

BANGALORE UNIVERSITY
UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

K R Circle, Bengaluru - 560001



Department of Computer Science and Engineering

A DBMS Mini-Project Report on

“CAR RENTAL SYSTEM”

Submitted By:

Kavya S R (18GAEC9028)
V SEM, B.Tech (CSE)

Under the Guidance of:

Dr. Champa. H.N
Professor
Department of CSE
UVCE, Bengaluru

YEAR 2020-21

BANGALORE UNIVERSITY
University Visvesvaraya College of Engineering

K R Circle, Bengaluru - 560001



Department of Computer Science and Engineering

CERTIFICATE

This is to certify that Mrs. Kavya S R of V Semester, B. Tech, (Computer Science and Engineering), bearing the register number 18GAEC09028 has submitted the DBMS Mini-Project Report on “**CAR RENTAL SYSTEM**”, in partial fulfillment for the DBMS Lab, prescribed by the Bangalore University for the academic year 2020-21.

Guide:

Dr. Champa H N
Professor
Department of CSE
UVCE

Chairperson:

Dr. Dilip Kumar S M
Professor & Chairperson
Department of CSE
UVCE

Examiner 1: _____

Examiner 2: _____

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task is incomplete without the mention of people who made it possible, whose constant guidance and encouragement crown all efforts and success.

I am grateful to acknowledge the honorable Vice Chancellor **Dr. Venugopal K R**, Bangalore University, for his valuable suggestions and relentless support, which has sustained me throughout the course of the project.

I express my gratefulness to **Dr. H N Ramesh**, Principal, UVCE who has been leading our college with a brighter vision in technical education.

I express my profound gratitude to **Dr. Dilip Kumar S M**, Chairperson, Department of Computer Science and Engineering, UVCE who has been a constant source of inspiration to the students work.

With extreme honor I express my deep sense of gratitude to my guide **Dr. Champa H N**, Professor, Department of Computer Science and Engineering, UVCE for her valuable guidance and supervision in this course of project.

I express my sincere thanks to all teaching and non-teaching staff, Department of Computer Science and Engineering, UVCE for all the facilities that they provided me to successfully complete this project.

I also thank my parents and friends for their continuous support and encouragement.

-Kavya S R

18GAEC9028

ABSTRACT

The project 'Car rental System' deals with the maintenance of large number of records of the different types of cars to be maintained by its user.

This is a website designed primarily for use of managing the car records. This system deals with the maintenance of the records of the different categories of car and their users. It also contains the necessary information of booking, cost and car-history details and their services.

The proposed system makes it easier to search these records and edit them. This system has a very user-friendly interface and all the operations that can be performed in the project are self-explanatory. It reduces the effort required to manually maintain all these records.

TABLE OF CONTENTS

SL.NO	TITLE	PAGE.NO
1	Introduction	1
2	Literature Survey	2
3	Software Environment	3
4	Requirement Specification	8
5	System Design	9
6	Implementation	22
7	Results	24
8	Conclusion and Future Work	27
9	References	28

LIST OF FIGURES

Figure 5.1: Notation for ER Diagram	10
Figure 5.2: ER Diagram	11
Figure 5.3: Relational Database Schema Diagram	12
Figure 6.1: Connection of Signup page with backend	22
Figure 6.2: Connection of Booking page with backend	23
Figure 6.3: Retrieving data from backend using various PHP functions	23
Figure 7.1: Welcome page	24
Figure 7.2: Login and signup form	24
Figure7.3: Car details	25
Figure 7.4: Booking and payment form	25
Figure 7.5: Customer reports	26
Figure 7.6: Admin page	26

CHAPTER 1

INTRODUCTION

The project 'Car Rental System' deals with the maintenance of large number of records of the different types of cars to be maintained by its admin. The proposed system makes it easier to edit records. This system has a very user-friendly interface and all the operations that can be performed in the project are self-explanatory. It reduces the effort required to manually maintain all these records.

This system is a website based system which fulfills the requirement of a typical management of details of cars, users, expenses and reports in a company along with booking option. It provides the Management Reports like Car Booking details and status etc.

The customer has to login with his respective credentials to view the details of cars he possesses, car-history, booking, expenses and fuel details. A new user can make use of this Car Rental System by signing up and providing the required details.

Login module contains the particular user login and username.

In car module, all the cars are listed. They can be of different types cars. We can specify the details of cars like model, cost per day, no_of_seats etc...

Booking module will provide you details such as date, time, location, etc...

Admin module contains admin name and password he has privilege to add or remove vehicles and he can view customer and booking details.

Payment module contains information related to payments. It will provide details such as paid amount date etc.

CHAPTER 2

LITERATURE SURVEY

Car Rental System is a website used to store data like Customer, Cars, Booking, Payment details which facilitates easy storage and retrieval of information.

Previous System Model:

In exiting system user (or) client will directly interact with the car owner and owner will decide whether the car is available or not. Then if it is available, he will give rent a car to the customer. User should manually go and book the car. Its time taking and expensive process.

Proposed System Model:

In this car rental system, we are going to introduce online booking of car renting process available. So, the burden of the customer will be reduced. Our aim is to design and create a data management system for a car rental company. This enables admin can rent a vehicle that can be used by a customer. By paying the money for specified period of time. This system increases customer retention and simplifies vehicle and staff management in an efficient way. This software car rental system has a very user-friendly interface. Thus, the users will feel very easy to use it. By using this system admin can manage their rental, payment, employment issues and vehicle issues such as insurance. The car information can be added to the system by admin.

CHAPTER 3

SOFTWARE ENVIRONMENT

3.1 Database Management System (DBMS)

DBMS is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications.

A Relational **database** is a database that has a collection of tables of data items, all of which is formally described and organized according to the relational model. Data in a single table represents a relation, from which the name of the database type comes. In typical solutions, tables may have additionally defined relationships with each other. In the relational model, each table schema must identify a column or group of columns, called the primary key, to uniquely identify each row. A relationship can then be established between each row in the table and a row in another table by creating a foreign key, a column or group of columns in one table that points to the primary key of another table.

3.1.1 Characteristics of Database Management Systems

- Self-describing nature.
- Keeps a tight control on data redundancy.
- Enforces user defined rules to ensure that integrity of table data.
- Provides insulation between Programs and data, Data abstraction.
- Supports multiple views of the data.
- Helps sharing of data and Multi-user transaction processing.

3.1.2 Advantages of using the DBMS approach

- Controlling the redundancy.
- Restricting unauthorized access.
- Providing persistent storage for program objects.

- Providing storage structures for efficient query processing.

3.2 MYSQL

MySQL is an Oracle-backed 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 applications and online publishing.

MySQL 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 has the relational database management system and PHP as the object-oriented scripting language. (Sometimes Perl or Python is used instead of PHP.)

Originally conceived by the Swedish company MySQL AB, MySQL was acquired by Sun Microsystems in 2008 and then by Oracle when it bought Sun in 2010. Developers can use MySQL under the GNU General Public License (GPL), but enterprises must obtain a commercial license from Oracle.

Today, MySQL is the RDBMS behind many of the top websites in the world and countless corporate and consumer-facing web-based applications, including Facebook, Twitter and YouTube.

SQL uses the terms table, row, and column for relation, tuple, and attribute, respectively. The SQL commands for data definition are CREATE, ALTER and DROP.

- **CREATE**

This command is used to create table or view by giving it a name and specifying its attributes and constraints. The attributes are specified first, and each attribute is given a name, a data type to specify its domain values, and any attribute constraints such as NOT NULL.

SYNTAX: CREATE TABLE <TNAME> (ATR1 TYP1 CONST1, ATR2 TYP2 CONST,...)

- **ALTER**

The definition of a base table can be altered by ALTER command which is a Schema Evolution command. The possible ALTER TABLE include adding or dropping a column (attribute), changing a column definition, and adding or dropping table constraints.

Example: ALTER TABLE STUDENT ADD NAME VARCHAR (12)

- **DROP**

If a whole schema is not needed any more, the DROP SCHEMA command can be used. There are two drop behaviour options: CASCADE and RESTRICT. CASCADE option is used to remove the database schema and all its tables, domains and other elements.

If the RESTRICT option is chosen in place of CASCADE, the schema is dropped only if it has no elements in it; otherwise, the DROP command will not be executed.

SYNTAX: DROP TABLE STUDENT CASCADE

3.2.1 Statements in SQL

Following are the important statements used in SQL.

- **SELECT** - Used to retrieve the information from the relation.
- **INSERT** - Used to insert the new values to the relation.
- **DELETE** - Used to delete one or more existing tuples from the relation.
- **UPDATE** - Used to update already existing values in the relation.

3.2.2 Aggregate Functions in SQL

Following aggregate functions are provided by the SQL.

- **COUNT** - Returns number of tuples.
- **SUM** - Returns sum of entries in a column.
- **MAX** - Returns Maximum value from an entire column.
- **MIN** - Returns Minimum value from an entire column.
- **AVG** - Returns Average of all the entries in a column.

3.2.3 Constraints in SQL

Following constraints are provided by the SQL.

- NOT NULL - Column should contain some value.
- PRIMARY KEY - Should not allow duplicate and null values to a column.
- UNIQUE - Each value of a column should be unique.

3.3 PHP

PHP: Hypertext Pre-processor (or simply **PHP**) is a general-purpose programming language originally designed for web development. It was originally created by Rasmus Lerdorf in 1994 the PHP reference implementation is now produced by The PHP Group.

PHP code may be executed with a command line interface (CLI), embedded into HTML code, or 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 a web server or as a Common Gateway Interface (CGI) executable.

The web server outputs the results of the interpreted and executed PHP code, which may be any type of data, such as generated HTML code or binary image data. PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control.

The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

The PHP language evolved without a written formal specification or standard until 2014, with the original implementation acting as the de facto standard which other implementations aimed to follow. Since 2014, work has gone on to create a formal PHP specification.

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server

- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

CHAPTER 4

REQUIREMENT SPECIFICATION

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behavior of the system.

For the successful, efficient and problem free designing any project or program, the system should meet some requirements.

4.1 Hardware Requirements

- **Processor:** Intel core Duo 2.0GHz or more.
- **RAM:** 1GB or more.
- **Hard disk:** 80GB or more
- **Monitor:** 15” CRT or LCD monitor
- **Keyboard:** Normal or Multimedia
- **Mouse:** Compatible mouse

4.2 Software Requirements

- **Operating System:** Platform independent. Windows used.
- **Software:** Xampp (for apache server)
- **Database:** MariaDB.
- **Front-End:** HTML, CSS, JS.
 - CSS and JS to create attractive layouts and animations.
 - Bootstrap for Navbar and for icons.
- **Back-end:** PHP.

PHP is used to validate the data coming through forms and for server-side scripting.

CHAPTER 5

SYSTEM DESIGN

5.1 ENTITY RELATIONSHIP DIAGRAM:

An Entity relationship diagram shows the relationships of entity sets stored in databases. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

An entity relationship diagram is a snapshot of data structure. An entity relationship diagram shows entities (tables) in a database and relationships between tables within that database. For a good database design, it is essential to have an entity relationship diagram.

There are three basic elements in entity relationship diagram

- Entities are the things for which we want to store information. An entity is a person, place, thing or event.
- Attributes are the data we want to collect for an entity.
- Relationships describe the relations between the entities.

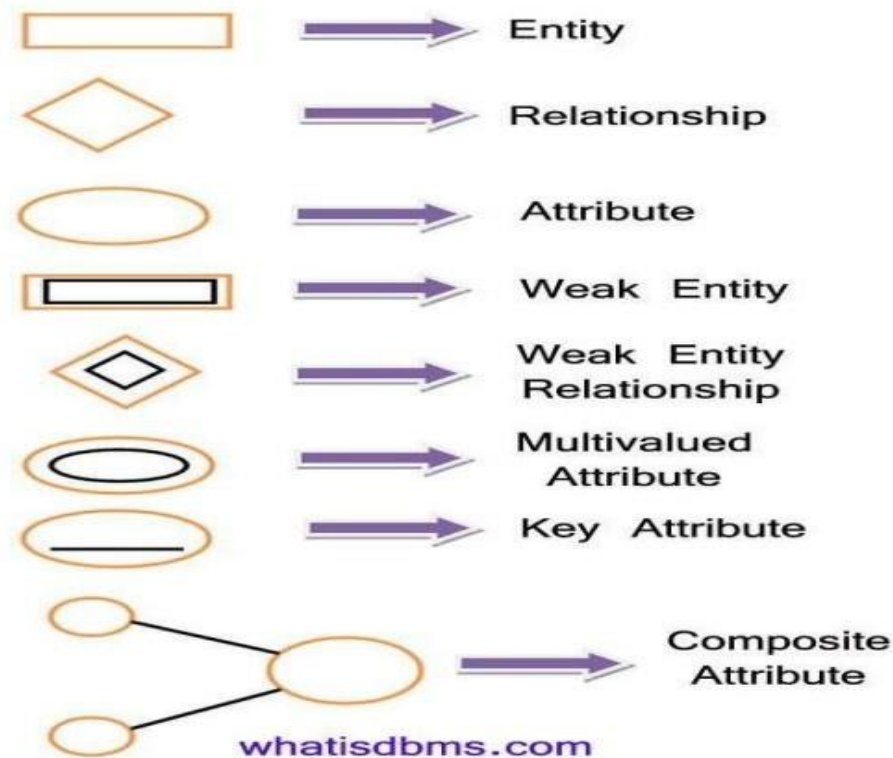
An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An entity-relationship diagram is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an entity-relationship diagram are:

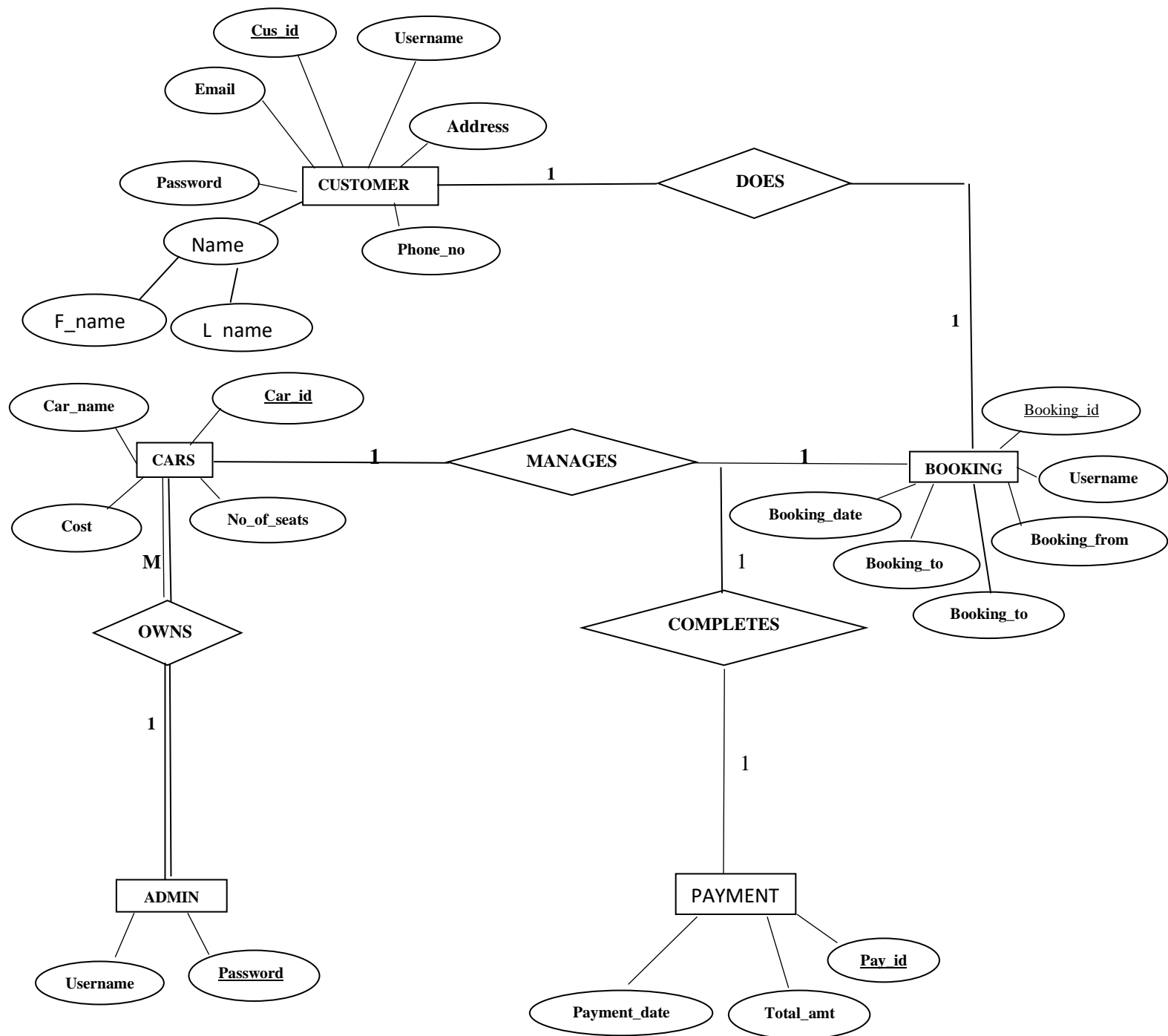
- Entities
- Relationships
- Attributes

Steps involved in creating an entity-relationship diagram include:

- Identifying and defining the entities
- Determining all interactions between the entities
- Analyzing the nature of interactions/determining the cardinality of the relationship
- Creating the entity-relationship diagram

Notations for ER diagram:**Fig.5.1 Notations for ER Diagram****Entities and their Attributes:**

- **Customer entity:** Attributes of customer are cus_id, Username, Fname, Lname, Phone_no, Email, Password, Address
- **Car entity:** Attributes of car are car_id, car_name, cost, no_of_seats
- **Booking entity:** Attributes of booking are booking_id, booking_from, booking_to, from_date, to_date, username
- **Payment entity:** Attributes of payment are pay_id, payment_date, total_amt
- **Admin entity:** Attributes of admin are username, password

ER DIAGRAM**Fig:5.2 ER Diagram for Vehicle Management system**

5.2 RELATIONAL DATABASE SCHEMA

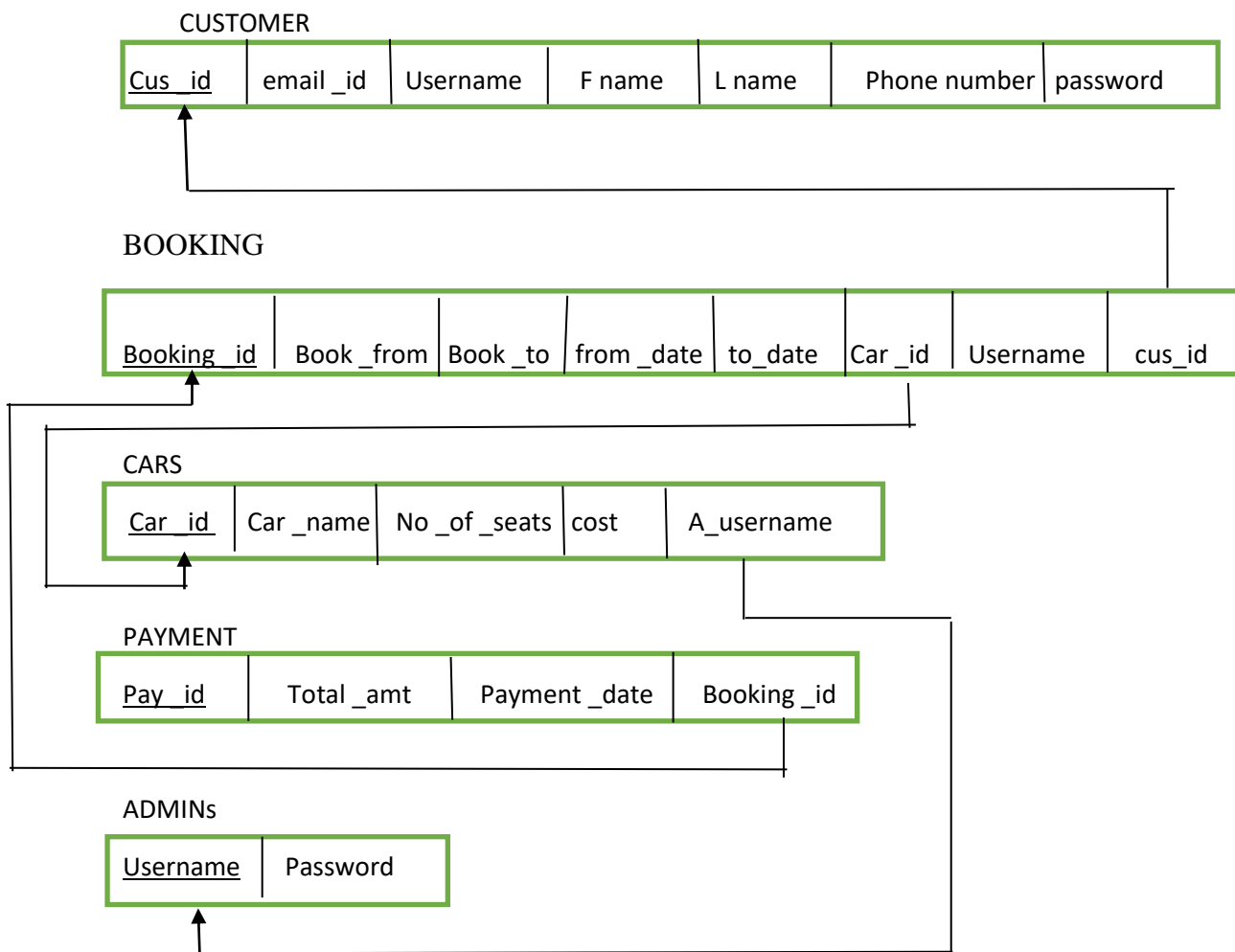


Fig:5.3 Relational Database Schema Diagram for Vehicle Management system

5.2.1 Mapping of 1:1 Relationship type:

For each 1:1 relationship type R can be migrated to any participating entity types. This relationship type ensures that each user in the database can lodge one complaint. There is 1:1 relationship between vehicle-to-vehicle history and vehicle to booked.

5.2.2 Mapping of 1: N Relationship type:

For each regular binary 1: N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as the foreign key in S the primary key of relation T that represents other entity type in R. We have to do this because each entity instance on N side is related to at most one entity instance on 1-side of relationship type.

There is 1: N relationship from user to vehicle and vehicle to fuel.

5.2.3 Mapping of M: N Relationship type:

For each M: N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity type, their combination will form the primary key of S. Also include any simple attributes of m: n relationship type.

There is a M: N relationship between driver and vehicle.

5.3 Database Tables / Relations

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 file 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 in a table 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. A table can be considered as a convenient representation of a relation. A table is perceived a two-dimensional structure composed of rows and columns. The order of rows and columns is immaterial to the DBMS.

Description of Tables:

Customer Table

Field	Type	Null	Key	Default	Extra
cus_id	int(20)	NO	PRI	NULL	auto_increment
username	varchar(20)	NO		NOT NULL	
Fname	varchar(20)	NO		NOT NULL	
Lname	varchar(20)	NO		NOT NULL	
email_id	varchar(20)	NO		NOT NULL	
password	varchar(20)	NO		NOT NULL	
Address	varchar(20)	NO		NOT NULL	
Phonenumber	varchar(20)	NO		NOT NULL	

Cars Table

Field	Type	Null	Key	Default	Extra
car_id	int(20)	NO	PRI	NULL	auto_increment
car_name	varchar(20)	NO		NOT NULL	
no_of_seats	int(20)	NO		NULL	
A_username	varchar(20)	NO	MUL	NULL	

Booking Table

Field	Type	Null	Key	Default	Extra
booking_id	int(20)	NO	PRI	NULL	auto_increment
booking_from	varchar(20)	NO		NOT NULL	
booking_to	varchar(20)	NO		NOT NULL	
booking_date	date	NO		NULL	
booking_date1	date	NO		NULL	
car_id	int(20)	NO	MUL	NULL	
username	varchar(20)	NO		NULL	
cus_id	int(20)	NO	MUL	NULL	

Payment Table

Field	Type	Null	Key	Default	Extra
pay id	int(20)	NO	PRI	NULL	auto_increment
total amount	varchar(20)	NO		NOT NULL	
payment date	date	NO		NULL	
booking_id	int(20)	NO	MUL	NULL	

Admin Table

Field	Type	Null	Key	Default	Extra
username	varchar(20)	NO	PRI	NULL	
password	varchar(20)	NO		NOT NULL	

5.4 Normalization

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

- Eliminating redundant(useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

1. First Normal Form –

If a relation contains composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is **singled valued attribute**.

Example: We will create a table to store student data which will have student's roll no., their name and the name of subjects they have opted for.

Roll_no	Name	Subject
101	Ram	OS, CN
103	Suresh	Java
102	Akshaya	C, C++

All our column names are unique, we have stored data in the order we wanted to and we have not inter-mixed different type of data in columns.

But out of the 3 different students in our table, 2 have opted for more than 1 subject. And we have stored the subject names in a single column. But as per the 1st Normal form each column must contain atomic value.

All we have to do is break the values into atomic values.

Here is our updated table and it now satisfies the First Normal Form.

Roll_no	Name	Subject
101	Ram	OS
101	Ram	CN
103	Suresh	Java
102	Akshay	C
102	Akshay	C++

By doing so, although a few values are getting repeated but values for the **subject** column are now atomic for each record/row.

Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

2. Second Normal Form –

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has **No Partial Dependency**, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

Example: Let's take an example of a Student table with columns **Student_id**, Name, Reg_no, Branch and Address.

In this table, **Student_id** is the primary key and will be unique for every row, hence we can use **Student_id** to fetch any row of data from this table.

Even for a case, where student names are same, if we know the **Student_id** we can easily fetch the correct record.

<u>Student_id</u>	Name	Reg_no	Branch	Address
10	Ram	07-WY	CSE	Kerala
11	Ram	08-WY	IT	Gujarat

Let's create another table for subject, which will have Subject_id(Primary key), and Subject_name fields.

<u>Subject_id</u>	Subject_name
1	Java
2	C++
3	Php

Let's create another table Score, to store the marks obtained by students in the respective subjects. We will also be saving name of the teacher who teaches that subject along with marks.

Score_id	<u>Student_id</u>	<u>Subject_id</u>	Marks	Teacher
1	10	1	70	Java Teacher
2	10	2	75	C++ Teacher
3	11	1	80	Java Teacher

Now if you look at the Score table, we have a column name **Teacher** which is only dependent on the Subject, for Java it's Java Teacher and for C++ it's C++ Teacher & so on. This is **Partial Dependency**, where an attribute in a table depends on only a part of the primary key and not on the whole key. The simplest solution is to remove columns Teacher from Score table and add it to the Subject table. Hence, the Subject table will become:

<u>Subject_id</u>	Subject_name	Teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

And our Score table is now in the second normal form, with no partial dependency.

Score_id	<u>Student_id</u>	<u>Subject_id</u>	Marks
1	10	1	70
2	10	2	75
3	11	1	80

3. Third Normal Form –

A relation is in third normal form, if there is **no transitive dependency** for non-prime attributes as well as it is in second normal form. A relation is in 3NF if **at least one of the following condition holds** in every non-trivial function dependency $X \rightarrow Y$

1. X is a super key.
2. Y is a prime attribute (each element of Y is part of some candidate key).

Transitive dependency – **If $A \rightarrow B$ and $B \rightarrow C$ are two FDs then $A \rightarrow C$ is called transitive dependency.**

Example: We have 3 tables Student, Subject and Score.

Student Table

<u>Student_id</u>	Name	Reg_no	Branch	Address
10	Ram	07-WY	CSE	Kerala
11	Ram	08-WY	IT	Gujarat
12	Akshay	09-WY	IT	Rajasthan

Subject Table

<u>Subject_id</u>	Subject_name	Teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

Score Table

Score_id	<u>Student_id</u>	<u>Subject_id</u>	Marks
1	10	1	70
2	10	2	75
3	11	1	80

In the Score table, we need to store some more information, which is the exam name and total marks, so let's add 2 more columns to the Score table.

<u>Score_id</u>	<u>Student_id</u>	<u>Subject_id</u>	Marks	Exam_name	Total_marks
-----------------	-------------------	-------------------	-------	-----------	-------------

Our new column Exam_name depends on both student and subject. For example, a mechanical engineering student will have Workshop exam but a computer science student won't. And for some subjects you have Practical exams and for some you don't. So we can say that Exam_name is dependent on both Student_id and Subject_id.

The column Total_marks depends on Exam_name as with exam type the total score changes. For example, practicals are of less marks while theory exams are of more marks.

Exam_name is just another column in the score table. It is not a primary key or even a part of the primary key, and Total_marks depends on it.

This is **Transitive Dependency**. When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

We take out the columns Exam_name and Total_marks from Score table and put them in an Exam table and use the Exam_id wherever required.

Score Table: In 3rd Normal Form

Score_id	<u>Student_id</u>	<u>Subject_id</u>	Marks	Exam_id
----------	-------------------	-------------------	-------	---------

The new **Exam table** :

<u>Exam_id</u>	Exam_name	Total_marks
1	Workshop	200
2	Mains	70
3	Practicals	30

The advantage of removing transitive dependency is,

- Amount of data duplication is reduced.
- Data integrity achieved.

CHAPTER 6

IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and giving confidence on the new system for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, an evaluation of change over methods. Apart from planning major task of preparing the implementation are education and training of users. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system. In network backup system no additional resources are needed.

Implementation using PHP:

```
<?php

$username = $_POST['username'];
$Fname = $_POST['firstname'];
$Lname = $_POST['lastname'];
$email_id = $_POST['email_id'];
$password = $_POST['pass1'];
$Address = $_POST['Address'];
$Phonenumber = $_POST['number'];

$conn = new mysqli('localhost','root','','rental');
if($conn->connect_error){
    die('Connection Failed : ' . $conn->connect_error);
}

$sql = "INSERT INTO customer(username, Fname, Lname, email_id, password, Address, Phonenumber)
VALUES ('$username', '$Fname', '$Lname', '$email_id', '$password', '$Address', '$Phonenumber')";

if ($conn->query($sql) === TRUE) {
    echo "<font size='18' face='Arial' color='#5de6de' text_align='center'>";
    echo "signup successfully";
    header('location:log.php');
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Figure 6.1: Connection of Signup page with backend

```

<?php
session_start();
$username = $_POST['username'];
$car_id = $_POST['car_id'];
$booking_from = $_POST['bookingfrom'];
$booking_to = $_POST['bookingto'];
$booking_date = $_POST['bookingdate'];
$booking_date1 = $_POST['bookingdate1'];
$conn = new mysqli('localhost','root','','rental');
if($conn->connect_error){
    die('Connection Failed : ' . $conn->connect_error);
}
$sql = "INSERT INTO booking(username, booking_from, booking_to, booking_date, booking_date1 ,car_id)
VALUES ('$username', '$booking_from', '$booking_to', '$booking_date', '$booking_date1', '$car')";
if($conn->query($sql) === TRUE) {
    //echo "Success";
    header('location:payment.php');
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
//if (isset($_COOKIE['cookie']))
//echo $_COOKIE["username"];
$conn->close();
?>

```

Figure 6.2: Connection of booking page with backend

```

<?php
session_start();
$username = $_SESSION['username'];
$conn = new mysqli("localhost", "root", "", "rental");
if($conn === false){
    die("ERROR: Could not connect. " . $conn->connect_error);
}
$sql = " select cars.car_name,booking.booking_id, booking.username, booking.booking_from, booking.booking_to, booking.booking_date, booking.booking_date1
from cars inner join booking ON cars.car_id = booking.car_id
WHERE booking.username = '$username'";
if($result = $conn->query($sql)){
    if($result->num_rows > 0){
        echo "<table id='1'>";
        echo "<tr>";
        echo "<th>car_name</th>";
        echo "<th>Booking_id</th>";
        echo "<th>Username</th>";
        echo "<th>Booking_from</th>";
        echo "<th>Booking_to</th>";
        echo "<th>Booked_date</th>";
        echo "<th>Booked_date1</th>";
        echo "<th>status of booking &
        payment</th>";
        echo "</tr>";
        while($row = $result->fetch_array()){
            echo "<tr>";
            echo "<td>" . $row['car_name'] . "</td>";
            echo "<td>" . $row['booking_id'] . "</td>";
            echo "<td>" . $row['username'] . "</td>";
            echo "<td>" . $row['booking_from'] . "</td>";
            echo "<td>" . $row['booking_to'] . "</td>";
            echo "<td>" . $row['booking_date'] . "</td>";
            echo "<td>" . $row['booking_date1'] . "</td>";
            echo "<td>" . 'Success' . "</td>";
            echo "</tr>";
        }
        echo "</table>";
        $result->free();
    } else{
        echo "No records matching your query were found.";
    }
} else{
    echo "ERROR: Could not able to execute $sql. " . $conn->error;
}
$conn->close();
?>

```

Figure 6.3: Retrieving data from backend using various PHP functions

CHAPTER 7

RESULTS

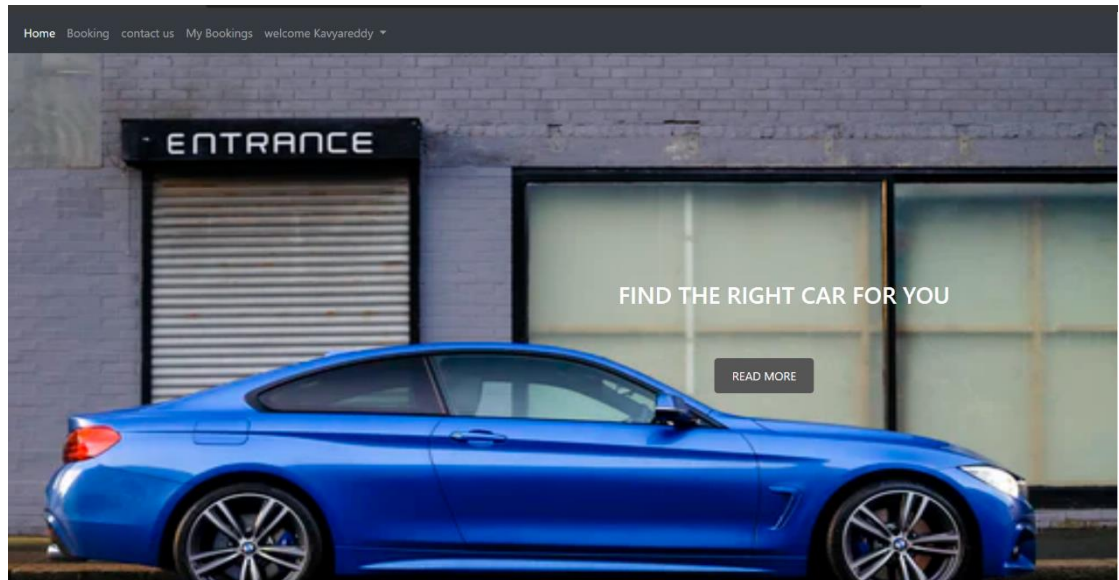


Figure 7.1: Welcome page

A screenshot of a "SIGN UP" form. The form is dark-themed with white text and input fields. It includes fields for Username, First Name, Last Name, Email Id, Password, Address, and Phone number. Below the Phone number field is a "Sign Up" button. At the bottom, there is a link: "Already User ? Login".A screenshot of a "Log In" form. The form is dark-themed with white text and input fields. It includes fields for Username and Password. Below the Password field is a "Log In" button. At the bottom, there is a link: "Not a user? Sign up".

Figure 7.2: Login and signup form

Click to go back, hold to see history







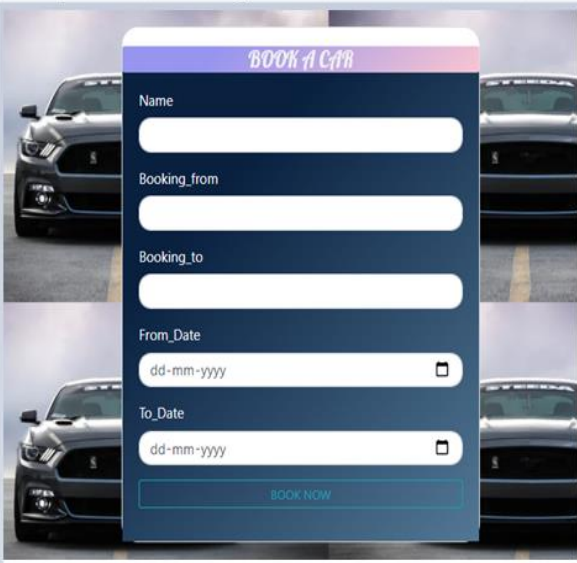
 <p>petrol 5 seats</p> <p>SHIFT BALENO ₹893/DAY</p> <p><input type="checkbox"/> shift baleno <input type="button" value="Submit"/></p>	 <p>Diesel 4 seats</p> <p>HYUNDAI I20 ₹1000/DAY</p> <p><input type="checkbox"/> hyundai i20 <input type="button" value="Submit"/></p>	 <p>Diesel 8 seats</p> <p>INNOVA CRYSTA ₹1200/DAY</p> <p><input type="checkbox"/> Innova crysta <input type="button" value="Submit"/></p>
 <p>Diesel 6 seats</p> <p>SUZUKI BALENO ₹800/DAY</p> <p><input type="checkbox"/> Suzuki nalen0 <input type="button" value="Submit"/></p>	 <p>Diesel 5 seats</p> <p>BMW ₹1400/DAY</p> <p><input type="checkbox"/> BMW <input type="button" value="Submit"/></p>	 <p>Diesel 9 seats</p> <p>MAHINDRA ₹1600/DAY</p> <p><input type="checkbox"/> Mahindra <input type="button" value="Submit"/></p>

Figure 7.3: Cars details



BOOK A CAR

Name

Booking_from

Booking_to

From_Date

To_Date

MAKE YOUR PAYMENT

NAME

CARD NUMBER

EXPIRY DATE(MM YYYY)


CVC/CVV

Figure 7.4: Booking and Payment pages

Booking Details

car_name	Booking_id	Username	Booking_from	Booking_to	Booked_date	Booked_date1	status of booking & payment
HYUNDAI I20	1	Kavyareddy	chintamani	hyderbhad	2020-12-24	2020-12-26	Sucesss
BMW	26	Kavyareddy	bangalore	hyderbhad	2021-01-10	2021-01-12	Sucesss
MAHINDRA	27	Kavyareddy	bangalore	hyderbhad	2021-01-19	2021-01-20	Sucesss

Figure 7.5: Customer report



Customer Details

cus_id	Username	Fname	Lname	Email	Password	Address	Phonenumber
1	Kavyareddy	kavya	reddy	kavyareddyw07@gmail	123456789	delhi	9836578348
2	sathyai2	sathya	nare:nareddy	nsathya7411@gmail.c	sathya	chintamani	08784358787
3	kishor24	kishore	reddy	test123@gmail.com	987654321	kochi	08784358787

Booking Details

car_id	car_name	Booking_id	Username	Booking_from	Booking_to	Booked_date
2	HYUNDAI I20	1	Kavyareddy	chintamani	hyderbhad	2020-12-24
4	SUZUKI BALENO	2	sathyai2	chintamani	hyderbhad	2020-12-23
3	INNOVA CRYSTA	9	kishor24	puti	sams	2020-12-28
5	BMW	26	Kavyareddy	bangalore	hyderbhad	2021-01-10
6	MAHINDRA	27	Kavyareddy	bangalore	hyderbhad	2021-01-19

Cars Details

car_id	carname	No. of seats	Cost
1	SHIFT BALENO	5	893/day
2	HYUNDAI I20	4	1000/day
3	INNOVA CRYSTA	8	1200/day
4	SUZUKI BALENO	6	800/day
5	BMW	5	1400/day
6	MAHINDRA	9	1600/DAY

Figure 7.6: Admin page (Registered customer, Booked cars, Available cars details)

CHAPTER 8

CONCLUSION AND FUTURE WORK

Conclusion

Car Rental System has rich user interface so that users can access the application easily. Based on the research done, some car rental companies still use desktop application for their car rental services and thus making it to be limited to so many important features that are not available unlike in the web-based application where there are so many features available. Also, some upcoming companies do not only make use of these desktop applications, but also make use of phone call reservation, which is still lacking so many features that are needed for this type of system.

Future Work

- The most recommended solution to these problems is to implement a web-based system that will have the features required for this kind of services or business.
- The system will be able to serve as a web base application when it is finally developed, where these small upcoming companies can make use of it to publish their services in a wide range and also help the company to manage their service more effectively.

Future Work

- Mobile Application could be developed for the system to reach out to more users.
- This application can be extended to be implemented in the RTO office by adding more features.
- We can further modify the application to include operations like renting the vehicles to customers.

CHAPTER 9

REFERENCES

1. Fundamental of Database Systems by Ramez Elamasri and Shamkant B Navathe, Sixth Edition, Addison Wesley, 2011.
2. <https://www.w3schools.com/html/default.asp>
3. <https://www.w3schools.com/css/default.asp>
4. <https://www.w3schools.com/sql/default.asp>
5. <https://www.w3schools.com/php/default.asp>
6. <https://www.geeksforgeeks.org/>