><title>Appointment</title><br><br>
<font size="8"><p style="text-align: center;">Book an appointment here!!!</p></font>

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

<div><font size="5"><p style="text-align:center;"><label for="first_name">First
Name:</p></label></font>

<font size="5"><p style="text-align:center;"><input type="text" name="firstname"
id="first_name" placeholder="Firstname" required /></p></font>

<font size="5"><p style="text-align:center;"><label for="last_name">Last
Name:</label></p></font>

<font size="5"><p style="text-align:center;"><input type="text" name="lastname"
id="last_name" placeholder="Lastname" required /></p></font>

<font size="5"><p style="text-align:center;"><label for="dob">Date of
Birth:</label></p></font>

<font size="5"><p style="text-align:center;"><input type="date" name="dob" id="dob"
placeholder="Enter Birthdate" required /></p></font>

<font size="5"><p style="text-align:center;"><label for="email">Email
ID:</label></p></font>

<font size="5"><p style="text-align:center;"><input type="text" name="email"
id="last_name" placeholder="email" required /></p></font>

<font size="5"><p style="text-align:center;"><input type="submit" value="Submit"
/></p></font><font size="5"><p style="text-align:center;"><input type="reset"
value="reset" /></p></font><br> </div><br>

</form>

</body>

</html>

```



**5.2.4 SEARCHING OF VALUES FROM FRONT END**

```
<html>
<head> <title>Search Physiotherapist</title></head>
div{  font-family: "verdana";
      font-weight: bold;
      font-size: 30px;
      font-style: bold;
      margin-left:25px;
      margin-top: 35px;  }
span{
  font-family: "verdana";
  background-color: lightcyan;
  color: black;
  margin-top:4px;
  border-radius: 8px;
  text-align: center;
  font-size: 30px;
  margin-left:0px;
  width: 35%;
  font-weight: bold;
}
<body style="background-color: lavender">
  <h1><center><font style="border:9px solid grey" face="arial">SEARCH FROM
  PHYSIOTHERAPIST TABLE </font></center></h1>
  <form action="search.php" method="POST">
  <div>Enter Physiotherapist ID:<input type="text" name="physioid"><br></div><br><br>
  <button type="submit" value ="Find" class="btn">SEARCH</button></form>
  <?php
  $host="localhost";
  $user="root";
  $password="";
  $con= new mysqli($host,$user,$password,"physio");
```

```

if ($con->connect_error) {
    die("Connection failed: " . $con->connect_error);
}
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
    $n1=$_POST['physioid'];
    echo "<b><br>Entered Physiotherapist ID is $n1<br></b>";
    $sql="SELECT * from physiotherapist where physioid='$n1'";
    $result = $con->query($sql);
    if ($result->num_rows > 0) {
        echo "<b><br>Search Successful<br><br></b>";
        while($row = $result->fetch_assoc()) { } else {
            echo "<span><br><br>OPPS!!! Search Unsuccessful!<br> such</span>"; }
        $con->close(); ?>
    }
</body>
</html>

```

### 5.2.5 DISPLAYING VALUES FROM FRONT END

```

<!DOCTYPE html>
<html>
<head>
<title>View Appointment</title><style>
td{
    padding: 12px;
    border-radius: 14px;
} tr:nth-child(odd) {background-color: #f2f2;
    border-radius: 14px;} </style>
</head><body style="background-color: #EBF4FA">
    <h1><center><font style="border:9px solid #736AFF"> STORED PROCEDURE /\
    TRIGGER /\ APPOINTMENT TABLE </font></center></h1>
<table>
<tr>
<th><br>Book ID<br><br></th>

```

```

<th><br>First Name<br><br></th>
<th><br>Last Name<br><br></th>
<th><br>Date of Birth<br><br></th>
<th><br>Age<br><br></th>
<th><br>Email<br><br></th><br><br></tr>
<?php
$con = mysqli_connect("localhost", "root", "", "physio");
echo " <b><center>Here is the DISPLAY of.....</center></b>";
echo " <b><center>Stored Procedure & Trigger....</center></b>";
if ($con->connect_error) {
    die("Connection failed: " . $con->connect_error); }
    $sql = "CREATE PROCEDURE GetAge() SELECT *, year(current_date())-year(dob) as age
from bookapt";
    mysqli_query($con,$sql);
    echo "<b><center>Procedure & Trigger Created Successfully.</center></b>";
    echo "<b><center>Calling Stored Procedure & Trigger(UpperCase)!!!</center></b>";
    if ($result = mysqli_query($con,"CALL GetAge()"))
        {   while($row = $result->fetch_assoc())   {
            echo "<tr><td>$row[\"bookid\"]</td><td>$row[\"firstname\"]</td><td>\" .
$row[\"lastname\"]. \"</td><td>\" . $row[\"dob\"]. \"</td><td>\" . $row[\"age\"]. \"</td><td>\" .
$row[\"email\"]. \"</td></tr>\";   }
            echo "</table>\";   }
        else {   echo "0 results\";   } $con->close();?>
</table></body>
</html>

```

### 5.2.6 DELETION OF VALUES FROM FRONT END

```

<html>
<head>
    <title>Delete your Appointment</title>
</head>
<style>
.btn:hover
{   opacity: 1;
    background-color:forestgreen; }
span{

```

```

font-family: "verdana";
background-color: lightcyan;
color: black;
margin-top:4px;
border-radius: 8px;
text-align: center;
font-size: 30px;
margin-left:0px;
width: 35%;
font-weight: bold; }
</style>
<body style="background-color: lavender">
<h1><center><font style="border:9px solid grey" face="arial">DELETE
</font></center></h1>
<form action="delete.php" method="POST">
<div>Enter your BookID:<input type="text" name="bookid"><br></div>
<button type="submit" value ="Find" class="btn">Delete</button></form>
<?php
$host="localhost";
$user="root";
$password="";
$con= new mysqli($host,$user,$password,"physio");
if ($con->connect_error) {
    die("Connection failed: " . $con->connect_error);
}
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
    $n2=$_POST['bookid'];
    echo "<b><br>Entered BOOK ID is $n2<br></b>";

    $sql="DELETE from bookapt where bookid='$n2'";
    $result = $con->query($sql);
    if ($result > 0) {
        echo '<span style="text-align: center;">Deleted Successfully</span>';
    } else {
        echo "<span><br><br>OPPS!! Delete Unsuccessful!<br><br>There is no such
BOOKING ID exists. Please Enter Correct BOOKING ID.</span>"; }}$con->close();
?>
</body>
</html>

```

## CHAPTER 6

# TESTING

This chapter gives the outline of all testing methods that are carried out to get a bug free system. Quality can be achieved by testing the product using different techniques at different phases of the project development. The purpose of testing is to discover error. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components sub-assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## 6.1 TESTING PROCESS

Testing is an integral part of software development. Testing process certifies whether the product that is developed complies with the standards that I was designed to. Testing process involves building of test cases against which the product has to be used.

## 6.2 TESTING OBJECTIVE

The main objectives of testing process are as follows.

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has high probability of finding undiscovered error.
3. A successful test is one that uncovers the undiscovered error.

## 6.3 TEST CASE

The test cases provided here test the most important features of the project.

### 6.3.1 Test cases for the project Physiotherapy Database

S.no	Test Input	Expected Results	Observed Results	Remarks
1	INSERT A RECORD	New tuple should be inserted	Query OK 1 row effected or inserted	PASS
2	DISPLAY A RECORD	Display the record	Record displayed	PASS
3	DELETE A RECORD	Delete the record	Record Deleted	PASS
4	SEARCH A RECORD	search Update the record the record	Query OK 1 row affected or Row searched	PASS
5	CREATE TRIGGER	Trigger created	Query OK Trigger Created	PASS
6	CREATE STORE PROCEDURE	Store procedure created	Query OK Stored Procedures Created	PASS

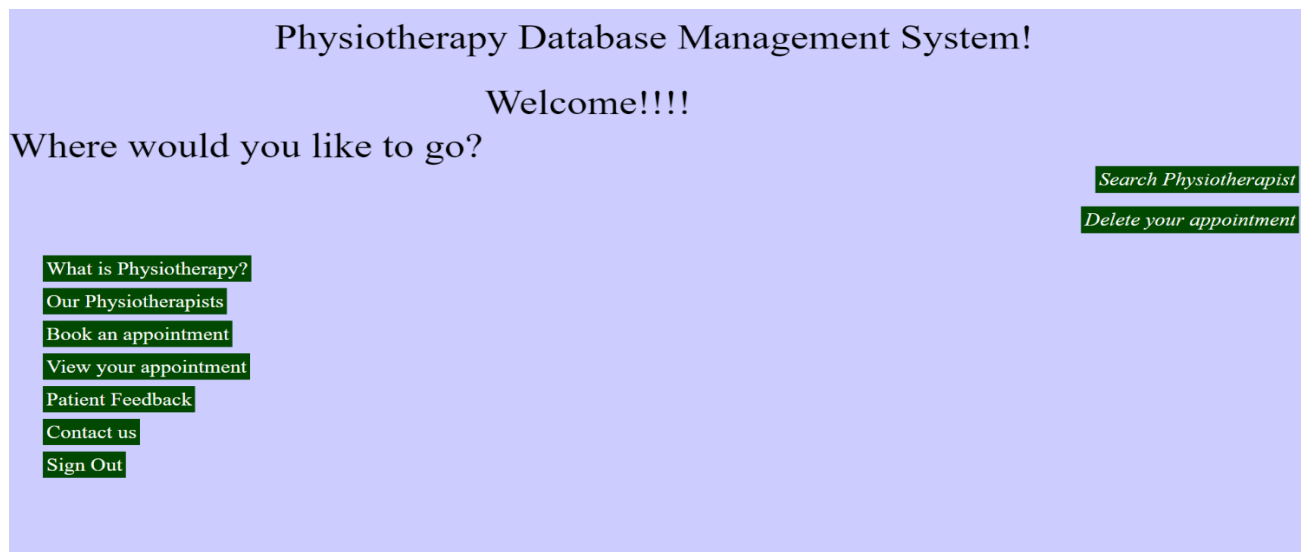
## CHAPTER 7

### RESULTS

This section describes the screens of “Physiotherapy Database”. The snapshots are shown below for each module.

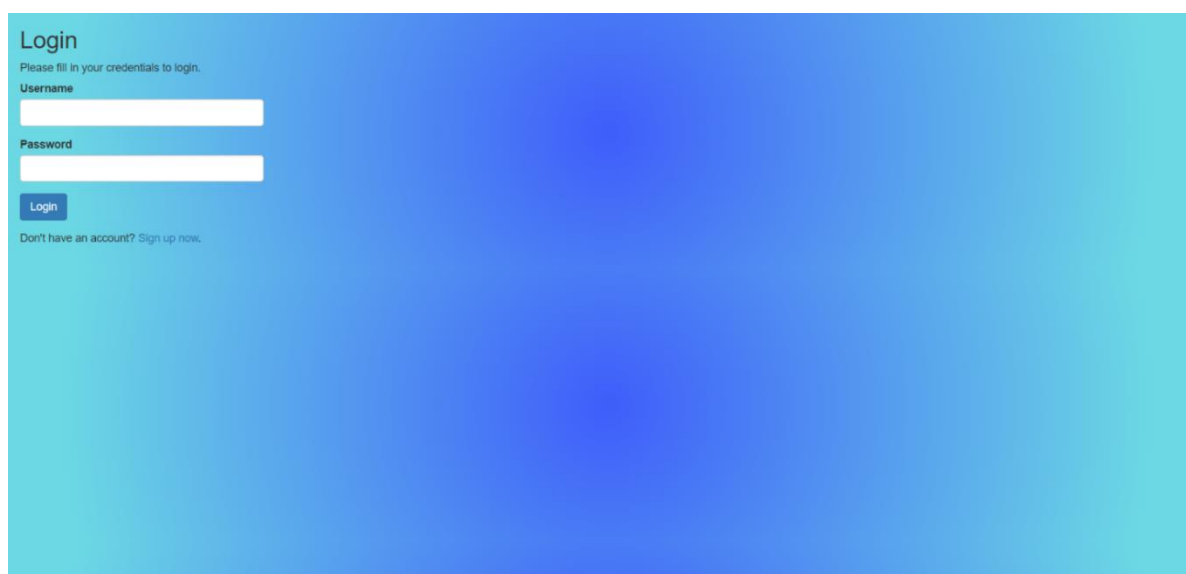
#### SNAPSHOTS

- This is the main page that shows all the operations which are present in Physiotherapy Database.



Snapshot 7.1 Physiotherapy Database Main Page

- This is the user **LOGIN IN** page



Snapshot 7.2 User Login Page

- This snapshot shows the **insertion page** of appointment table. This front end page supports the insertion of all the attributes in appointment like bookid, fullname, email.

**Welcome!!!**

**Book an appointment here!!!**

First Name:

Last Name:

Date of Birth:

Email ID:

**Snapshot 7.3: Insertion Page Of Appointment Table**

- This snapshot contains all the details of given PhysioID by using Search method. Search table of Physiotherapy contains the values like PhysioID, Name ,qualification etc.

**SEARCH FROM PHYSIOTHERAPIST TABLE**

Enter Physiotherapist ID:

**Entered Physiotherapist ID is 3**

**Search Successful**

Physiotherapist ID	Name	Qualification	Specialization	Experiences
3	Dr.Neha R	BPT, MPT (Neuro)	Neurology	26

**Snapshot 7.4: Search Page Of Physiotherapist Table**



- This snapshots **displays** all the values entered in that table. Here, BOOKAPT table is displayed on frontend page by using proper display query.

DISPLAY CONTENTS // PHYSIOTHERAPIST TABLE				
Physio ID	Full Name	Qualification	Specilization	Experiences
1	Dr.Pranav Rao	Bachelor of Physiotherapy	Cardiorespiratory	20
2	Dr.Prajwal YB	B.PTh, CCCE	Cardiopulmonary	15
3	Dr.Neha R	BPT, MPT (Neuro)	Neurology	26
4	Dr.Yogesh Karan	Bachelor of Physiotherapy	Cardiorespiratory	29
5	Dr.Shilpa Gr	Sports Medicine (ABPMR)	Sports Physiotherapy	19

**Snapshot 7.5: Display Page Of Appointment Table**

- This snapshot shows the working status of Delete Page of APPOINTMENT Table. In this page, **bookid** is used as a parameter to delete certain record with all of their attributes in that rows. If the entered value is present in database then the value will be deleted.

DELETE FROM APPOINTMENT TABLE

Enter your BookID:

Delete

Entered BOOK ID is 19

!! Record Deleted Successfully!!

**Snapshot 7.6: Deletion Page Of Appointment Table**

- This snapshot shows the working status of **Stored Procedure & Trigger** of APPOINTMENT table. By using stored procedure, we're calculating the Age of USER using Date of Birth as parameter where as in case of trigger we are taking users name in uppcase.

STORED PROCEDURE // TRIGGER // APPOINTMENT TABLE					
Here is the DISPLAY of.....					
Stored Procedure & Trigger....					
Procedure & Trigger Created Successfully.					
Calling Stored Procedure & Trigger(UpperCase)!!!					
Book ID	First Name	Last Name	Date of Birth	Age	Email
7	CRISTIANO	RONALDO	1975-02-04	44	criscr7@gmail.com
11	PRANAV	RAO	1998-10-10	21	prao@gmail.com
12	PRAJWAL	YB	1997-05-13	22	praj@yahoo.com
13	NEHA	RAMESH	1999-04-01	20	neha@gmail.com

**Snapshot 7.7: Stored Procedure & Trigger Of Appointment Table**

## CONCLUSION

In Physiotherapy database management we have made an attempt to create an automated system where an user could create their account in the physiotherapy website and would be able to access the website. User would be able to book an online appointment by paying Rs 100 amount and also the user would be able to search the list of Physiotherapists in order to get the requisite information about the physiotherapist by giving the PhysioID. This system would also enable the user to share their contact details so that they could be able to get more information. User is also enable to delete his appointment .At the end the user have the opportunity to provide a sincere feedback for the improvement of the website. This project also helps me in developing many soft skills like time management, allocation of resources and building up of confidence on your work.

The database structure is quite simple, which makes it easy for also other programmers to understand it. In conclusion, a database is a far more efficient mechanism to store and organize data than spreadsheets it allows for a centralized facility that can easily be modified and quickly shared among multiple users. Having a web based front end removes the requirement of users having to understand and use a database directly, and allows users to connect from anywhere with an internet connection and a basic web browser. It also allows the possibility of queries to obtain information for various surveys. Due to the number of users reading and modifying data it is an ideal use for such a system.

## REFERENCES

a) Textbooks

1. Fundamentals of Database System, 7<sup>th</sup> Edition-By Elmasri Ramez and Navathe Shamkanth
2. PHP and MySQL Web Development-By Luke Welling and Laura Thompson
3. HTML & CSS: Design and Build Web Sites-By John Duckett

b) Websites

1. For Front End Code and CSS

- 1.1 <https://www.w3schools.com/html>
- 1.2 <https://www.stackoverflow.com>
- 1.3 <https://www.tutorialsrepublic.com/PHP>

c) Video tutorials

1. <https://www.udemy.com>
2. <http://www.youtube.com>