## CHAPTER 1

**INTRODUCTION**

### Introduction to Database Management System

A database management system (DBMS) refers to the technology for creating and managing databases. DBMS is a software tool to organize (create, retrieve, update, and manage) data in a database. A software for storing and retrieving user by considering appropriate security measures. It allows users to create their own databases as per their requirement.

It consists of group of programs which manipulate the database and provide an interface between the database. It includes the user of the database and other application programs. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

If it is any field that has contributed to the greatest advancements in the world today, it is the field of information technology, commonly known by the acronym IT. When most people hear of the phrase ‘information technology’, they tend to picture several Tecno-geeks seated by the computer and fixing some computer software.

### Types of DBMS

There are mainly 4 types of DBMS, which are Hierarchical, Relational, Network, and Object-Oriented DBMS

* **Hierarchical DBMS**: As the name suggests, this type of DBMS has a style of predecessor- successor type of relationship. So, it has a structure similar to that of a tree, wherein the nodes represent records and the branches of the tree represent fields.
* **Relational DBMS (RDBMS)**: This type of DBMS, uses a structure that allows the users to identify and access data in relation to another piece of data in the database.
* **Network DBMS**: This type of DBMS supports many to many relations wherein multiple member records can be linked.
* **Object-oriented DBMS**: This type of DBMS uses small individual software called objects. Each object contains a piece of data, and the instructions for the actions to be done with the data.

### Applications of DBMS

* + 1. **Library Management System**: There are lots of books in the library so; it is tough to store the record of all the books in a register or copy. So, the database management system (DBMS) is used to maintain all the information related to the name of the book, issue date, availability of the book, and its author.
    2. **Railway Reservation System** In the railway reservation system, the database is required to store the record or data of ticket bookings, status about train’s arrival, and departure. Also, if trains get late, people get to know it through database update.
    3. **Banking**: Database management system is used to store the transaction information of the customer in the database.
    4. **Credit card transactions:** Database Management system is used for purchasing on credit cards and generation of monthly statements.
    5. **Education Sector:** Presently, examinations are conducted online by many colleges and universities. They manage all examination data through the database management system (DBMS). In spite that student’s registrations details, grades, courses, fee, attendance, results, etc. all the information is stored in the database.
    6. **Social Media Sites:** We all use of social media websites to connect with friends and to share our views with the world. Daily, millions of peoples sign up for these social media accounts like Pinterest, Facebook, Twitter, and Google plus. By the use of the database management system, all the information of users is stored in the database and, we become able to connect with other people.
    7. **Telecommunications:** Without DBMS any telecommunication company can’t think. The Database management system is necessary for these companies to store the call details and monthly postpaid bills in the database.
    8. **Finance:** The database management system is used for storing information about sales, holding and purchases of financial instruments such as stocks and bonds in a database.
    9. **Online Shopping:** Everyone wants to shop through online shopping websites (such as Amazon, Flipkart, snap deal) from home. So, all the products are sold and added only with the help of the database management system (DBMS). Invoice bills, payments, purchase information all of these are done with the help of DBMS.
    10. **Human Resource Management:** Big firms or companies have many workers or employees working under them. They store information about employee’s salary, tax, and work with the help of database management system (DBMS).
    11. **Manufacturing:** Manufacturing companies make different types of products and sale them on a daily basis. In order to keep the information about their products like bills, purchase of the product, quantity, supply chain management, database management system (DBMS) is used.
    12. **Airline Reservation System:** This system is the same as the railway reservation system. This system also uses a database management system to store the records of flights departure, arrival, and delay status.

### Introduction to MySQL

MySQL is a Relational Database Management System (“RDBMS”). It is used by most modern websites and web-based services as a convenient and fast-access storage and retrieval solution for large volumes of data. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company.

It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

MySQL can also be accessed using many tools. It can be easily communicated with via PHP (PHP Hypertext Preprocessor), a scripting language whose primary focus is to manipulate HTML for a webpage on the server before it is delivered to a client’s machine. A user can submit queries to a database via PHP, allowing insertion, retrieval and manipulation of information into/from the database.

MySQL server design is multi-layered with independent modules and is fully multithreaded by using kernel threads. It can handle multiple CPUs if they are available.

### MySQL Command Syntax

The four main categories of SQL statements are as follows

#### DML (Data Manipulation Language)

DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

* + - * **SELECT** – select records from a table

#### SELECT column1, column2, ... FROM table\_name;

* + - * **INSERT** – insert new records

#### INSERT INTO table\_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);

* + - * **UPDATE** – update/Modify existing records

#### UPDATE table\_name

**SET column1 = value1, column2 = value2, ... WHERE condition;**

* + - * **DELETE** – delete existing records

#### DELETE FROM table\_name WHERE condition;

#### DDL (Data Definition Language)

DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

* + - * **CREATE** – create a new Table, database, schema **CREATE TABLE table\_name ( column1 datatype,**

#### column2 datatype, column3 datatype, ....

**);**

* + - * **ALTER** – alter existing table, column description

#### ALTER TABLE table\_name ADD column\_name datatype;

* + - * **DROP** – delete existing objects from database

#### DROP TABLE table\_name;

#### DCL (Data Control Language)

DCL statements control the level of access that users have on database objects.

* + - * **GRANT** – allows users to read/write on certain database objects **GRANT ALL PRIVILEGES ON database\_name.\* TO 'username'@'localhost';**
      * **REVOKE** – keeps users from read/write permission on database objects

#### REVOKE privileges ON object FROM user;

#### TCL (Transaction Control Language)

TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.

* + - * **BEGIN** Transaction – opens a transaction **[begin\_label:] BEGIN [statement\_list]**

#### END [end\_label]

* + - * **COMMIT** Transaction – commits a transaction

#### COMMIT [ {TRAN | TRANSACTION} [ transaction\_name | @tran\_name\_variable]] [ WITH (DELAYED\_DURABILITY = {OFF | ON})]

**[ ; ]**

* + - * **ROLLBACK** Transaction – ROLLBACK a transaction in case of any error

#### ROLLBACK {TRAN | TRANSACTION}

**[ transaction\_name | @tran\_name\_variable**

**| savepoint\_name | @savepoint\_variable] [;]**

## 

## CHAPTER 2

**ANALYSIS AND REQUIREMENT SPECIFICATION**

### Purpose of this project

### The Preowned Vehicle Showroom Database Management System aims to revolutionize the way users buy and manage preowned vehicles. By leveraging programming and database techniques, this system simplifies the process of managing preowned vehicle inventory, customer interactions, and sales transactions. Through an online platform, users can easily browse through available vehicles, inquire about specific models, and make purchases from anywhere at any time. This not only saves the user's time but also enhances their overall experience by providing convenient access to preowned vehicle information. The system's goal is to provide a user-friendly and efficient platform for accessing information about preowned vehicles, ensuring seamless operations and enhancing customer satisfaction in the preowned vehicle market.

### Scope of this project

* Simplify the process of browsing and purchasing preowned vehicles through an intuitive online platform.
* Enhance efficiency in managing preowned vehicle inventory and streamline the process of vehicle acquisition.
* Offer access to preowned vehicle information in various formats, including PDFs, allowing users to read anywhere and at any time.
* Provide email communication functionality for future correspondence, facilitating seamless interaction between users and the showroom for inquiries, updates, and support.

### Functional Requirements

#### Modules

* Login Module: This module manages user authentication and login details. Users can access the system by providing their email and password.
* Inventory Module: This module handles the management of vehicle inventory. It provides services related to adding, updating, and deleting vehicle details, as well as viewing available vehicles.
* Booking Module: Used for managing vehicle bookings made by customers. It allows users to record and track bookings, including the vehicle ID, customer ID, and booking date.
* Sales Module: This module tracks sales transactions made by managers. It records details such as the vehicle sold, customer ID, sale date, and sale price.
* Customer Management Module: Used for managing customer information. It includes functionalities for adding, updating, and deleting customer details, such as username, password, email, full name, and phone number.

### Non-Functional Requirements

#### Hardware requirements

* + - * **Processor** – Intel Core i3, 2Ghz or above for PCs and 1Ghz
      * **Memory Space** – 1GB recommended.
      * **RAM** – 2GB or above.

#### Software requirements

* + - * **Operating System** – Windows 7 and above, MacOS X and above
      * **Browser** – Google Chrome, Microsoft Edge or any other basic browsers.
      * **Front End** – HTML, CSS, JS
      * **Back End** – MySQL
      * **Server-side Scripting** – PHP

## CHAPTER 3

**DESIGN**

### Entity Relationship Diagram

Data schema in graphical form is called ER Diagram. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities. An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business. An ER model does not define the business processes; it only presents a business.

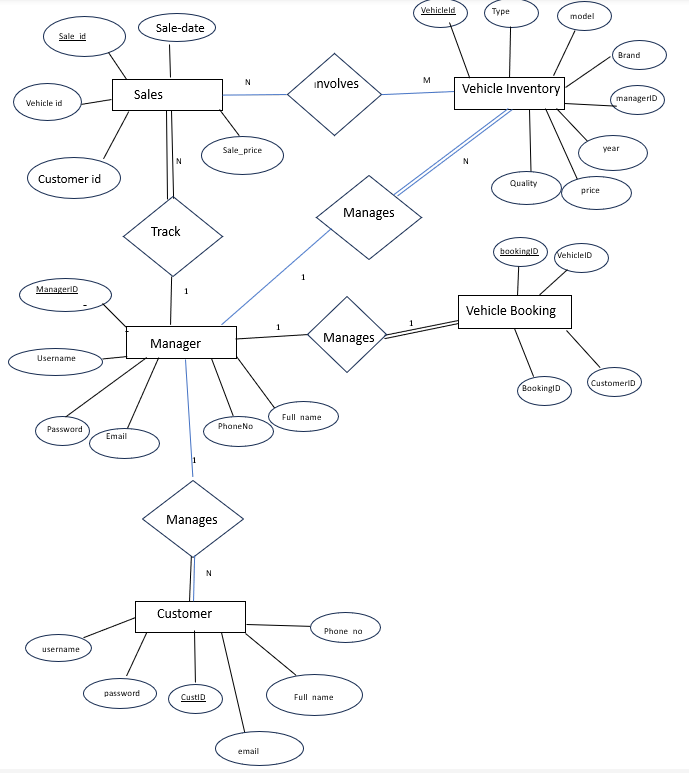


Fig 3.1 ER diagram for Showroom database

### Schema Diagram

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database. A relational schema shows references among fields in the database. When a primary key is referenced in another table in the database, it is called a foreign key. This is denoted by an arrow with the head pointing at the referenced key attribute. A schema diagram helps organize values in the database. The following diagram shows the schema diagram for the database.

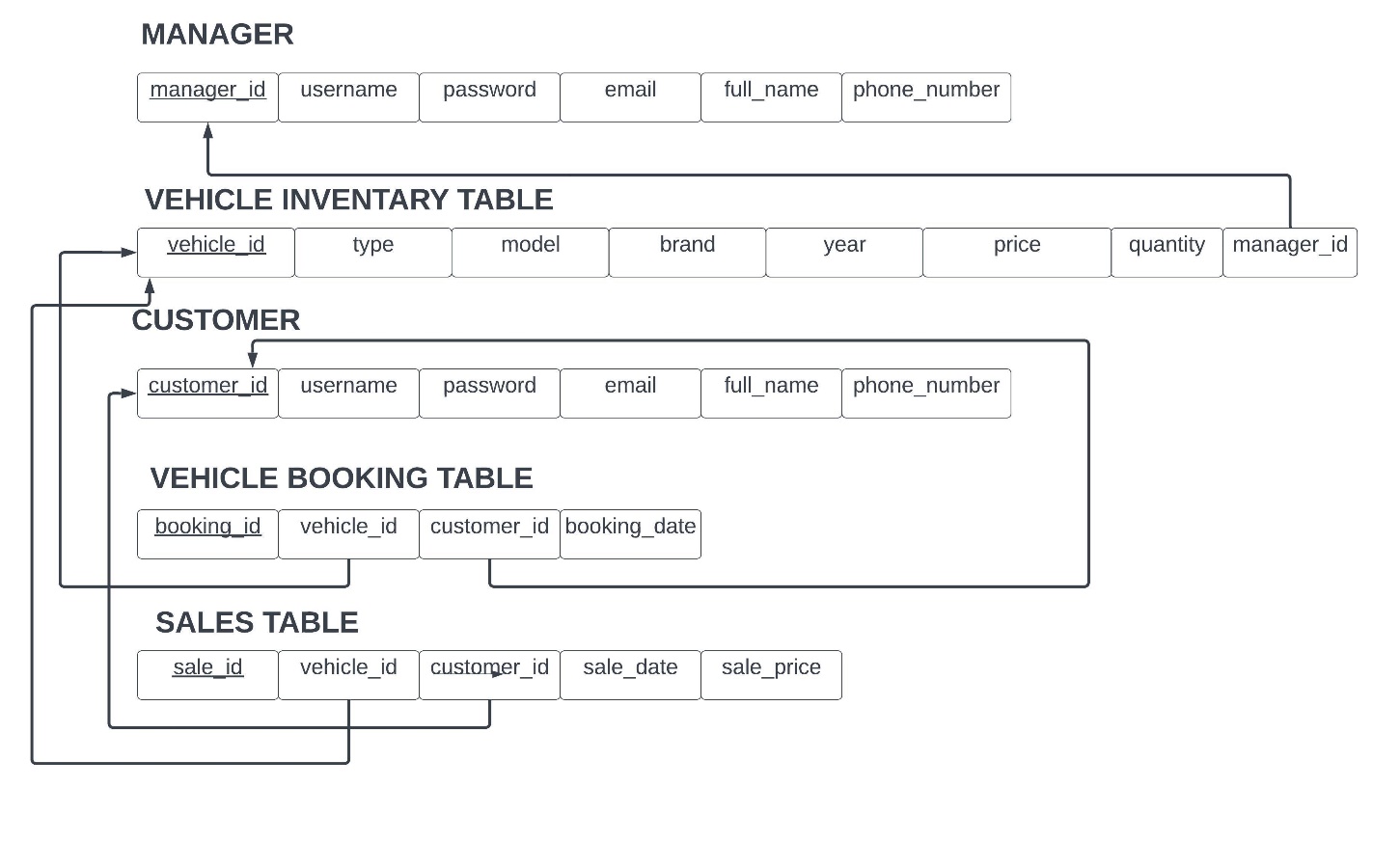
****

Fig 3.2 Schema diagram of Showroom database

## CHAPTER 4

**IMPLEMENTATION**

### Implementation of Table Creation page

* Manager Table: Stores information about managers who oversee the vehicle inventory.

CREATE TABLE Manager (

Manager\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Username VARCHAR(50) UNIQUE NOT NULL,

Password VARCHAR(255) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

Full\_Name VARCHAR(100) NOT NULL,

Phone\_Number VARCHAR(20) NOT NULL

);

* Vehicle Inventory Table: Contains details of all vehicles available in the inventory.

CREATE TABLE VehicleInventory (

Vehicle\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Type ENUM('Car', 'Bike') NOT NULL,

Model VARCHAR(100) NOT NULL,

Brand VARCHAR(100) NOT NULL,

Year INT NOT NULL,

Price DECIMAL(10, 2) NOT NULL,

Quantity INT NOT NULL,

Manager\_ID INT,

FOREIGN KEY (Manager\_ID) REFERENCES Manager(Manager\_ID)

);

* Customer Table: Stores information about customers who interact with the system.

CREATE TABLE Customer (

Customer\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Username VARCHAR(50) UNIQUE NOT NULL,

Password VARCHAR(255) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

Full\_Name VARCHAR(100) NOT NULL,

Phone\_Number VARCHAR(20) NOT NULL

);

* Vehicle Booking Table: Records bookings made by customers for vehicles.

CREATE TABLE VehicleBooking (

Booking\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Vehicle\_ID INT,

Customer\_ID INT,

Booking\_Date DATE NOT NULL,

FOREIGN KEY (Vehicle\_ID) REFERENCES VehicleInventory(Vehicle\_ID),

FOREIGN KEY (Customer\_ID) REFERENCES Customer(Customer\_ID)

);

* Sales Table: Tracks sales made by managers.

CREATE TABLE Sales (

Sale\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Vehicle\_ID INT,

Customer\_ID INT,

Sale\_Date DATE NOT NULL,

Sale\_Price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (Vehicle\_ID) REFERENCES VehicleInventory(Vehicle\_ID),

FOREIGN KEY (Customer\_ID) REFERENCES Customer(Customer\_ID)

);

**4.2 Implementation of Connection:**

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "showroom";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

?>

**4.3 Implementation of Updation**

<?php

include("database.php");

// Check if form is submitted

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Get values from the form

$vehicleID = $\_POST["VehicleID"];

$type = $\_POST["Type"];

$model = $\_POST["Model"];

$brand = $\_POST["Brand"];

$year = $\_POST["Year"];

$price = $\_POST["Price"];

$quantity = $\_POST["Quantity"];

$managerID = $\_POST["ManagerID"];

// Update query

$sql = "UPDATE VehicleInventory SET Type = '$type', Model = '$model', Brand = '$brand', Year = $year, Price = $price, Quantity = $quantity, Manager\_ID = $managerID WHERE Vehicle\_ID = $vehicleID";

// Execute the query

try{

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

echo "Record updated successfully";

} }catch(mysqli\_sql\_exception) {

echo "Error updating record: " . $conn->error;

}

// Close the database connection

$conn->close();

}

?>

**4.4 Implementation of Insertion:**

// Check if the form is submitted

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Collect form data

$type = $\_POST["Type"];

$model = $\_POST["Model"];

$brand = $\_POST["Brand"];

$year = $\_POST["Year"];

$price = $\_POST["Price"];

$quantity = $\_POST["Quantity"];

$managerID = $\_POST["ManagerID"];

// SQL to insert new record

$insertSQL = "INSERT INTO VehicleInventory (Type, Model, Brand, Year, Price, Quantity, Manager\_ID)

VALUES ('$type', '$model', '$brand', $year, $price, $quantity, $managerID)";

// Execute the query

try{

if ($conn->query($insertSQL) === TRUE) {

echo "New record has been inserted successfully.";

} }catch(mysqli\_sql\_exception) {

echo "Error inserting record: " . $conn->error;

}

}

// Close the database connection

$conn->close();

?>

**4.5 Implementation of Search Module**

<script>

function searchVehicle() {

var input, filter, table, tr, td, i, txtValue;

input = document.getElementById("vehicleSearchInput");

filter = input.value.toUpperCase();

table = document.querySelector("table");

tr = table.getElementsByTagName("tr");

for (i = 1; i < tr.length; i++) {

td = tr[i].getElementsByTagName("td")[2];

if (td) {

txtValue = td.textContent || td.innerText;

if (txtValue.toUpperCase().indexOf(filter) > -1) {

tr[i].style.display = "";

} else {

tr[i].style.display = "none";

}

}

}

}

function insertVehicle() {

// Redirect to the insert page

window.location.href = "insert\_vehicle.php";

}

</script>

**4.6 Implementation of Deletion**

<?php

include("database.php"); // Assuming you have a database connection file

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Check if Vehicle\_ID is provided

if (isset($\_POST["Vehicle\_ID"]) && !empty($\_POST["Vehicle\_ID"])) {

$vehicleID = $\_POST["Vehicle\_ID"];

// SQL to delete the record based on Vehicle\_ID

$deleteSQL = "DELETE FROM VehicleInventory WHERE Vehicle\_ID = $vehicleID";

// Execute the query

if ($conn->query($deleteSQL) === TRUE) {

echo "Record with Vehicle\_ID $vehicleID has been deleted successfully.";

} else {

echo "Error deleting record: " . $conn->error;

}

} else {

echo "Vehicle\_ID is required.";

}

}

// Close the database connection

$conn->close();

?>

## CHAPTER 5

**SNAPSHOTS**

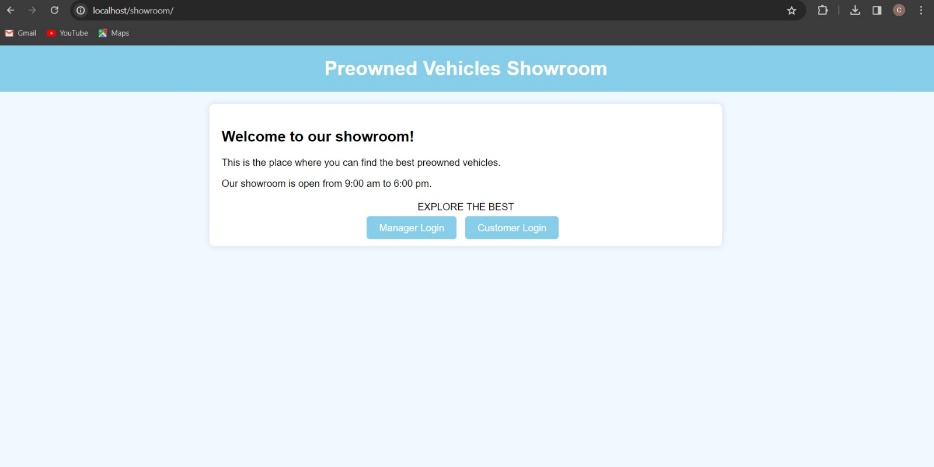


Fig 5.1 Home Page

Home page Which has opening and closing time of the Showroom along with that it also has the login option for customer and manager

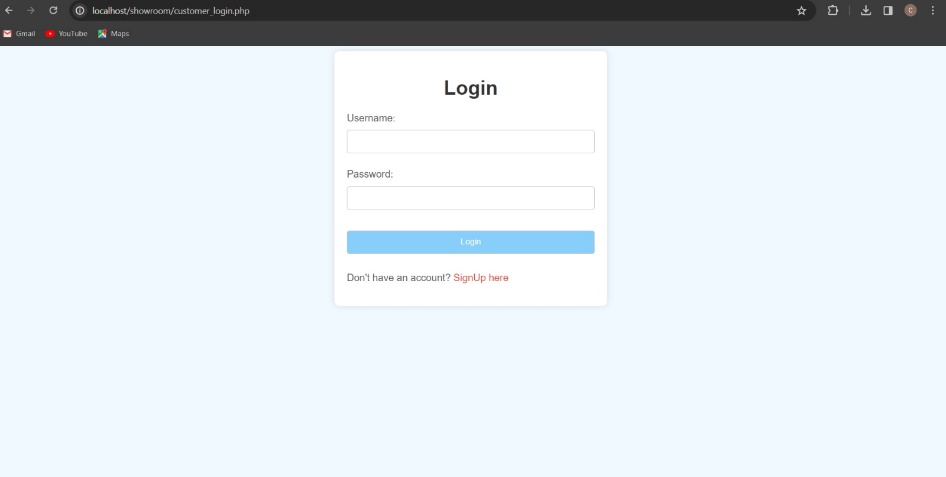


Fig 5.2 Customer Login Page

Customer Login Page : here We can Login Using Our Login Credentials

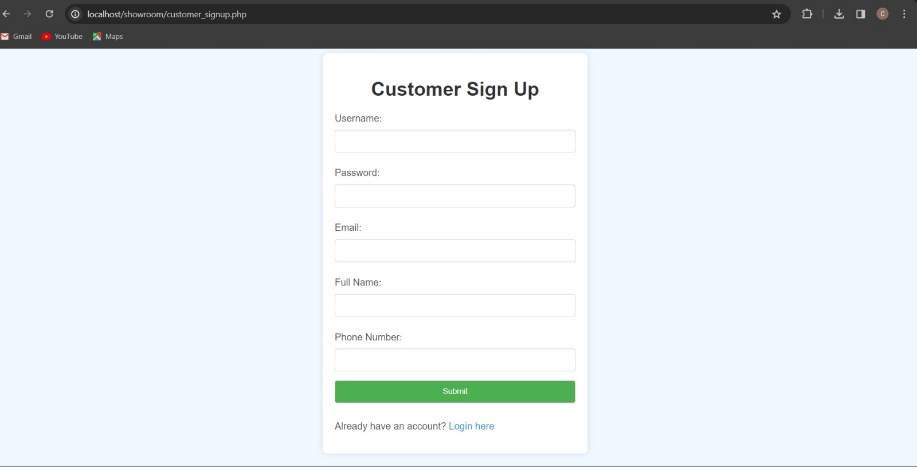


Fig 5.3 Customer Signup page

Customer Signup Page: If you are new then you should signup first and then they can Login

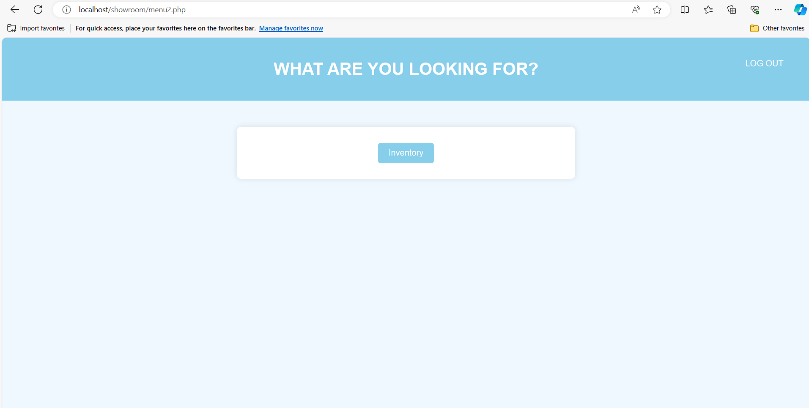


Fig 5.4 Menu for Customer

Menu Page for Customer: After Login, Above page will be displayed to the Customer. In that they can look for Inventory Details and They Can Book Any vehicle with the help of Manager from the inventory.

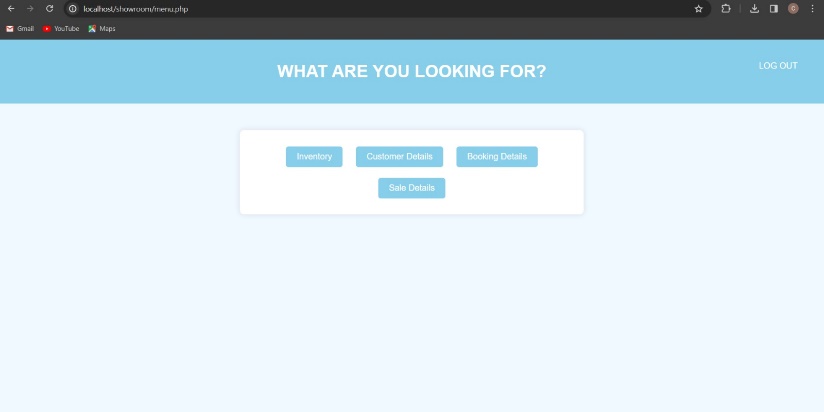


Fig 5.5 Menu for Manager

Menu Page for Manager: After Login, Above page will be displayed to the Manager. In that they can look for Inventory Details, Customer Details, booking Details, Sale Details etc.

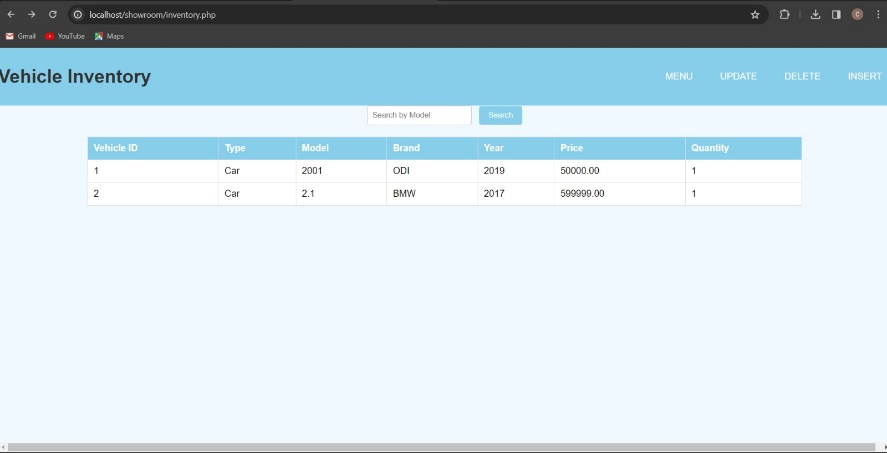


Fig 5.6 Retrieving Data of Inventory Table

Retrieving Data of Inventory Table: Manager can see all retrieved data and also Have search Button if many Entries are there.

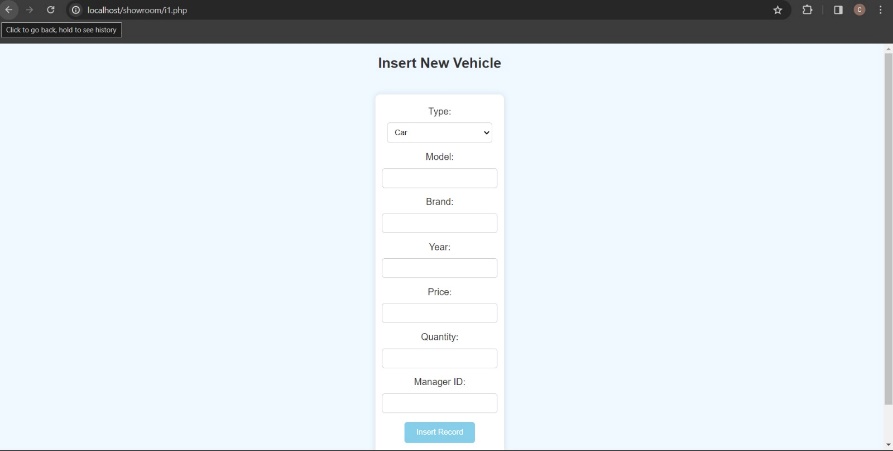


Fig 5.7 Inserting Data to the Table

Inserting Data to the Table: We can insert The Data To the table. Tables are: Sales, Inventory, Vehicle Booking

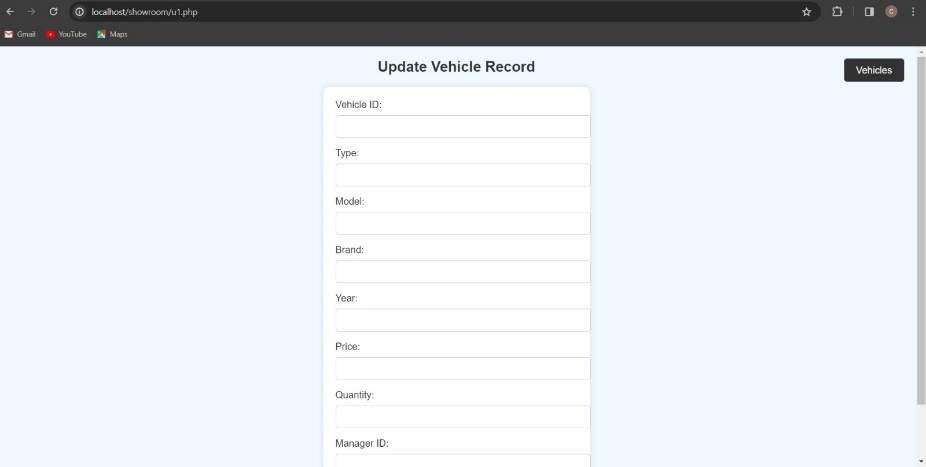


Fig 5.8 Updating the contents of table

Updating Data to the Table: We can update The Data To the table. Tables are: Sales, Inventory, Vehicle Booking

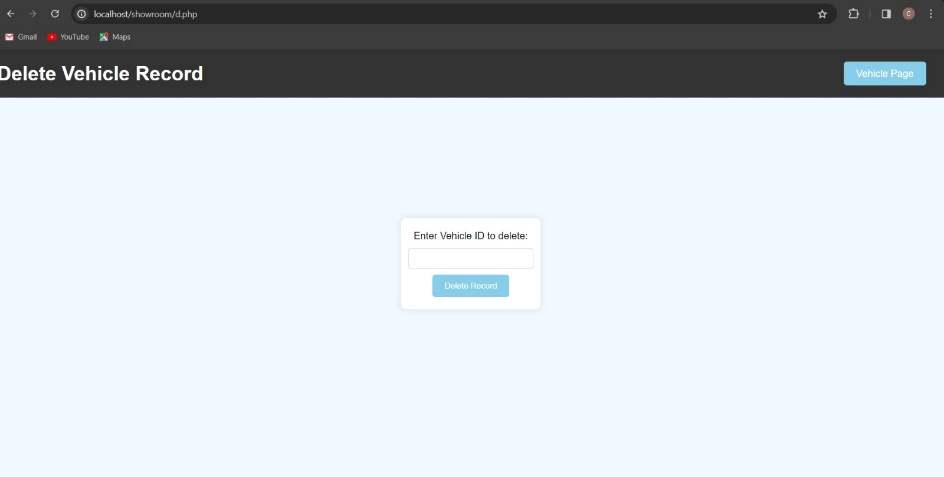


Fig 5.9 Deleting Contents from the Table

Deleting Data from Table

## CHAPTER 6

**CONCLUSION**

We had a wonderful experience developing this project. By working on this project, we gained hands-on experience of using the software using which we experimented this project. We have made use of hardware devices such as mouse and keyboard driven interface, thus to reduce the complexity and make it user friendly. The project helped in understanding the working of computer software using XAMPP and various concepts, functions and methodologies for the development of an ordering website. We hope our project will serve its purpose without any hassles.

## REFERENCES

1. Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill.