CHAPTER-1

INTRODUCTION

Data is unprocessed information which is a known fact that can be recorded and specifies implicit meaning. A **database** is the collection of related data organized in a way that data can be easily accessed, managed and updated. Database is actually a place where related piece of information is stored and various operations can be performed on it. Database can be recorded manually or computerized. The size and complexity of database is variable database is designed, built and populated for specific purpose.

OUR PROJECT:

Nowadays, the apartments are increasing day by day. People coming from abroad in order to settle down in the city, they will not be knowing the information about the apartments. As the name suggests, the apartment management system helps in maintaining the information of the people in the particular apartment, number of apartments available and so on. The user interface must be simple and easy to understand even by the common man. This application will help in reducing the pen paper work through which we store the details of the people who stay in the apartments. Their details can be obtained just in one click and with great ease. This will be one of the interesting projects that one can work on and implement in real time since it will be very useful to the owners of the apartments to maintain the details of the people with great ease.

As our software is coded in html/php/javascript, so it is platform independent i.e. It can work on any operating system whether it can be any version of Microsoft window, Linux, Mac OS or any other.

1.1 DBMS:

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible. The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified -- and the database schema, which defines the database's logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity.

A database management system can limit what data the end user sees as well as how the end user can view the data providing many views of a single database schema. End user and software programs are free to understand where the data is physically located or on what type of storage media it resides because the database management handles all. Example: MySQL, Oracle etc.… The general-purpose DBMS allows the definition, creation, querying, update and administration of databases.

There are four structural types of database management systems: Hierarchical databases, Network databases, Relational databases, Object oriented database.

The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data (storage and hardware). As long as programs use the application programming interface (API) for the database that is provided by the DBMS, developers won't have to modify programs just because changes have been made to the database.

With relational DBMSs (RDBMSs), this API is SQL, a standard programming language for defining, protecting and accessing data in a RDBMS.

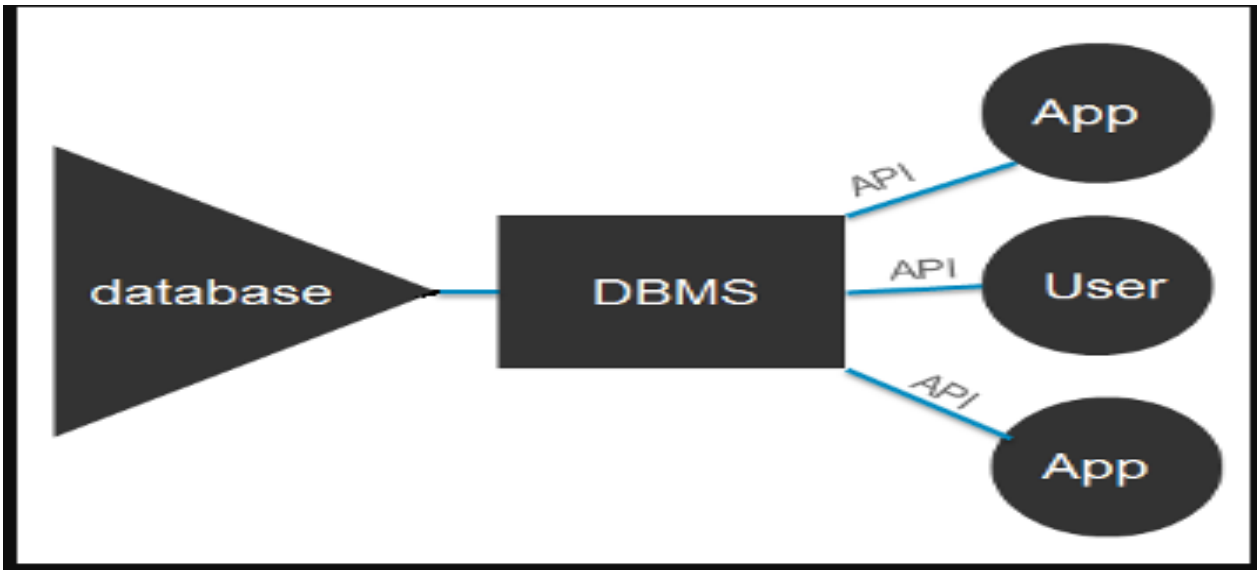


Fig 1.1: DBMS

- **ADVANTAGES OF DBMS:**

Using a DBMS to store and manage data comes with advantages, but also overhead. One of the biggest advantages of using a DBMS is that it lets end users and application programmers access and use the same data while managing data integrity. Data is better protected and maintained when it can be shared using a DBMS instead of creating new iterations of the same data stored in new files for every new application. The DBMS provides a central store of data that can be accessed by multiple users in a controlled manner.

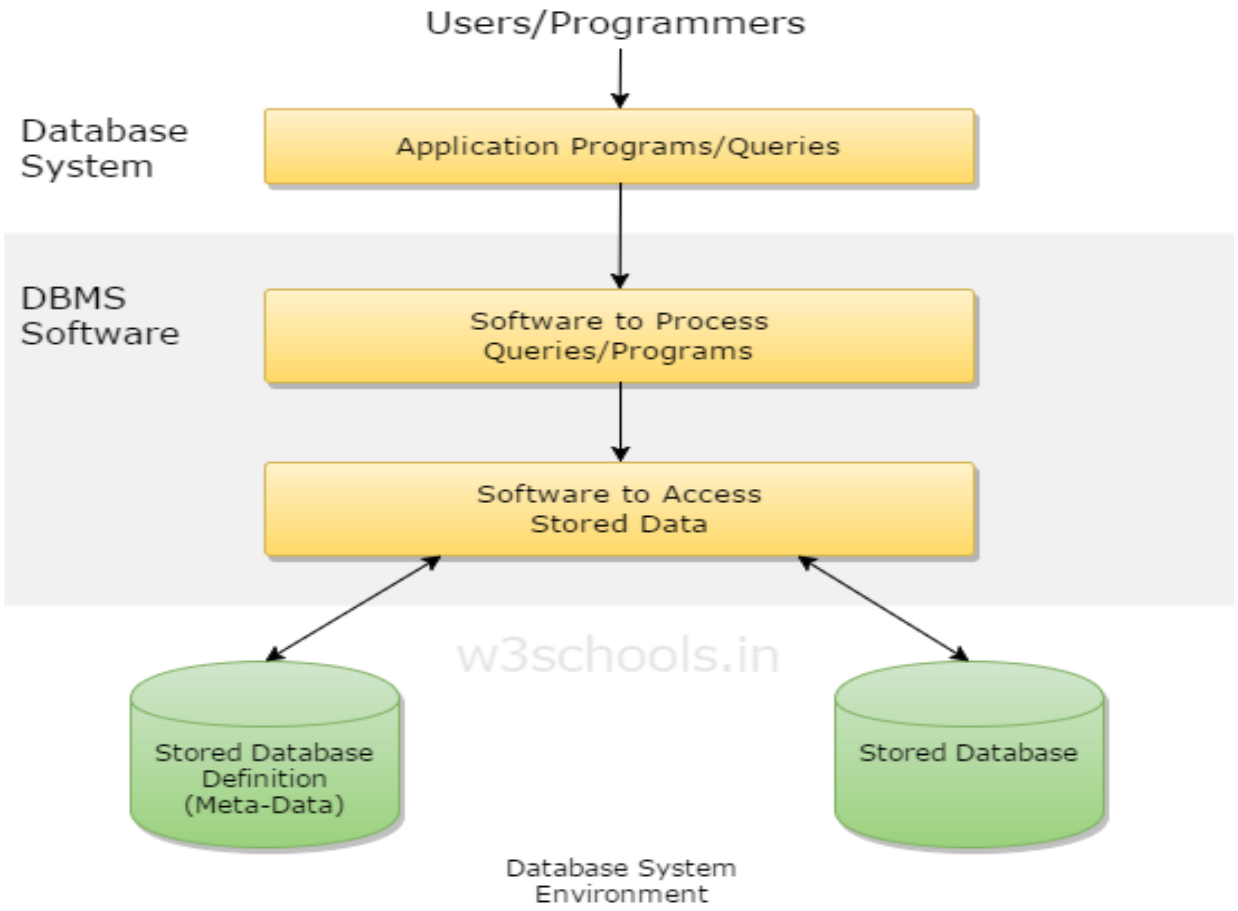


Fig 1.2: Database Environment

Central storage and management of data within the DBMS provides:

- Data abstraction and independence
- Data security
- A locking mechanism for concurrent access
- An efficient handler to balance the needs of multiple applications using the same data
- The ability to swiftly recover from crashes and errors, including restart ability and recoverability
- Robust data integrity capabilities
- Logging and auditing of activity
- Simple access using a standard application programming interface (API)

1.2: MYSQL:

MYSQL is an open source relational database management system (RDBMS) based on Structured Query Language (SQL). MYSQL runs on virtually all platforms, including Linux, UNIX, and Windows. Although it can be used in a wide range of applications, MYSQL is most often associated with web-based applications and online publishing and is an important component of an open source enterprise stack called LAMP. LAMP is a Web development platform that uses Linux as the operating system, Apache as the Web server, MYSQL as the relational database management System and PHP the object-oriented scripting language.(Sometimes Perl or Python is used instead of PHP).MYSQL is a full-featured relational database management system (RDBMS) that competes with the likes of Oracle DB and Microsoft's SQL Server. MYSQL is sponsored by the Swedish company MYSQL AB, which is owned by Oracle Corp.

MYSQL was a free-software database engine originally developed and first released in 1995. MYSQL is named after my, the daughter Michael Widenus of one of the product's originators. It was originally produced under the GNU General Public License, in which source code is made freely available. MYSQL is very popular for Web-hosting applications because of its plethora of Web-optimized features like HTML data types, and because it's available for free. It is part of the Linux, Apache, MYSQL, PHP (LAMP) architecture, a combination of platforms that is frequently used to deliver and support advanced Web applications. MYSQL runs the back-end databases of some famous websites, including Wikipedia, Google and Facebook - a testament to its stability and robustness despite its decentralized, free-for-all philosophy.

- **ADVANTAGES OF MYSQL:**

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

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

- It's Open-Source (Sort Of): Oracle's purchase of Sun Microsystems. The general fear was that Oracle would transform the tool into a closed, proprietary ecosystem. Thankfully, though Oracle has tightened its grip on MYSQL somewhat, it can still be considered an open-source database option, as the code is still available for free online.

- It's Incredibly Inexpensive: Depending on what you plan to use it for, a MYSQL implementation could range in price from free to $10, 0000 or more. Either way, it's significantly less expensive than most other database options on the market.

- It's An Industry Standard (And Still Extremely Popular): Although MySQL popularity has waned somewhat in recent years, it remains one of the most-used database systems in the world. It's compatible with virtually every operating system, and is more or less an industry standard. This is, of course, in spite of all the folks who say it's on the way out.

Following are the basic syntaxes of MySQL with examples:

CREATE:-

CREATE TABLE table_name (column_name column_type);

INSERT:-

INSERT INTO table_name (field1, field2,…. fieldN) VALUES (value1, value2,….valueN) ;

To insert string data types, it is required to keep all the values into double or single quotes.

SELECT:-

SELECT field1, field2…fieldN from table_name1, table_name2 [WHERE Clause];

DELETE:-

DELETE FROM table_name [WHERE Clause];

UPDATE:-

UPDATE table_name

SET attribute = 'value' [WHERE Clause]

1.3 HTML:

HTML (Hypertext Markup Language) is the set of mark-up symbols or codes inserted in a file intended for display on a World Wide Web browser page. The mark-up tells the Web browser how to display a Web page's words and images for the user. Each individual mark-up code is referred to as an element (but many people also refer to it as a tag). Some elements come in pairs that indicate when some display effect is to begin and when it is to end.

HTML is a formal Recommendation by the World Wide Web Consortium (W3C) and is generally adhered to by the major browsers, Microsoft's Internet Explorer and Netscape's Navigator, which also provide some additional non-standard codes. The current version of HTML is HTML 4.0. However, both Internet Explorer and Netscape implement some features differently and provide non-standard extensions. Web developers using the more advanced features of HTML 4 may have to design pages for both browsers and send out the appropriate version to a user. Significant features in HTML 4 are sometimes described in general as dynamic HTML. What is sometimes referred to as HTML 5 is an extensible form of HTML called Extensible Hypertext Markup Language (XHTML).

- **ADVANTGES OF HTML:**

- HTML is easy to use and understand
- All browsers support HTML
- HTML and XML syntax is very similar
- Most development tools support HTML
- HTML is most search engine friendly

How does it work?

HTML consists of a series of short code typed into a text-file by the site author-these are the tags. The text is then saved as an html file and viewed through a browser link internet explorer. The browser reads the file and translates the text into a visible form.HTML documents from a web server or from local storage and render them into multimedia web pages.HTML describes the structure of a web pages semantically and originally included for the appearance of the document.

HTML stands for Hypertext Mark-up Language, and is used to describe the visual appearance of

a document to be displayed by an internet browser. HTML documents consist of document tags which act to directly describe the visual appearance of a web page or to provide a directive command such as inserting imagery or a link to another web page within a document. HTML documents are saved in text format and are designed to be viewed or edited on any operating system that is able to connect to the Internet. XHTML refers to the latest version(s) of the HTML definition that are designed to make use of the extensible mark-up language definition rules and syntax in order to permit web developers to continue to do Advanced web page development.

HTML Tags

The tags are what separate normal text from HTML code, you might know them as that words between the <angle-brackets>They allow all the cool stuffs like images and tables and stuffs just by telling your browser what to render on the page. Different tags perform different functions. Some of the basic tags are mentioned below:

HTML Tag:

SL.No	TAGS	DESCRPITION
1	<!DOCTYPE>	Defines the document type
2	<html>	Defines an html document
3	<head>	Defines information about the document
4	<title>	Defines a title of the document
5	<body>	Defines the document body
6	<h1> to <h6>	Defines HTML headings
7	<p>	Defines paragraph
8	 	Inserts a single line break
9	<hr>	Defines a thematic change in the content

1.4: PHP:

PHP is a script language and interpreter that is freely available and used primarily on Linux Web servers. PHP originally derived from Personal Home Page Tools, now stands for PHP: Hypertext Pre-processor, which the PHP FAQ describes as a "recursive acronym." PHP executes on the server, while a comparable alternative, JavaScript, executes on the client. PHP is an alternative to Microsoft's Active Server Page (ASP) technology. As with ASP, the PHP script is embedded within a Web page along with its HTML. Before the page is sent to a user that has requested it, the Web server calls PHP to interpret and perform the operations called for in the PHP script. An HTML page that includes a PHP script is typically given a file name suffix of ".php" ".php7," or ".phtml". Like ASP, PHP can be thought of as "dynamic HTML pages," since content will vary based on the results of interpreting the script. PHP is free and offered under a source license. The characteristics of PHP are Simplicity, Efficiency, Security, Flexibility, and Familiarity

COMMON USES OF PHP:-

PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them. The other uses of PHP are:

PHP can handle forms, i.e. gather data from files, save data to a file, through email we can send data, return data to the user. We can add, delete, modify elements within your database thru PHP. Access cookies variables and set cookies. Using PHP, you can restrict users to access some pages of your website. It can encrypt data.

- **ADVANTAGES OF PHP:**

- Open source: It is developed and maintained by a large group of PHP developers, this will helps in creating a support community, abundant extension library.
- Speed: It is relative fast since it uses much system resource.
- Easy to use: It uses C like syntax, so for those who are familiar with C, it's very easy for them to pick up and it is very easy to create website scripts.
- Stable: Since it is maintained by many developers, so when bugs are found, it can be quickly fixed.

- Powerful library support: You can easily find functional modules you need such as PDF, Graph etc.
- Built-in database connection modules: You can connect to database easily using PHP, since many websites are data/content driven, so we will use database frequently, this will largely reduce the development time of web apps.
- Can be run on many platforms, including Windows, Linux and Mac, it's easy for users to find hosting service providers.

1.5 JAVASCRIPT:

A script is a small piece of program that can add interactivity to your website. For example, a script could generate a pop-up alert box message, or provide a dropdown menu. This script could be written using JavaScript or VBScript.

You can write various small functions, called event handlers using any of the scripting language and then you can trigger those functions using HTML attributes.

Now-a-days, only JavaScript and associated frameworks are being used by most of the web developers, VBScript is not even supported by various major browsers.

You can keep JavaScript code in a separate file and then include it wherever it's needed, or you can define functionality inside HTML document itself. Let's see both the cases one by one with suitable examples.

External Javascript:

If you are going to define a functionality which will be used in various HTML documents then it's better to keep that functionality in a separate JavaScript file and then include that file in your HTML documents. A JavaScript file will have extension as **.js** and it will be included in HTML files using <script> tag.

Example:

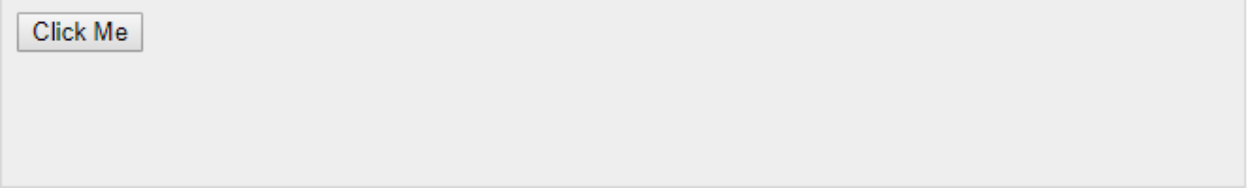
Consider we define a small function using JavaScript in **script.js** which has following code –

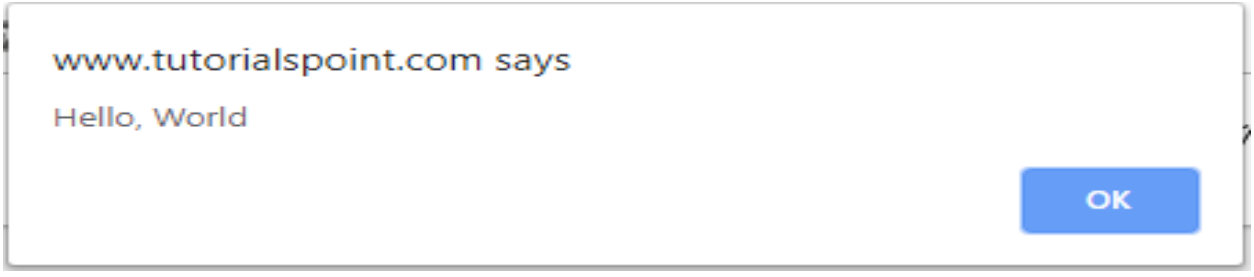
```
function Hello() {
   alert("Hello, World");
}
```

Now let's make use of the above external JavaScript file in our following HTML document –

```
<!DOCTYPE html>
<html>

   <head>
      <title>Javascript External Script</title>
      <script src = "/html/script.js" type =
"text/javascript"/></script>
   </head>

   <body>
      <input type = "button" onclick = "Hello();" name = "ok" value
= "Click Me" />
   </body>

</html>
```

This will produce the following result, where you can try to click on the given button –





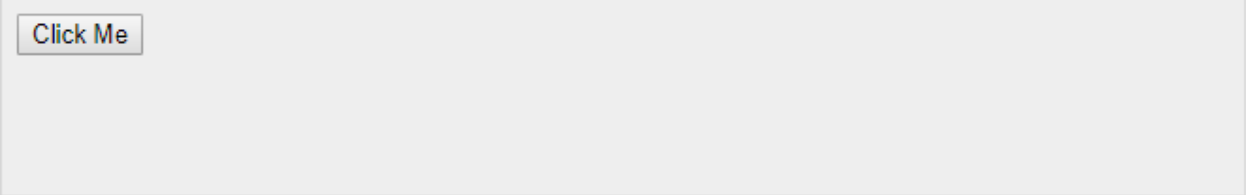
Internal Javascript:

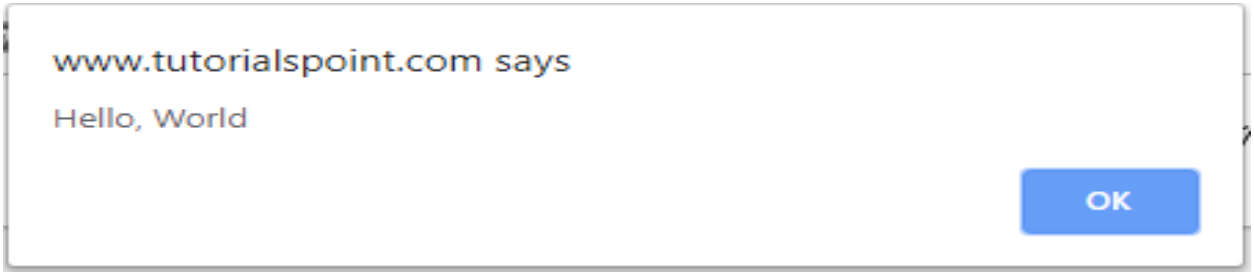
You can write your script code directly into your HTML document. Usually we keep script code in header of the document using <script> tag, otherwise there is no restriction and you can put your source code anywhere in the document but inside <script> tag.

Example:

```
<!DOCTYPE html>
<html>

   <head>
      <title>JavaScript Internal Script</title>
      <base href = "https://www.tutorialspoint.com/" />

      <script type = "text/JavaScript">
         function Hello() {
            alert("Hello, World");
         }
      </script>
   </head>

   <body>
      <input type = "button" onclick = "Hello();" name = "ok" value
= "Click Me" />
   </body>
</html>
```

This will produce the following result, where you can try to click on the given button –





1.6 XAMPP:

The most obvious characteristic of XAMPP is the ease at which a WAMP webserver stack could be developed and got running. Later some common packaged applications that could be easily installed were provided by Bitnami.

Officially, XAMPP's designers intended it for use only as a development tool, to allow website designers and programmers to test their work on their own computers without any access to the internet. To make this as possible, many important features are disabled by default.

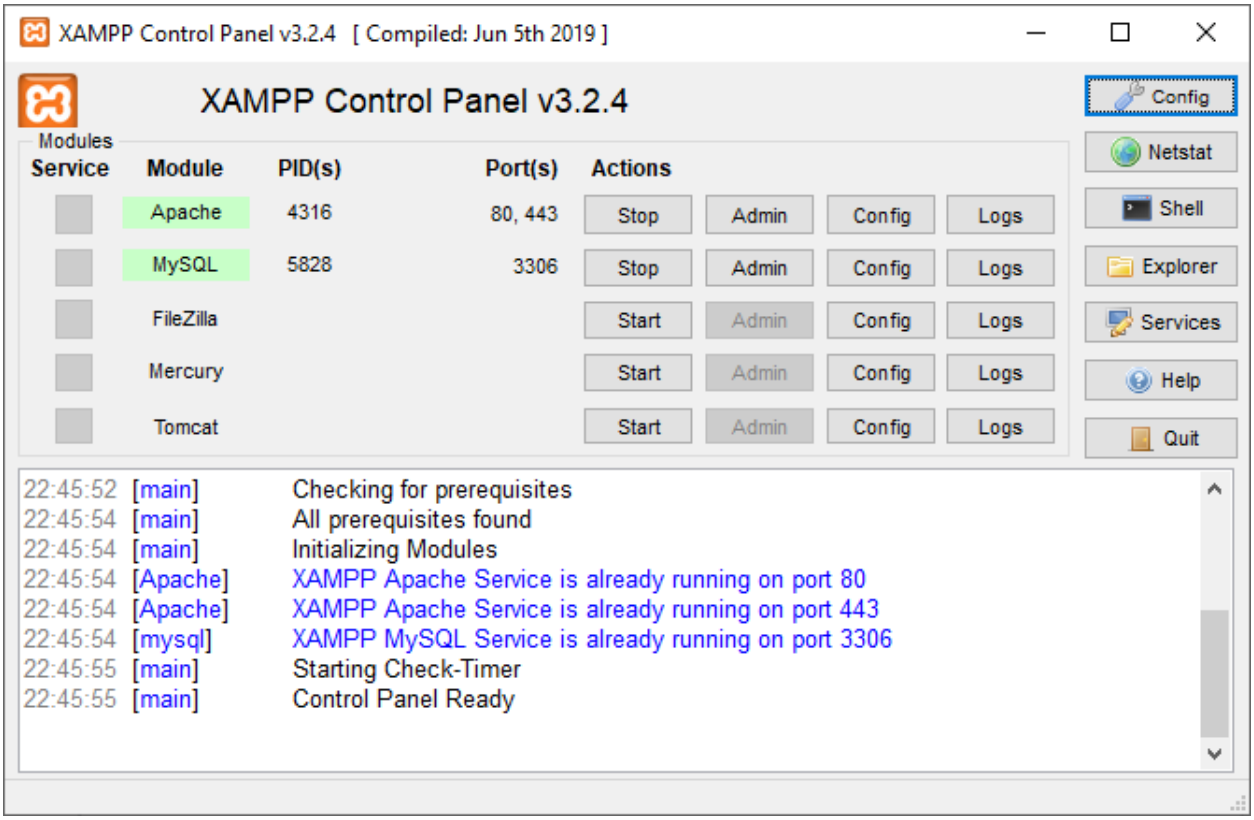


Fig 1.3: XAMPP

CHAPTER-2

SOFTWARE AND HARDWARE REQUIREMENTS

2.1 Hardware Requirements:

- Minimum 350MB Hard Disk space for installation
- 4GB HD space required for a typical live system with 1000-2000 events
- Recommended minimum CPU - Pentium 4, 3.2GHz
- Recommended 1GB RAM for a Central Server with 3 Nodes

2.2 SOFTWARE Requirements:

- Operating System: Windows OS.
- Backend: MySQL
- Coding Language: PHP
- Frontend: HTML

All the types of software automatically configured inside operating system after installation it has PHP, MySQL, Apache and operating system base configuration file.

CHAPTER 3

DESIGN

3.1 TABLES:

A table is a collection of related data held in a structured format within a database. It consists of columns, and rows. In relational databases, and flat filed databases, a table is a set of data elements (values) using a model of vertical columns (identifiable by name) and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows. Each row is identified by one or more values appearing in a particular column subset. The columns subset which uniquely identifies a row is called the primary key. "Table" is another term for "relation"; although there is the difference in that a table is usually a multistep (bag) of rows where a relation is a set and does not allow duplicates. Besides the actual data rows, tables generally have associated with them some metadata, such as constraints on the table or on the values within particular column. The data in a table does not have to be physically stored in the database. Views also function as relational tables, but their data are calculated at query time.

3.1 TABLES:

APARTMENT ATTRIBUTES

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	Apt_ID	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
☐	2	Build_ID	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
☐	3	Price	int(10)			No	None			Change Drop More
☐	4	No_of_Rooms	int(1)			No	None			Change Drop More
☐	5	Apt_For	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More

BUILDING ATTRIBUTES

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Build_Name	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
2	Build_ID	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
3	No_of_Apts	int(3)			No	None			Change Drop More

LEASE ATTRIBUTES

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Lease_ID	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
2	Apt_ID	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
3	Deposit	int(10)			No	None			Change Drop More
4	Start_Date	date			No	None			Change Drop More
5	End_Date	date			No	None			Change Drop More

LOGIN ATTRIBUTES

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	UserName	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
2	Password	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More

OWNER ATTRIBUTES

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	O_ID	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
2	Apt_ID	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
3	O_Name	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
4	DOB	date			No	None			Change Drop More
5	Gender	varchar(5)	utf8mb4_general_ci		No	None			Change Drop More
6	Occupation	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
7	Phone	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More

RENT ATTRIBUTES

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	Rent_ID	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
☐	2	Apt_ID	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
☐	3	Rent_Fee	int(6)			No	None			Change Drop More
☐	4	Late_Fee	int(5)			No	None			Change Drop More
☐	5	Due_Date	date			No	None			Change Drop More

3.2 ER Diagram:

An entity-relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

- Entities
- Relationships
- Attributes

Steps involved in creating an ERD include:

1. Identifying and defining the entities.
2. Determining all interactions between the entities.
3. Analysing the nature of interactions/determining the cardinality of the relationships.
4. Creating the ERD.

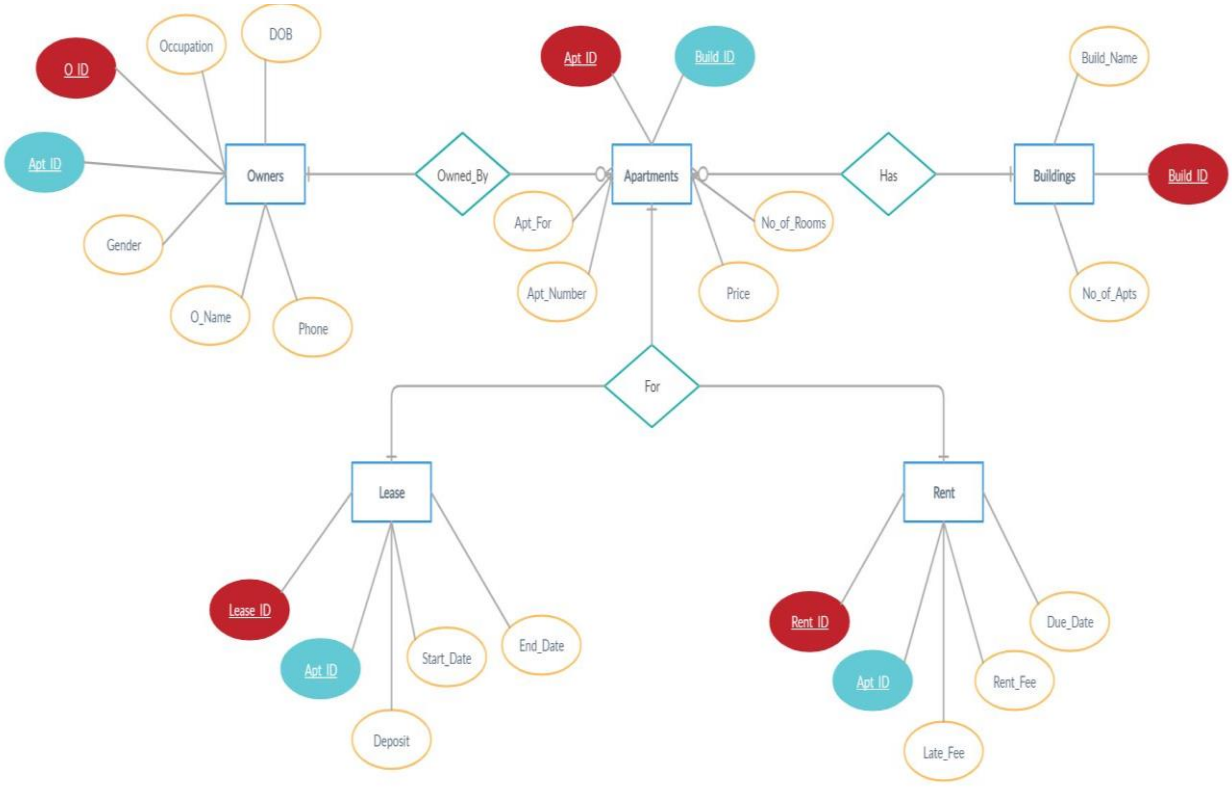


Fig 3.2: ER Diagram

3.3 Schema Diagram:

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams.

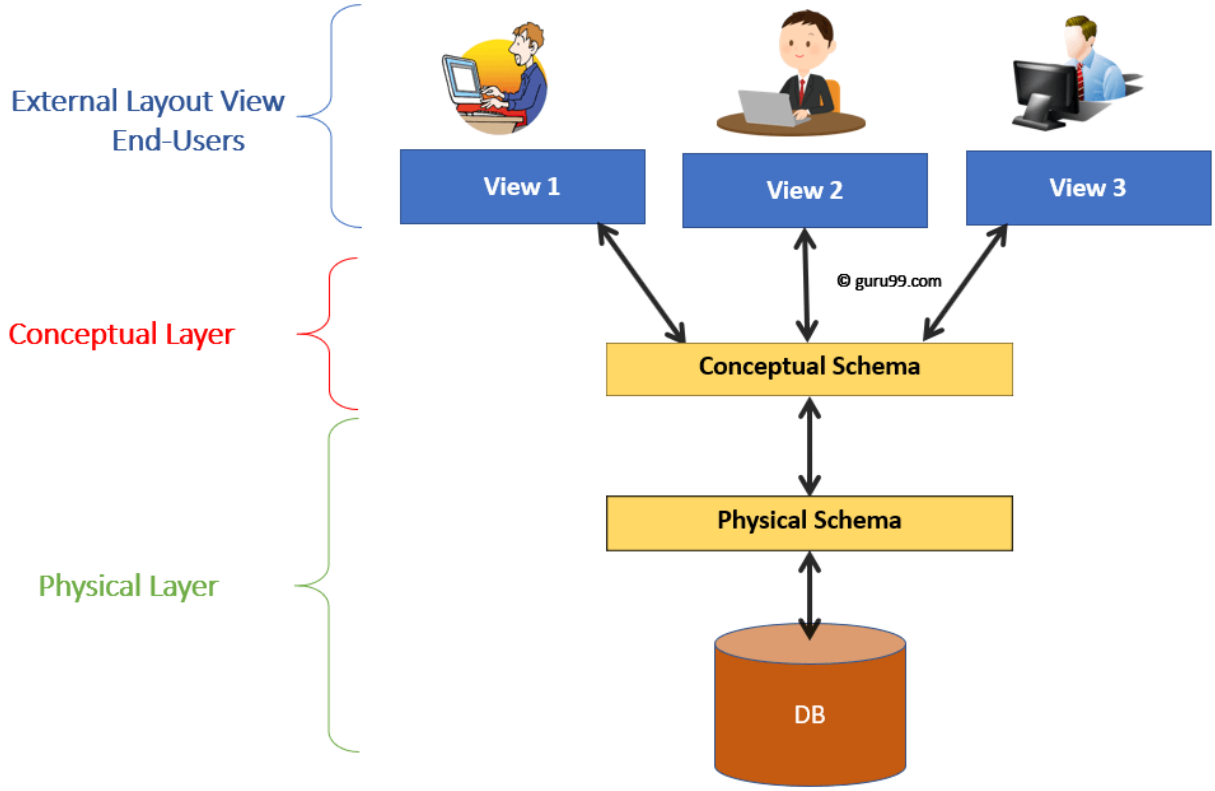


Fig 3.2: Structural view of schema diagram

A database schema can be divided broadly into two categories:-

- PHYSICAL DATABASE SCHEMA − this schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- LOGICAL DATABASE SCHEMA − this schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

BUILDING

Build_ID	Build_Name	No_of_Apts

APARTMENT

Apt_ID	Build_ID	No_of_Rooms	Price	**Apt_For**

OWNER

O_ID	Apt_ID	O_Name	DOB	Gender	Occupation	Phone

LEASE

Lease-ID	Apt_ID	Deposit	Start_Date	End_Date

RENT

Rent_ID	Apt_ID	Rent_Fee	Late_Fee	Due_Date

Fig 3.3: Schema Diagram

CHAPTER-4

IMPLEMENTATION

4.1 Description:

The system should have a login. A login box should appear when the system is invoked. The Admin should have all the type of authority. The Admin should maintain property. Admin identify property type as it is residential or commercial property. The Admin user can inform their agents for regarding to property and update the information regarding property and cancellation of property or changing buyer choice. The user should book the property for sell or rent with detail of property. The system is very useful for the companies or builders that can post and edit their properties and their personal info and admin can monitor records of all of them. The system is also useful which also keeps track of Account details of buyers.

The real of World Wide Web have spread across millions of households, so naturally, Internet has become by far the best platform for real estate marketing today. Now a day when everything is online, how is it possible that real estate left web application behind? There are lots of real estate companies who advertise their property online so idea behind developing this application is that their property can also sell, or buy or even rent property using this. These applications are not widely popular but in future, they have large scope of growth. This website is an online real estate management through which individual agents or buyer can maintain their property document keeping and managing property registration and access its information and manage all the adding, updating, deleting the ads and some of its tasks. The Admin user can inform their agents for regarding to property and update the information regarding property and cancellation of property or changing buyer choice. The system is very useful for the companies or builders that can post and edit the information of their properties and their personal info and admin can monitor records of all of them.

The system is also useful which also keeps track of Account details of buyers and Investors. The system should have a login. A login box should appear when the system is invoked. The Admin should have all the type of authority. The Admin should maintain property. The Admin user can inform their agents for regarding to property and update the information regarding property and cancellation of property or changing buyer choice.

The user should book the property for sell or rent with detail of property. The system is very useful for the companies or builders that can post and edit their properties and their personal info and admin can monitor records of all of them. The system is also useful which also keeps track of Account details of buyers and Investors and RES Industry. Admin's interface: Admin is a person who will handle the entire website. For that person must give the user name and password to enter the admin page. After entering right password admin person can enter the admin home area. Here user buy different property & sell them to the system. Buyer user property & builder verify each-other & make reliable communication to each other. User's interface: User can visit the home page of real estate in which first the introduction of our site mention first. The registered user can login from the login module. Here guest can register free account to sell and buy property &buyer verify each- other & make reliable communication to each other. User can search the property and it can select the type of property and its budget and also find the location of property.

4.2 Coding:

HTML CODE FOR LOGIN PAGE

```
<!DOCTYPE html>

<html>

<head>

      <title>Apartment DB</title>

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

<body>

<h1 title="By Shoaib And Imaad">

<bold>APARTMENT MANAGEMENT SYSTEM</bold>

</h1>

<img src="Img/logo.jpg" alt="LOGO">

<div class="background">

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

      <input type="text" name="username" placeholder="ENTER USER ID"
required> <br>

      <input type="password" name="password" placeholder="ENTER PASSWORD"
required><br>

      <button type="submit">LOGIN</button>

      </form>

</div>

</body>

</html>
```

PHP CODE FOR LOGIN PAGE:

```
<?php
session_start();

// Connect to Database
$con = mysqli_connect("localhost", "root", "");
if(isset($_POST['username'])){
    $username = $_POST['username'];
}
 $password = "";
if(isset($_POST['username'])){
    $password = $_POST['password'];
 }

//Select Database
 mysqli_select_db($con,"apt_db");



// Query the database for user
$result = mysqli_query($con,"select * from login where UserName = '$username'
and Password = '$password'")
  or die('Failed to query database'.mysqli_error());
 $row = mysqli_fetch_array($result);

// user and password verification
 if ( $row['UserName'] == $username && $row['Password'] == $password )
 {
      $_SESSION['uname'] = $username;
       $message="Login Succesfull. Welcome ".$_SESSION['uname']."";
                echo"<script > { alert('$message');}
window.location.replace('Menu.html');</script>";
 }
 else
 {
     $message="Credentials mismatch please try again";
                echo"<script > { alert('$message');}
window.location.replace('login.html');</script>";
}

?>
```

HTML CODE FOR INSERTION:

```
<!DOCTYPE html>
<html>
<head>
      <title>Insert Building</title>
      <link rel="stylesheet" type="text/css" href="style.css">
</head>
<script>
      function home()
    {
      window.location.replace('Menu.html');
    }

     function back()
    {
      window.location.replace('Insert.html');
    }
</script>
<body>
<button type="submit" onclick="home()">Home</button>
<button type="submit" onclick="back()">Back</button>

<form class="box" action="ins_building.php" method="post">

<h1>Add Building Details</h1>

<input type="text" name="b_name" placeholder="Building Name" required=""><br>

<input type="text" name="b_id" placeholder="Building ID" required=""><br>

<input type="text" name="no_of_apts" placeholder="Number Of Apartments"
required=""><br>

<button type="submit">Submit</button>

</form>

</body>
</html>
```

PHP CODE FOR INSERTION:

```
<?php
session_start();

$conn = new mysqli('localhost','root','');
if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }
mysqli_select_db($conn,"apt_db");
$x=$_POST['b_id'];
$sql = "SELECT * FROM building WHERE Build_ID='$x'";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
            $message="Building ID ALREADY IN USE";
            echo"<script>{alert('$message');}
window.location.replace('ins_building.html');</script>";
}

else {
$sql="INSERT    INTO    building    (Build_Name,Build_ID,No_of_Apts)    VALUES
('$_POST[b_name]','$_POST[b_id]','$_POST[no_of_apts]')";

if ($conn->query($sql) === TRUE) {
      $msg="Details Uploaded Successfully";
      echo"<script>{alert('$msg');}
window.location.replace('Insert.html');</script>";
}
else {
      echo "Error: " . $sql . "<br>" . $conn->error;
}
mysqli_close($conn);

}

?>
```

PHP CODE FOR UPDATION:

```
if (isset($_POST['update'])) {

            $apt_id = mysqli_real_escape_string($db, $_POST['apt_id']);

            $build_id = mysqli_real_escape_string($db, $_POST['build_id']);

            $no_of_rooms = mysqli_real_escape_string($db,
$_POST['no_of_rooms']);

            $price = mysqli_real_escape_string($db, $_POST['price']);

            $apt_for = mysqli_real_escape_string($db, $_POST['apt_for']);

      $updt_query = "UPDATE apartment SET Build_ID = '$build_id' ,
No_of_Rooms = '$no_of_rooms' , Price = '$price' , Apt_For = '$apt_for' WHERE
Apt_ID = '$apt_id'";

      mysqli_query($db, $updt_query);

      $_SESSION['message'] = "Details Updated!";

      header('location: show_apt.php');
```

PHP CODE FOR DELETION:

```
if (isset($_GET['del'])) {

  $apt_id = $_GET['del'];

  $del_query = "DELETE FROM apartment WHERE Apt_ID = '$apt_id'";

  mysqli_query($db, $del_query);

  $_SESSION['message'] = "Details Dropped!";

  header('location: show_apt.php');

}
```

HTML/PHP CODE TO DISPLAY DATA:

```
<?php

session_start();

$db = mysqli_connect('localhost', 'root', '', 'apt_db');

$show = "SELECT * FROM building";

$results = mysqli_query($db,$show);

?>

<!DOCTYPE html>

<html>

<head>

  <title>Show Building</title>

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

</head>

<body>

<h1>Building Details</h1><br />

    <table>

     <thead>

      <tr>

       <th>Sl No</th>>

       <th>Building ID</th>

       <th>Building Name</th>

       <th>Number Of Apartments</th>

       <th colspan="2">Action</th>

      </tr>

     </thead>

     <tbody>

     <?php $count = 1; ?>
```

```
     <?php while ($row = mysqli_fetch_array($results)) { ?>
    <tr>
      <td><?php echo $count++; ?></td>
      <td><?php echo $row['Build_ID']; ?></td>
      <td><?php echo $row['Build_Name']; ?></td>
      <td><?php echo $row['No_of_Apts']; ?></td>
      <td>
        <a   href="show_building.php?edit=<?php   echo   $row['Build_ID'];   ?>"
class="edit_btn" >Edit</a>
      </td>
      <td>
        <a   href="updt_building.php?del=<?php   echo   $row['Build_ID'];   ?>"
class="del_btn">Delete</a>
      </td>
    </tr>
  <?php } ?>
     </tbody>
    </table>
</body>
</html>
```

4.3 STORED PROCEDURE:

Stored procedure offer advantages over embedding queries in a graphical user interface. Since stored procedure are modular, it is easier to troubleshoot when a problem arises in an application. Stored procedures are also tunable, which eliminates the need to modify the GUI source code to improve its performance. It's easier to code stored procedures than to build a query through GUI.

Use of stored procedures can reduce network traffic between client and server, because the commands are executed as a single batch of code. This means only the call to execute call to execute the procedure is sent over a network, instead of every single line of code being sent individually.

Stored procedure in SQL server can accept input parameters and return multiple value of output parameters, in SQL server stored procedure program statements to perform operations in the database and return a status values to a calling procedure or batch.

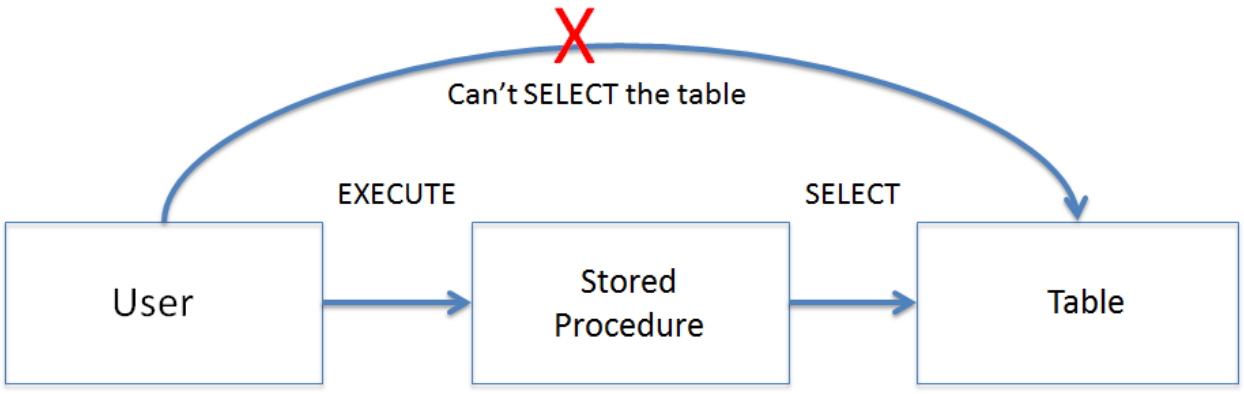


Fig 4.1: Stored Procedure

Why Stored Procedure?

Most of us are quite familiar with the normal setup to build a database application: creating database, creating tables, set up indexes, CRUD the data, issue queries from the client side and do further processing if necessary.

That workflow works fine in most cases but there is one important aspect of database programming missing: the Stored Procedure.

There are at least four advantages I can think of to use an SP in a database application.

Firstly, it reduces the network traffic and overhead. In a typical PHP database web application, there are four layers:

- The client layer, which is normally a web browser. It receives user interactions and presents the data in a UI.
- The web server layer, which handles and dispatches user requests and sends back responses to the client layer.
- The PHP layer, which handles all PHP interpretation, does the application logic and generates the PHP part of response.
- The database layer, which handles all database queries, including but not limited to a SELECT query, an INSERT statement, etc.

CREATING A STORED PROCEDURE:

SYNTAX:

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

CODE USED IN OUR PROJECT:

```
CREATE DEFINER = `root`@`localhost` PROCEDURE `getLogDetails`()
NOT DETERMINISTIC NO SQL SQL SECURITY DEFINER SELECT * FROM apt_logs;
```

CALLING A STORED PROCEDURE:

SYNTAX:

```
CALL procedure_name;
```

CODE USED IN OUR PROJECT:

```
$show = "CALL `getLogDetails`();";
```

4.4 Triggers:

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

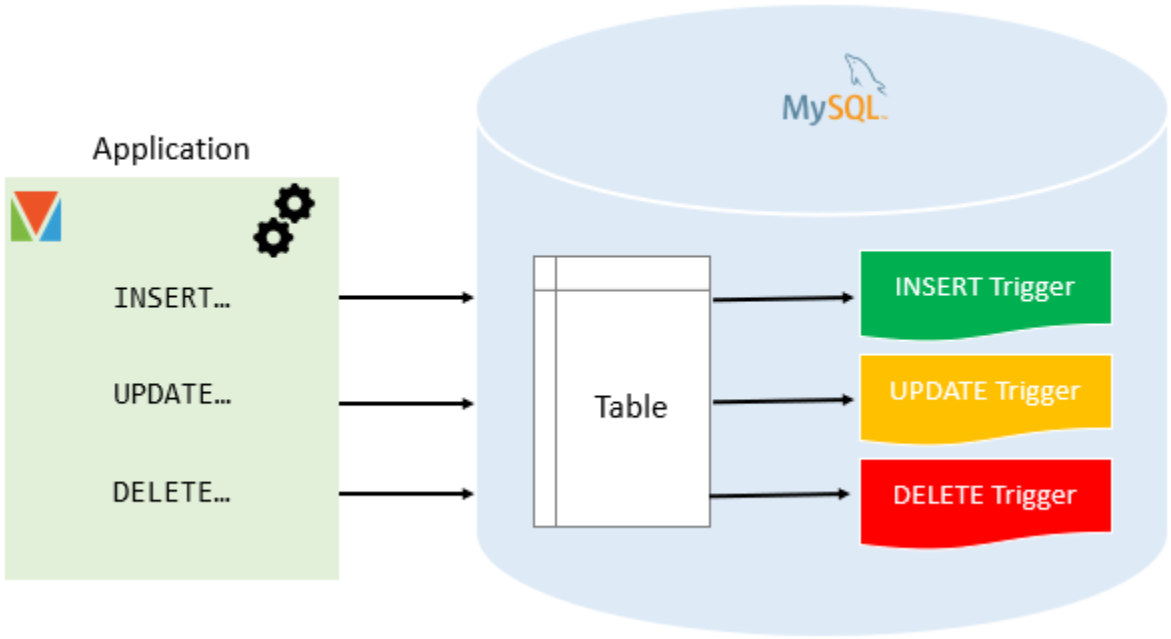


Fig 4.2: Triggers

SYNTAX:

```
create trigger [trigger_name] [before | after]
{insert | update | delete}  on [table_name]
[for each row]
[trigger_body];
```

Explanation of syntax:

- create trigger [trigger_name]: Creates or replaces an existing trigger with the trigger_name.
- [before | after]: This specifies when the trigger will be executed.
- {insert | update | delete}: This specifies the DML operation.
- on [table_name]: This specifies the name of the table associated with the trigger.
- [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.
- [trigger_body]: This provides the operation to be performed as trigger is fired.

BEFORE AND AFTER TRIGGERS:

BEFORE triggers run the trigger action before the triggering statement is run. AFTER triggers run the trigger action after the triggering statement is run.

CODE USED IN OUR PROJECT

```
CREATE DEFINER=`root`@`localhost` TRIGGER `del_trigger`
BEFORE DELETE ON
`apartment` FOR EACH ROW INSERT INTO apt_logs VALUES (null, OLD.Apt_ID, OLD.B
uild_ID, OLD.No_of_Rooms, OLD.Price, OLD.Apt_For, NOW());
```

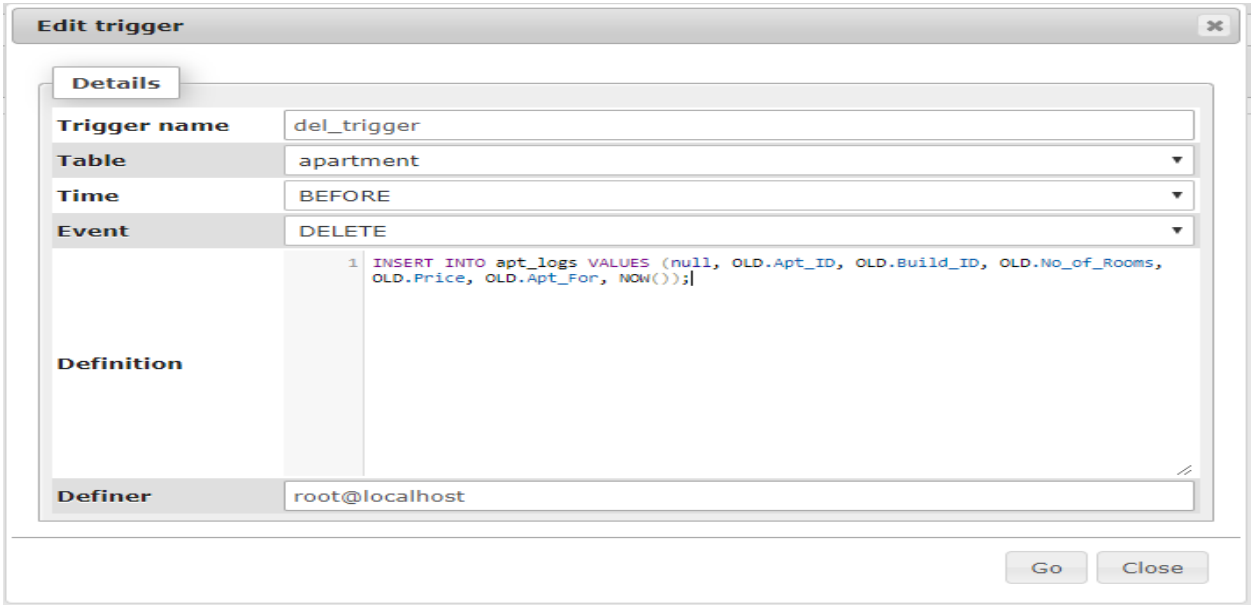


Fig 4.3: Creating triggers in XAMPP

CHAPTER 5

SNAPSHOTS:

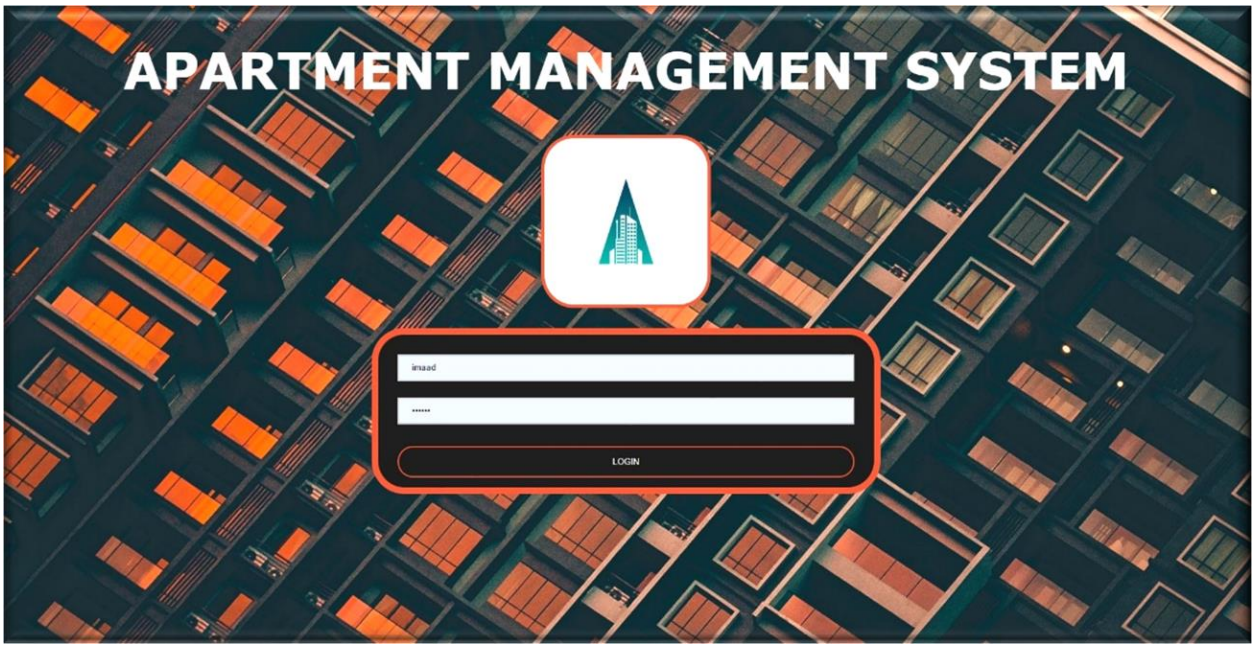


Fig 5.1: Login Page

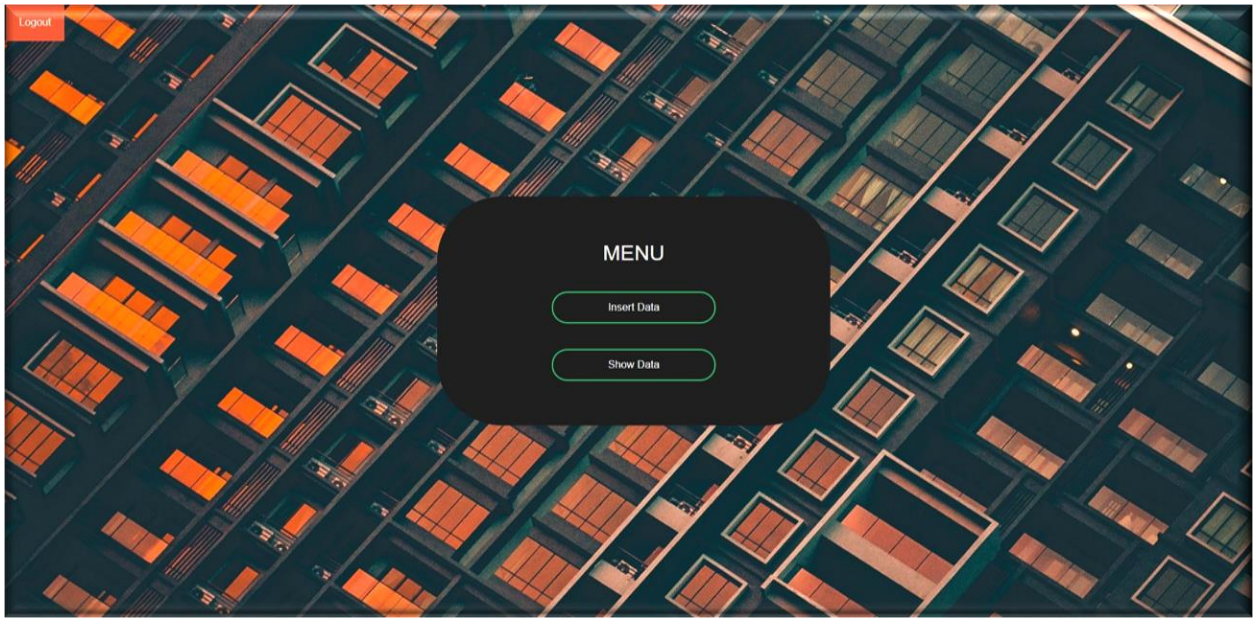


Fig 5.2: Menu Page

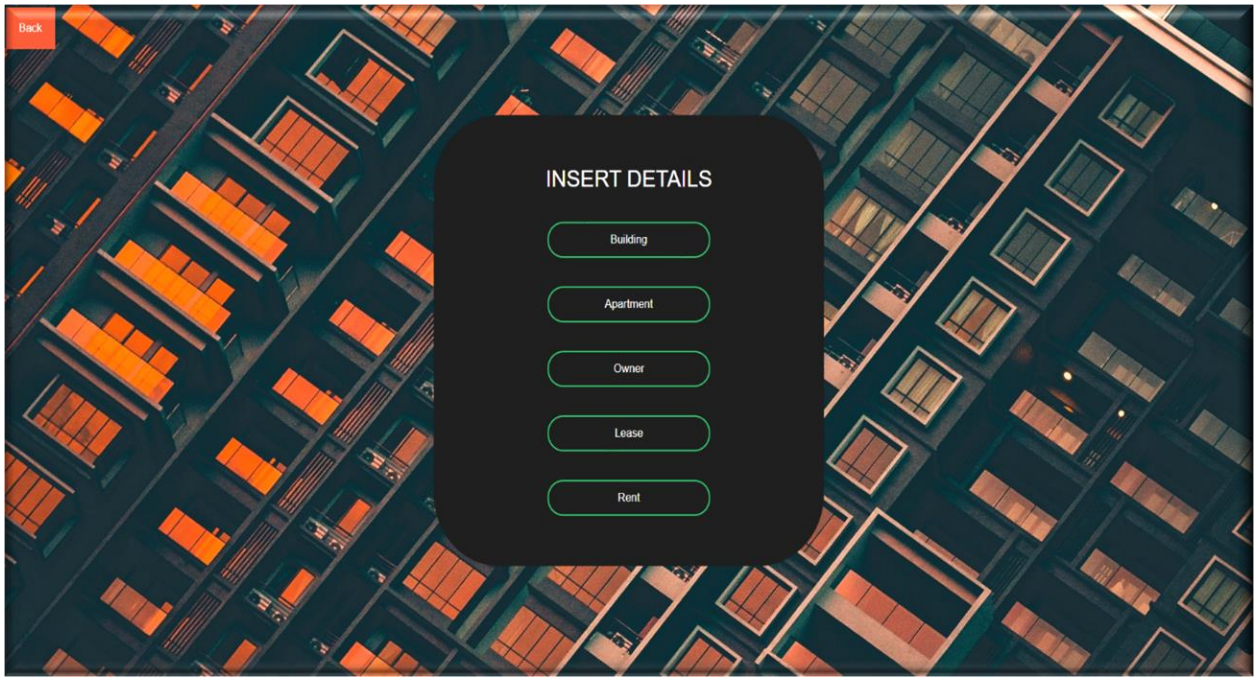


Fig 5.3: Insert

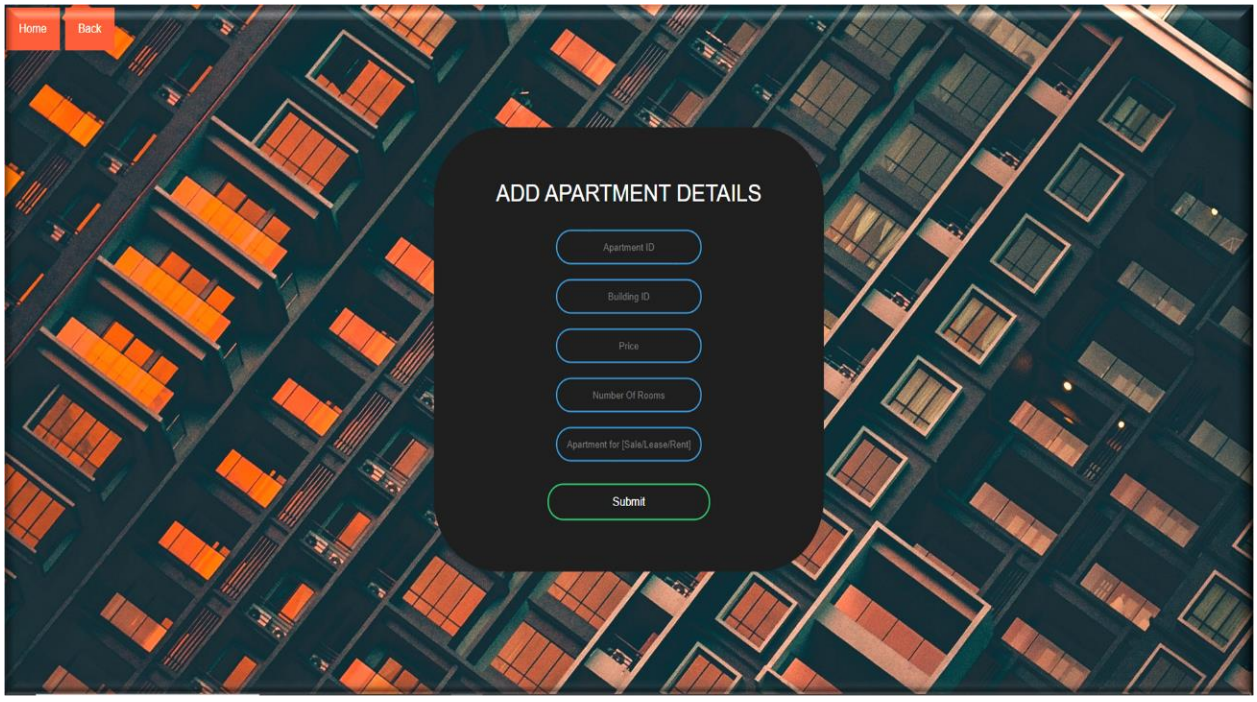


Fig 5.4: Inserting Data

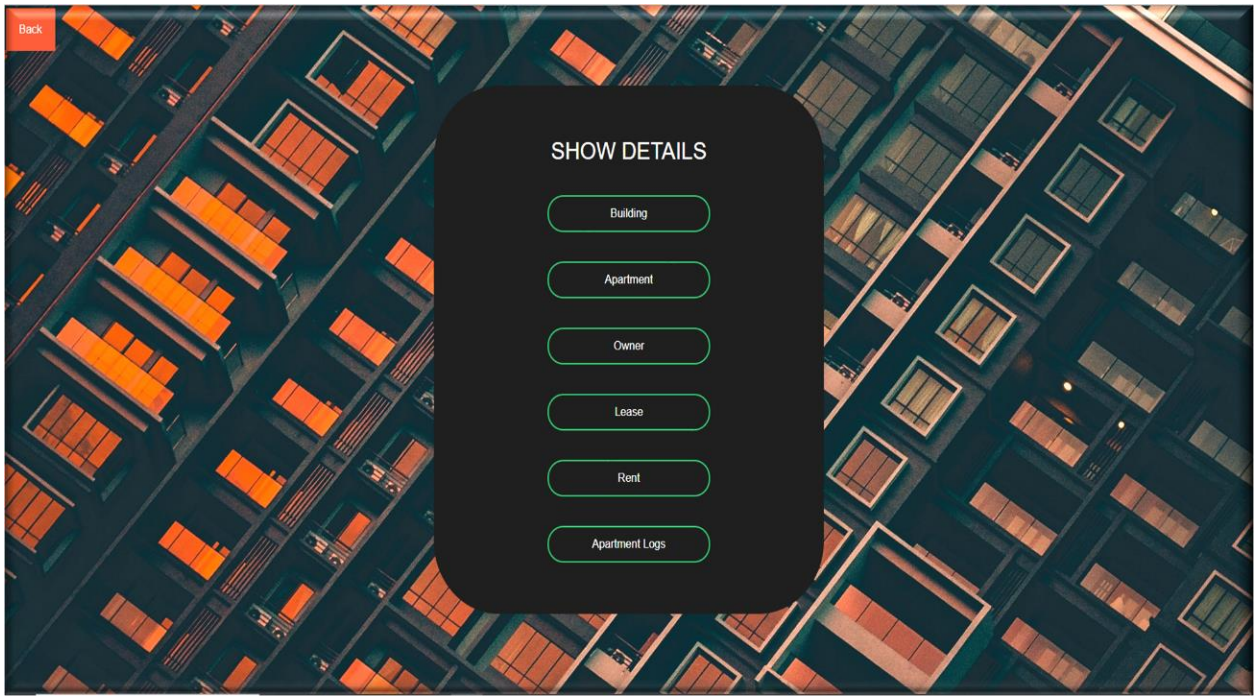


Fig 5.5: Display

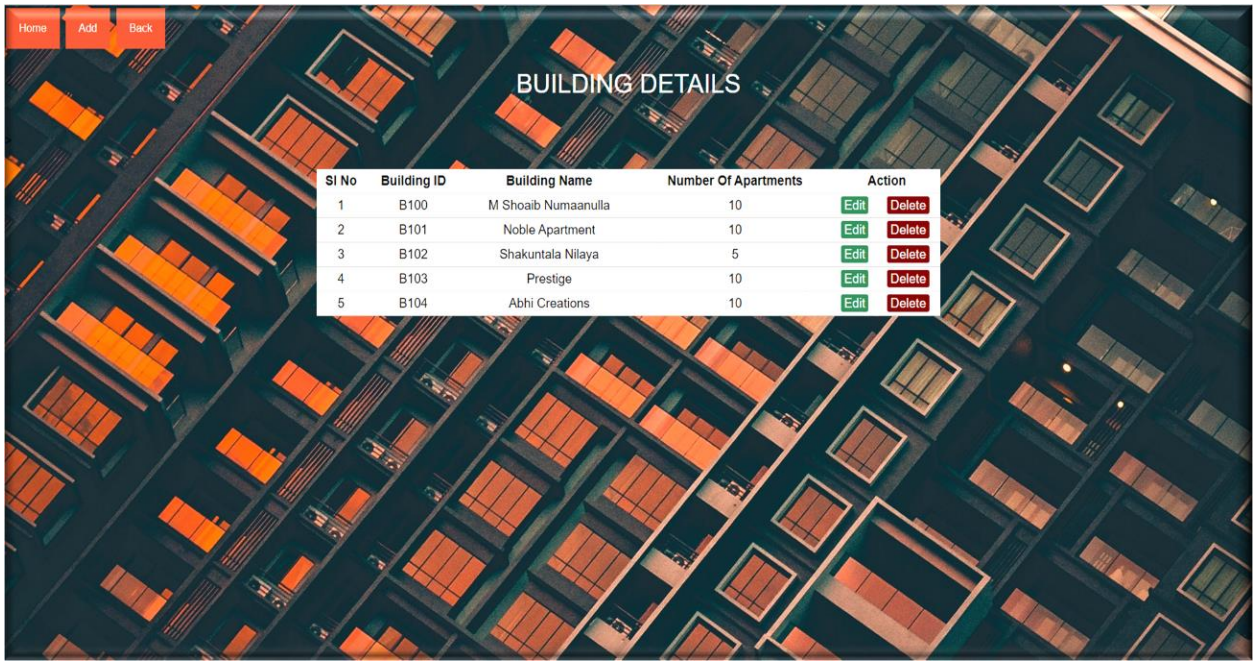


Fig 5.6: Edit/Delete

CHAPTER 6

CONCLUSION

The project apartment management system was completed successfully. The system has been developed with much care and free of errors and the same time it is efficient and less time consuming. The purpose of this project was to develop web application designing and web application for business and stock item in the shop. This project helped me in graining valuable information and practical knowledge on several topics like designing web page using html, php, javascript and management of database using mysql. The entire system is secured also the project and software development life cycle. I learnt to test different features of a project.

The project has given great satisfaction in having designed an application which can be implemented to any nearby shop selling various kinds of products by simple modification. This is scope of future development in our project to a great extent a number of features can be added in the system in future providing.

CHAPTER 7

BIBILOGRAPHY

WEBSITES:

- www.google.co.in
- www.w3school.com
- www.geeksforgeeks.org
- www.mysqltutorial.org
- www.sitepoint.com
- www.tutorialspoint.com