CONTENTS

**CHAPTERS PAGE NO**

## **1. INTRODUCTION**

1.1 General Overview ………….03

1.2 Problem Definition ………….03

1.3 Objectives ………….03

1.4 Purpose …………04

**2. SOFTWARE AND HARDWARE REQUIREMENT**

2.1 Software Requirements …………05

2.2 Hardware requirements …………05

2.3 Front End Tool …………05

2.4 Back End Tool …………07

**3. DESIGN**

3.1 ER Diagram …………09 .4 CHEMAER THE SONGS(joining 3 tables)ABLES)ast preformed of a oloning 3 tables here.It is a view.noted.

3.2 Schema …………11

3.3 Normalization …………11

**4. IMPLEMENTATION**

4.1 Front-End code

4.1.1 Hyper Text Markup Language …………13

4.1.2 HTML code of index page ………...13

4.2 Back-End Code

4.2.1 The Initial Code ………...36

4.2.2 Database Connectivity and Access ………..37

4.2.3 Triggers ……….38

4.2.4 Stored Procedure ………. 39

**5. TESTING** ……….42

**6. SNAPSHOTS** ……….44

**7. REFERENCES** ………49

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENREAL OVERVIEW:**

A Database Management System (DBMS) is system software for creating and managing databases. A database is a collection of information that is organized so that it can be easily accessed managed and updated. The DMBS provides users and programmers with a systematic way to create, retrieve, update and manage data. Most of giant IT farms are investing billions on business information from a raw data manage them in files and document which may sometimes even suffer loss hence no security or backup. This project makes a decent effort overcome this problem.

Cab rides database system project aims at developing a website that provides the required details about the cab rides, customers and admin are various users of this database synchronized services. The database contains information about the details of cab ride like the trip start and end date-time, driver name, rider name, fare etc.

.

**1.2 PROBLEM DEFINITION:**

This project aims at developing a website that provides the required details about the cab rides customers have taken and related information available to the various users. By considering the inputs from the admin regarding the different tables, the required modifications can be done and corresponding changes are updated in database. This project provides practical solution of managing a company providing cab service as per the details given by the user.

**1.3 OBJECTIVES:**

The main objective of this project is to maintain and manage all the details regarding any cab ride a user takes. Any organization which wishes to manage its cab rides database can easily make use of this model and make it easier and advanced. The main objectives are given below:

* The project needs proper details of cab rides, driver and rider.
* Each rider can see details of each ride they have taken. This enables the easy access to necessary information.
* Rides table has trip date and timings with start date-time and end date-time which gives the information about the timings and duration of the required concert. Also it contains the fare charged for each ride.
* Admin interface for easy modifications of database and also access to the complete database which is not provided for riders.

All these objectives are fulfilled by the proposed project and it includes triggers and stored procedures wherever required. Also, provision for insertion, deletion and modification increases the ease of using this system.

**1.4 PURPOSE:**

The main purpose of this project is to build a database that would benefit the customers (riders) in viewing the details of their rides and admin privileges for management of this database. In the present era there is a wave of digitalizing most of the services being provided. So, organizing and managing these services becomes a need. This process can be supported only if we have an effective database system.

The following are the facilities that are provided by the system:

* View RIDE details
* View DRIVER details
* View RIDER details
* Insert, delete and modify all the details required
* Rider account login
* Separate interface for Admin following successful login
* Stored procedure for registering of customers
* Trigger for the calculation of fare

**CHAPTER 2**

**SOFTWARE AND HARDWARE REQUIREMENTS**

The hideous methods to search for the details of a completed ride which is practically tedious in the current system is overcome by this project which will save the valuable time.

**2.1 SOFTWARE REQUIREMENTS:**

|  |  |
| --- | --- |
| **NAME OF THE COMPONENT** | **SPECIFICATION** |
| Operating System | Ubuntu, Windows 10 |
| Language | HTML, CSS, PYTHON |
| Database | MYSQL DATABASE |
| Browser | Any of Mozilla, Google Chrome, Microsoft Edge |
| Web Server | Apache |
| Software Development kit | Visual Studio Code, Flask |
| Scripting Language enable | HTML, JS, CSS |
| Database Connection | MySQL Connection |

FIG 2.1 Software requirements

**2.2 HARDWARE REQUIREMENTS:**

|  |  |
| --- | --- |
| **NAME OF THE COMPONENT** | **SPECIFICATION** |
| Processor | Core i7 processor |
| RAM | 8 GB |
| Hard Disk | 60 GB |
| Monitor | 15” color monitor |
| Keyboard | 122 keys |

FIG 2.2 Hardware requirements

**2.3 FRONT END TOOLS: HTML, CSS, JS**

* **Hypertext Markup Language** (**HTML**) is the standard markup language for creating web pages and web applications.
* With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.
* Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
* HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items.
* HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.
* **Cascading Style Sheets** (**CSS**) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.
* CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
* CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.
* The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.
* **JavaScript** often abbreviated as **JS**, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.
* Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.
* As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles.
* It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.
* Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design; JavaScript was influenced by programming languages such as ‘Self’ and ‘Scheme’.

**2.4 BACK-END TOOL: MySQL, Flask-PYTHON**

* MYSQL server provides facility to store large amount of data is used in no. of present websites. It is used at the back end to store information related to site requirement.
* Now MYSQL server provides transaction support and tightly integrated width with PHP and Apache for fast and secure performance.
* MYSQL has emerged as the most popular RDBMS. As with other RDBMS products. MYSQL provides you with a rich set of features that support a secure environment for storing, maintaining and accessing data.
* MYSQL is an open source database management system (DBMS) in the world. MYSQL supports a vast array of security and configuration options, enabling you to take total control over just about every imaginable aspect of its operation.
* **Flask** is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
* However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.
* Extensions are updated far more regularly than the core Flask program. Flask is commonly used with MongoDB, which gives it more control over databases and history.
* Applications that use the Flask framework include Pinterest, LinkedIn and the community web page for Flask itself.

The following list provides an overview of the important features found in MYSQL:

* MYSQL can handle large databases, which has been demonstrated by its implementation in organizations. MYSQL includes a powerful system to control access data. The system uses a host and user based structure that controls who can access specific information and the level of access to that information.
* MYSQL is simple to install and implement. A user can have a MYSQL installation up and running within minutes after downloading the files. Even at administrative level, MYSQL is relatively easy to optimize, especially compared to other RDBMS products.
* MYSQL makes the MYSQL source code available to everyone to download and use. The open source philosophy allows a global audience to participate in the review.

**CHAPTER 3**

**DESIGN**

The most creative and challenging face of the system development is System Design. It provides the understanding and procedural details necessary for the logical and physical stages of development. In designing a new system, the system analyst must have a clear understanding of the objectives, which the design is aiming to fulfill. The first step is to determine how the output is to be produced and in what format. Second, input data and master files have to be designed to meet the requirements of the proposed output. The operational phases are handled through program construction and testing.

Design of the system can be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Thus system design is a solution to “how to” approach to the creation of a new system. This important phase provides the understanding and the procedural details necessary for implementing the system recommended in the feasibility study. The design step provides a data design, architectural design, and a procedural design.

**3.1 ER DIAGRAM**

The schemas for the database application can be displayed by means of graphical notation known as Entity Relationship diagram.

The ER model describes data as entities, relationships and attributes.

*3.1.1. ENTITIES AND ATTRIBUTES*

An entity may be an object with a physical existence (for e.g. A particular person, car or employee) or it may be an object with a conceptual existence (for e.g. a company, a job, or a university course)

Each entity has attributes i.e. the particular properties that describe it. The attribute values that describe each entity become a major part of the data store in the database.

*3.1.2 RELATIONSHIPS BETWEEN ENTITIES*

whenever an attribute of one entity type refers to another entity type, a relationship exists. In the initial design of entity types, relationships are typically captured in the form of attributes. As the design is refined these attributes get converted into relationships between entity types. In the ER diagrams the emphasis is on representing the schemas rather than the instances. This is more useful in the database design because a database schema changes rarely, whereas contents of the entity sets change frequently.

**ER Explanation:**

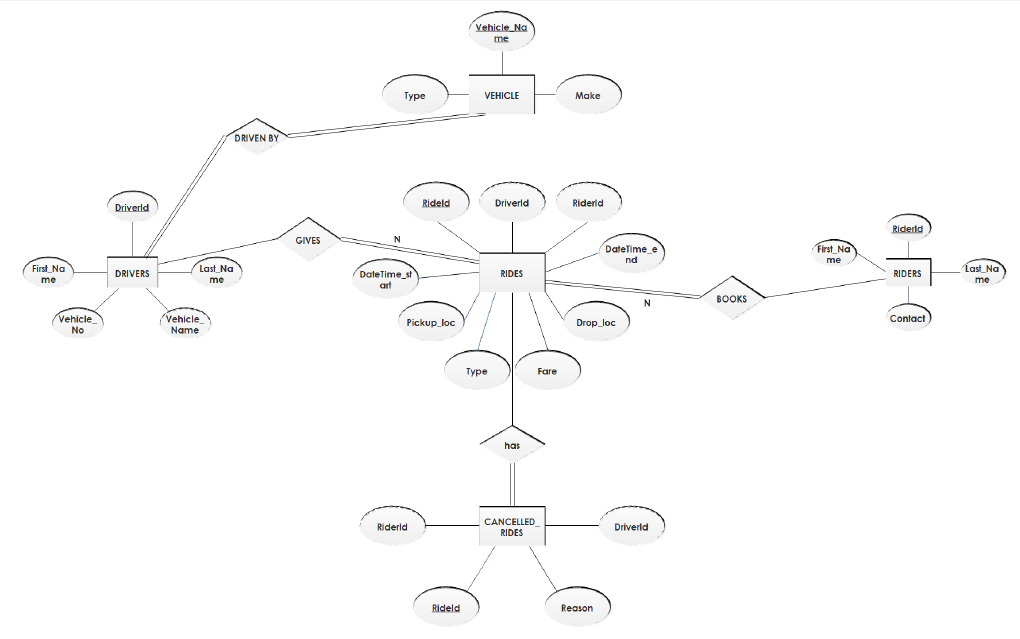
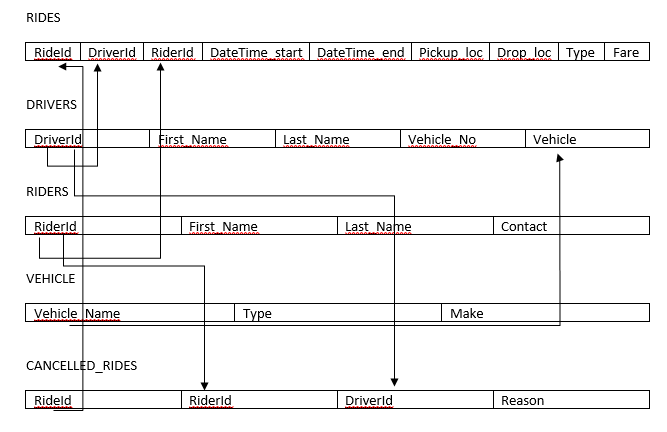


FIGURE 3.1 ER Diagram of Cab rides database system

* The system has 5 regular entities and 4 relational entity.
* The attributes are shown above.
* All attributes are atomic attributes. With those underlined are primary key of the entities.
* The relationship DRIVEN BY has a cardinality ratio 1:n and it has total participation at DRIVERS side. As, 1 driver can have only one vehicle registered in system of the cab rides database.
* The relationship BOOKS has a cardinality ratio 1:n and it has total participation at RIDERS side. As, 1 ride cannot be booked by more than one rider in system of the cab rides database.
* The relationship GIVES has a cardinality ratio 1:n and it has total participation at DRIVERS side. As, 1 ride cannot have more than one driver in system of the cab rides database.
* HAS is relational entity having m:n and total participation at CANCELLED\_RIDES end.
* RIDES, DRIVERS, RIDERS, VEHICLE and CANCELLED\_RIDES are the regular entities.

**3.2. ER SCHEMA**



3.2 ER SCHEMA DIAGRAM

**3.3 NORMALIZATION**

* FIRST NORMAL FORM (1NF)
* There are no repeated or duplicate fields
* Each cell contains only one single value and each record is unique
* Hence, the database is in the first normal form
* SECOND NORMAL FORM (2NF)
* All non-key fields depend on all components of the primary key which is guaranteed when the primary key is a single field
* Hence, the database is in 2NF as it satisfies all the required conditions.
* THIRD NORMAL FORM (3NF)
* All non-key fields depend only on the primary key
* All entities are under 3NF, hence we can conclude that the entities satisfy all the normal forms.

**CHAPTER 4**

**IMPLEMENTATION**

* 1. **FRONT END CODE**

***4.1.1 HYPER TEXT MARKUP LANGUAGE:***

The Hyper Text Markup language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of applications. This specifications defines HTML version 4.0 HTML 4.0 aims to capture recommended practice as of early ’96 and as such to be used as a replacement for HTML 3.2.

**4.1.2 HTML CODE FOR INDEX PAGE**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Cab RIDES</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/png" href="../static/images/icons/cab.jpg"/>

<link rel="stylesheet" type="text/css" href="../static/vendor/bootstrap/css/bootstrap.min.css">

<link rel="stylesheet" type="text/css" href="../static/fonts/font-awesome-4.7.0/css/font-awesome.min.css">

<link rel="stylesheet" type="text/css" href="../static/vendor/animate/animate.css">

<link rel="stylesheet" type="text/css" href="../static/vendor/css-hamburgers/hamburgers.min.css">

<link rel="stylesheet" type="text/css" href="../static/vendor/select2/select2.min.css">

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

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

</head>

<body>

<div class="limiter">

<div class="container-login100">

<div class="wrap-login100">

<div class="container-login100-form-btn">

<a href="/logout" style="margin-left: 80%;">

<button class="login100-form-btn" >

Logout

</button>

</a>

</div>

<a href='rider'>

<button class="btn btn-dark">Rider Info</button>

</a>

<span class="login100-page-title" >

<u><b>Cab Rides Database</b></u>

</span>

<table class="table table-striped table-dark">

<thead>

<tr>

<!--<th scope="col">#</th>-->

<th scope="col">RideId</th>

<th scope="col">DriverId</th>

<th scope="col">Ride Start Time</th>

<th scope="col">Ride End Time</th>

<th scope="col">Pickup Location</th>

<th scope="col">Drop Location</th>

<th scope="col">Type</th>

<th scope="col">Fare</th>

</tr>

</thead>

<tbody>

{% for row in rows %}

<tr>

<td>{{row[0]}}</td>

<td>{{row[1]}}</td>

<td>{{row[2]}}</td>

<td>{{row[3]}}</td>

<td>{{row[4]}}</td>

<td>{{row[5]}}</td>

<td>{{row[6]}}</td>

<td>{{row[7]}}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

<script src="../static/vendor/jquery/jquery-3.2.1.min.js"></script>

<script src="../static/vendor/bootstrap/js/popper.js"></script>

<script src="../static/vendor/bootstrap/js/bootstrap.min.js"></script>

<script src="../static/vendor/select2/select2.min.js"></script>

<script src="../static/vendor/tilt/tilt.jquery.min.js"></script>

<script >

$('.js-tilt').tilt({

scale: 1.1

})

</script>

<script src="../static/js/main.js"></script>

</body>

</html>

**CSS CODE FOR STYLING**

/\*//////////////////////////////////////////////////////////////////

[ FONT ]\*/

@font-face {

font-family: Poppins-Regular;

src: url('../fonts/poppins/Poppins-Regular.ttf');

}

@font-face {

font-family: Poppins-Bold;

src: url('../fonts/poppins/Poppins-Bold.ttf');

}

@font-face {

font-family: Poppins-Medium;

src: url('../fonts/poppins/Poppins-Medium.ttf');

}

@font-face {

font-family: Montserrat-Bold;

src: url('../fonts/montserrat/Montserrat-Bold.ttf');

}

/\*//////////////////////////////////////////////////////////////////

[ RESTYLE TAG ]\*/

\* {

margin: 0px;

padding: 0px;

box-sizing: border-box;

}

body, html {

height: 100%;

font-family: Poppins-Regular, sans-serif;

}

/\*---------------------------------------------\*/

a {

font-family: Poppins-Regular;

font-size: 14px;

line-height: 1.7;

color: #666666;

margin: 0px;

transition: all 0.4s;

-webkit-transition: all 0.4s;

-o-transition: all 0.4s;

-moz-transition: all 0.4s;

}

a:focus {

outline: none !important;

}

a:hover {

text-decoration: none;

color: #57b846;

}

/\*---------------------------------------------\*/

h1,h2,h3,h4,h5,h6 {

margin: 0px;

}

p {

font-family: Poppins-Regular;

font-size: 14px;

line-height: 1.7;

color: #666666;

margin: 0px;

}

ul, li {

margin: 0px;

list-style-type: none;

}

/\*---------------------------------------------\*/

input {

outline: none;

border: none;

}

textarea {

outline: none;

border: none;

}

textarea:focus, input:focus {

border-color: transparent !important;

}

input:focus::-webkit-input-placeholder { color:transparent; }

input:focus:-moz-placeholder { color:transparent; }

input:focus::-moz-placeholder { color:transparent; }

input:focus:-ms-input-placeholder { color:transparent; }

textarea:focus::-webkit-input-placeholder { color:transparent; }

textarea:focus:-moz-placeholder { color:transparent; }

textarea:focus::-moz-placeholder { color:transparent; }

textarea:focus:-ms-input-placeholder { color:transparent; }

input::-webkit-input-placeholder { color: #999999; }

input:-moz-placeholder { color: #999999; }

input::-moz-placeholder { color: #999999; }

input:-ms-input-placeholder { color: #999999; }

textarea::-webkit-input-placeholder { color: #999999; }

textarea:-moz-placeholder { color: #999999; }

textarea::-moz-placeholder { color: #999999; }

textarea:-ms-input-placeholder { color: #999999; }

/\*---------------------------------------------\*/

button {

outline: none !important;

border: none;

background: transparent;

}

button:hover {

cursor: pointer;

}

iframe {

border: none !important;

}

/\*//////////////////////////////////////////////////////////////////

[ Utility ]\*/

.txt1 {

font-family: Poppins-Regular;

font-size: 13px;

line-height: 1.5;

color: #999999;

}

.txt2 {

font-family: Poppins-Regular;

font-size: 13px;

line-height: 1.5;

color: #666666;

}

/\*//////////////////////////////////////////////////////////////////

[ login ]\*/

.limiter {

width: 100%;

margin: 0 auto;

}

.container-login100 {

width: 100%;

min-height: 100vh;

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

flex-wrap: wrap;

justify-content: center;

align-items: center;

padding: 15px;

background: #d7f13e;

background: -webkit-linear-gradient(-135deg, #e92233, #13141a);

background: -o-linear-gradient(-135deg, #e92233, #13141a);

background: -moz-linear-gradient(-135deg, #e92233, #13141a);

background: linear-gradient(-135deg, #e92233, #13141a);

}

.wrap-login100 {

width: 90%;

background: #fff;

border-radius: 10px;

overflow: hidden;

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

flex-wrap: wrap;

justify-content: space-between;

padding: 120px 30px 30px 75px;

}

/\*------------------------------------------------------------------

[ ]\*/

.login100-pic {

width: 316px;

}

.login100-pic img {

max-width: 100%;

}

/\*------------------------------------------------------------------

[ ]\*/

.login100-form {

width: 300px;

}

.login100-form-title {

font-family: Poppins-Bold;

font-size: 24px;

color: #333333;

line-height: 1.0;

text-align: center;

width: 100%;

display: block;

padding-bottom: 54px;

}

.login100-page-title {

font-family: Montserrat-Bold;

font-size: 35px;

color: #202c21cb;

line-height: 0;

text-align: center;

top: 0px;

width: 100%;

display: inline-block;

padding-bottom: 70px;

}

/\*---------------------------------------------\*/

.wrap-input100 {

position: relative;

width: 100%;

z-index: 1;

margin-bottom: 10px;

}

.input100 {

font-family: Poppins-Medium;

font-size: 15px;

line-height: 1.5;

color: #666666;

display: block;

width: 100%;

background: #e6e6e6;

height: 50px;

border-radius: 25px;

padding: 0 30px 0 68px;

}

/\*------------------------------------------------------------------

[ Focus ]\*/

.focus-input100 {

display: block;

position: absolute;

border-radius: 25px;

bottom: 0;

left: 0;

z-index: -1;

width: 100%;

height: 100%;

box-shadow: 0px 0px 0px 0px;

color: rgba(87,184,70, 0.8);

}

.input100:focus + .focus-input100 {

-webkit-animation: anim-shadow 0.5s ease-in-out forwards;

animation: anim-shadow 0.5s ease-in-out forwards;

}

@-webkit-keyframes anim-shadow {

to {

box-shadow: 0px 0px 70px 25px;

opacity: 0;

}

}

@keyframes anim-shadow {

to {

box-shadow: 0px 0px 70px 25px;

opacity: 0;

}

}

.symbol-input100 {

font-size: 15px;

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

align-items: center;

position: absolute;

border-radius: 25px;

bottom: 0;

left: 0;

width: 100%;

height: 100%;

padding-left: 35px;

pointer-events: none;

color: #666666;

-webkit-transition: all 0.4s;

-o-transition: all 0.4s;

-moz-transition: all 0.4s;

transition: all 0.4s;

}

.input100:focus + .focus-input100 + .symbol-input100 {

color: #57b846;

padding-left: 28px;

}

/\*------------------------------------------------------------------

[ Button ]\*/

.container-login100-form-btn {

width: 100%;

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

flex-wrap: wrap;

justify-content: center;

padding-top: 20px;

}

.login100-form-btn {

font-family: Montserrat-Bold;

font-size: 15px;

line-height: 1.5;

color: rgb(228, 240, 59);

text-transform: uppercase;

width: 100%;

height: 50px;

border-radius: 25px;

background: #5a4fb9;

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

justify-content: center;

align-items: center;

padding: 0 25px;

-webkit-transition: all 0.4s;

-o-transition: all 0.4s;

-moz-transition: all 0.4s;

transition: all 0.4s;

}

.login100-form-btn:hover {

background: #333333;

}

/\*------------------------------------------------------------------

[ Responsive ]\*/

@media (max-width: 992px) {

.wrap-login100 {

padding: 177px 90px 33px 85px;

}

.login100-pic {

width: 35%;

}

.login100-form {

width: 50%;

}

}

@media (max-width: 768px) {

.wrap-login100 {

padding: 100px 80px 33px 80px;

}

.login100-pic {

display: none;

}

.login100-form {

width: 100%;

}

}

@media (max-width: 576px) {

.wrap-login100 {

padding: 100px 15px 33px 15px;

}

}

/\*------------------------------------------------------------------

[ Alert validate ]\*/

.validate-input {

position: relative;

}

.alert-validate::before {

content: attr(data-validate);

position: absolute;

max-width: 70%;

background-color: white;

border: 1px solid #c80000;

border-radius: 13px;

padding: 4px 25px 4px 10px;

top: 50%;

-webkit-transform: translateY(-50%);

-moz-transform: translateY(-50%);

-ms-transform: translateY(-50%);

-o-transform: translateY(-50%);

transform: translateY(-50%);

right: 8px;

pointer-events: none;

font-family: Poppins-Medium;

color: #c80000;

font-size: 13px;

line-height: 1.4;

text-align: left;

visibility: hidden;

opacity: 0;

-webkit-transition: opacity 0.4s;

-o-transition: opacity 0.4s;

-moz-transition: opacity 0.4s;

transition: opacity 0.4s;

}

.alert-validate::after {

content: "\f06a";

font-family: FontAwesome;

display: block;

position: absolute;

color: #c80000;

font-size: 15px;

top: 50%;

-webkit-transform: translateY(-50%);

-moz-transform: translateY(-50%);

-ms-transform: translateY(-50%);

-o-transform: translateY(-50%);

transform: translateY(-50%);

right: 13px;

}

.alert-validate:hover:before {

visibility: visible;

opacity: 1;

}

@media (max-width: 992px) {

.alert-validate::before {

visibility: visible;

opacity: 1;

}

}

**4.2 BACK END CODE**

from flask import Flask, render\_template, redirect, request, session, url\_for

from flask\_mysqldb import MySQL

app=Flask(\_\_name\_\_)

app.config['MYSQL\_HOST'] ='localhost'

app.config['MYSQL\_USER']='root'

app.config['MYSQL\_PASSWORD']='vaibhav99'

app.config['MYSQL\_DB']='cabrides'

app.secret\_key = '%jsdj!@'

mysql=MySQL(app)

afname = "Vaibhav"

acontact = "7587708800"

@app.route('/')

def main():

if request.method == "POST":

if 'contact' in session:

if session['contact'] == acontact:

return redirect('/adminPage')

return redirect(url\_for('index'))

return redirect(url\_for('index'))

@app.route('/index', methods=['GET'])

def index():

if 'contact' in session:

con = mysql.connection

cur = con.cursor()

contact=session['contact']

cur.execute("select RiderId from RIDERS where Contact=(%s)",(contact,))

riderid = cur.fetchall()

cur.execute("select RideId,DriverId,DateTime\_start,DateTime\_end,Pickup\_loc,Drop\_loc,Type,Fare from RIDES where RiderId=(%s)",(riderid[0],))

rows=cur.fetchall()

return render\_template('index.html',rows=rows)

else:

return redirect(url\_for('login'))

**4.2.2 DATABASE CONNECTIVITY AND ACCESS**

Before reading or writing a database, a connection must be made to it. MySQL database connection in flask requires importing MySQL module and then giving the important credentials needed to connect to database.

The code needed to connect to the Access database is:

from flask\_mysqldb import MySQL

app=Flask(\_\_name\_\_)

app.config['MYSQL\_HOST'] ='localhost'

app.config['MYSQL\_USER']='root'

app.config['MYSQL\_PASSWORD']='vaibhav99'

app.config['MYSQL\_DB']='cabrides'

mysql=MySQL(app)

A successful connection will allow SQL commands to be executed from Python to read or write the database.

**4.2.3 TRIGGERS**

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database . For example: when a new record(representing a new worker ) is added to the employees table,new records also should be created in the tables of the taxes ,vacations and salaries

.Triggers can also be used to log historical data ,

For example: To keep the track of employees previous salaries.

\* Trigger used in this application is calculating fare for a ride by considering the duration of ride.

DELIMITER $$

CREATE TRIGGER `RIDES\_FARE` BEFORE INSERT ON `RIDES` FOR EACH ROW

BEGIN

UPDATE RIDES

SET new.Fare = (TIMESTAMPDIFF(MINUTE,new.DateTime\_start,new.DateTime\_end)\*3);

END$$

DELIMITER ;

**4.2.4 STORED PROCEDURE**

A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs.

\* This stored procedure is called to obtain RIDERS details after the SignUp page submission.

CODE FOR STORED PROCEDURE**:**

CREATE PROCEDURE sp() NOT DETERMINISTIC NO SQL SQL SECURITY DEFINER select \* from RIDERS

**QUERIES:**

**Query1:**

@app.route('/index', methods=['GET'])

def index():

if 'contact' in session:

con = mysql.connection

cur = con.cursor()

contact=session['contact']

cur.execute("select RiderId from RIDERS where Contact=(%s)",(contact,))

riderid = cur.fetchall()

cur.execute("select RideId,DriverId,DateTime\_start,DateTime\_end,Pickup\_loc,Drop\_loc,Type,Fare from RIDES where RiderId=(%s)",(riderid[0],))

rows=cur.fetchall()

return render\_template('index.html',rows=rows)

else:

return redirect(url\_for('login'))

**Query2:**

@app.route('/rider')

def rider():

if 'contact' in session:

con=mysql.connection

cur=con.cursor()

contact=session['contact']

if session['contact']==acontact:

cur.execute('select \* from RIDERS where Contact=%s',(acontact,))

row=cur.fetchall()

return render\_template('admin.html',rows=row)

cur.execute('select \* from RIDERS where Contact=%s',(contact,))

row=cur.fetchall()

return render\_template('rider.html',rows=row)

return render\_template('login.html')

**Query3:**

@app.route('/adminPage')

def adminPage():

if (session['contact'] == acontact):

con = mysql.connection

cur = con.cursor()

cur.execute("select \* from RIDES")

rows = cur.fetchall()

cur.execute("select \* from DRIVERS")

rows1=cur.fetchall()

cur.execute("select \* from RIDERS")

rows2=cur.fetchall()

cur.execute("select \* from VEHICLE")

rows3=cur.fetchall()

cur.execute("select \* from CANCELLED\_RIDES")

rows4=cur.fetchall()

return render\_template('adminPage.html',rows=rows, rows1=rows1, rows2=rows2, rows3=rows3, rows4=rows4)

return redirect('/index')

**Query4:**

@app.route('/delete',methods=['GET','POST'])

def delete():

if session['contact']==acontact:

con=mysql.connection

cur=con.cursor()

if(request.method=='POST'):

if 'dcrideform' in request.form:

crideid=request.form['crideid']

cur.execute('delete from CANCELLED\_RIDES where RideId=%s',(crideid))

return redirect('/adminPage')

if 'drideform' in request.form:

rideid=request.form['rideid']

cur.execute('delete from RIDES where RideId=%s',(rideid))

return redirect('/adminPage')

if 'ddriverform' in request.form:

driverid=request.form['driverid']

cur.execute('delete from DRIVERS where DriverId=%s',(driverid))

return redirect('/adminPage')

if 'driderform' in request.form:

riderid=request.form['riderid']

cur.execute('delete from RIDERS where RiderId=%s',(riderid))

con.commit()

return redirect('/adminPage')

if 'dvform' in request.form:

vname=request.form['vname']

cur.execute('delete from VEHICLE where Vehicle\_Name=%s',(vname))

return redirect('/adminPage')

return render\_template('delete.html')

return render\_template('login.html')

**CHAPTER 5**

**TESTING**

Testing is vital for the success of any software. No system design is ever perfect. Testing is also carried in two phases. First phase is during the software engineering that is during the module creation. Second phase is after the completion of software. This is system testing which verifies that the whole set of programs hanged together.

The main aspect of software engineers is to distinguish software faults from software failures. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

Software Testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software.

**TYPES OF TESTING:**

**Unit Testing**

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

**Integration Testing**

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

**System Testing:** System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | INPUT | OUTPUT | REMARKS |
| 1 | Open in chrome, localhost:5000/ | Localhost refused to connect  This site can’t be reached.  ERR\_CONNECTION | Flask not started  My sql and Flask are not connected. |
| 2 | Open in chrome, the localhost | This site can’t be reached  Localhost refused to connect.  ERR\_CONNECTION\_REFUSED | Port not specified. |
| 3 | Insertion  INSERT INTO VEHICLE(White Bolt, Hatchback, Tata); | Error: duplicate entry ‘100’ for key ‘primary’ | UNIQUE constraint violated. Integrity constraint. |
| 4 | Insertion  INSERT INTO DRIVERS VALUES (4, Sebastian, Vettel, KA 51 MV 0210); | Cannot add or update a child row: a foreign key constraint fails.  CONSTRAINT dvehicle FOREIGN KEY(Vehicle) REFERENCES VEHICLE(Vehicle\_Name); | Vehicle is a foreign key hence can’t be inserted until the Vehicle\_Name is added in the VEHICLE table. |

**CHAPTER 6**

**SNAPSHOTS**

1. LOGIN PAGE

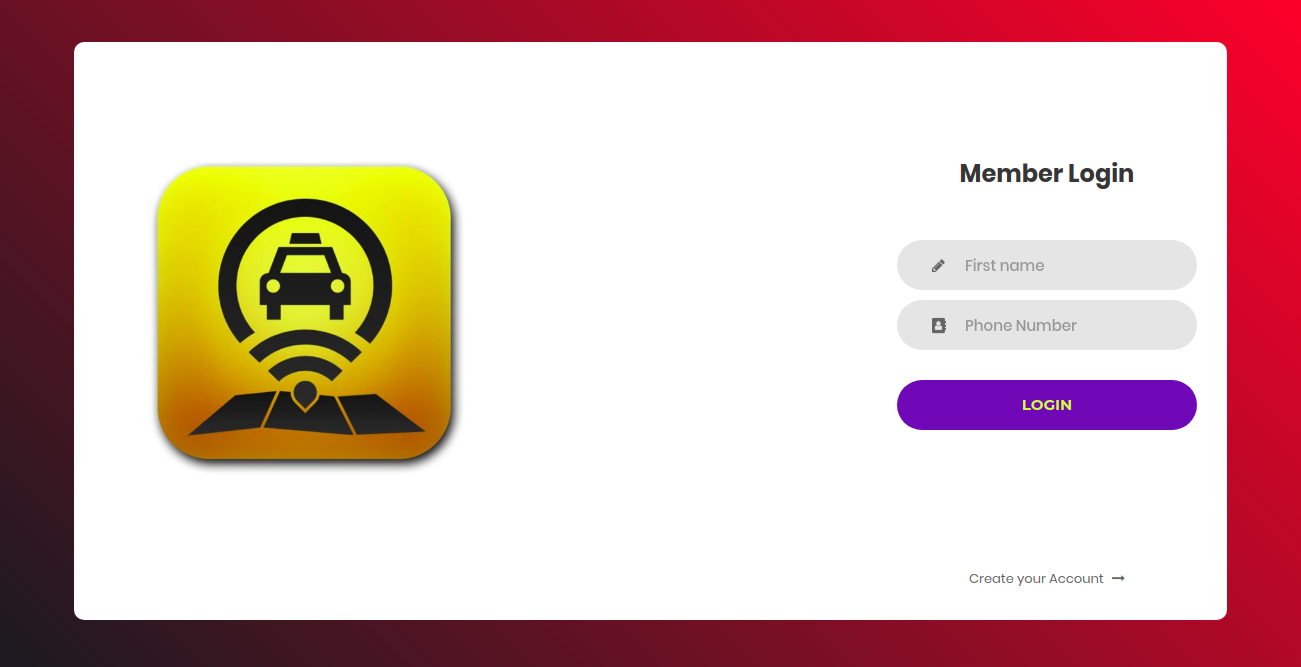
****

FIG 6.1 LOGIN PAGE

2. USER HOMEPAGE

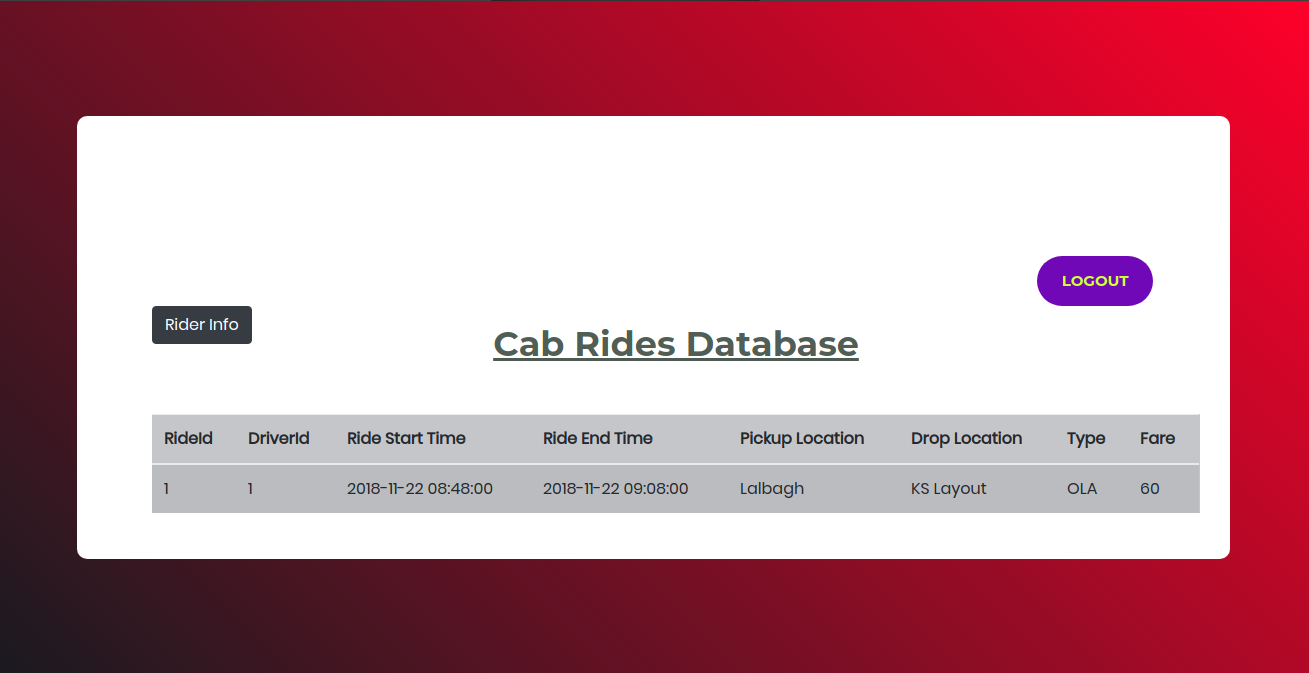
****

FIG 6.2: USER HOMEPAGE

3. RIDER INFO

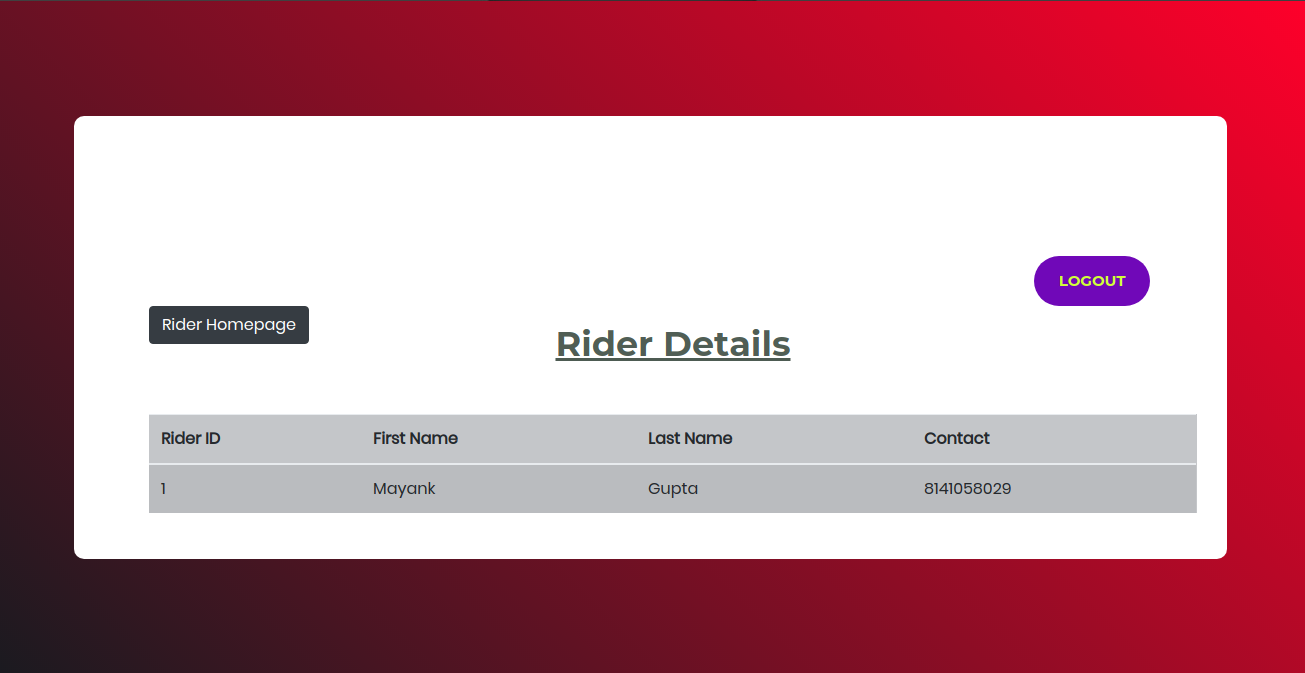
****

FIG 6.3 RIDER INFO

4. ADMIN HOME PAGE

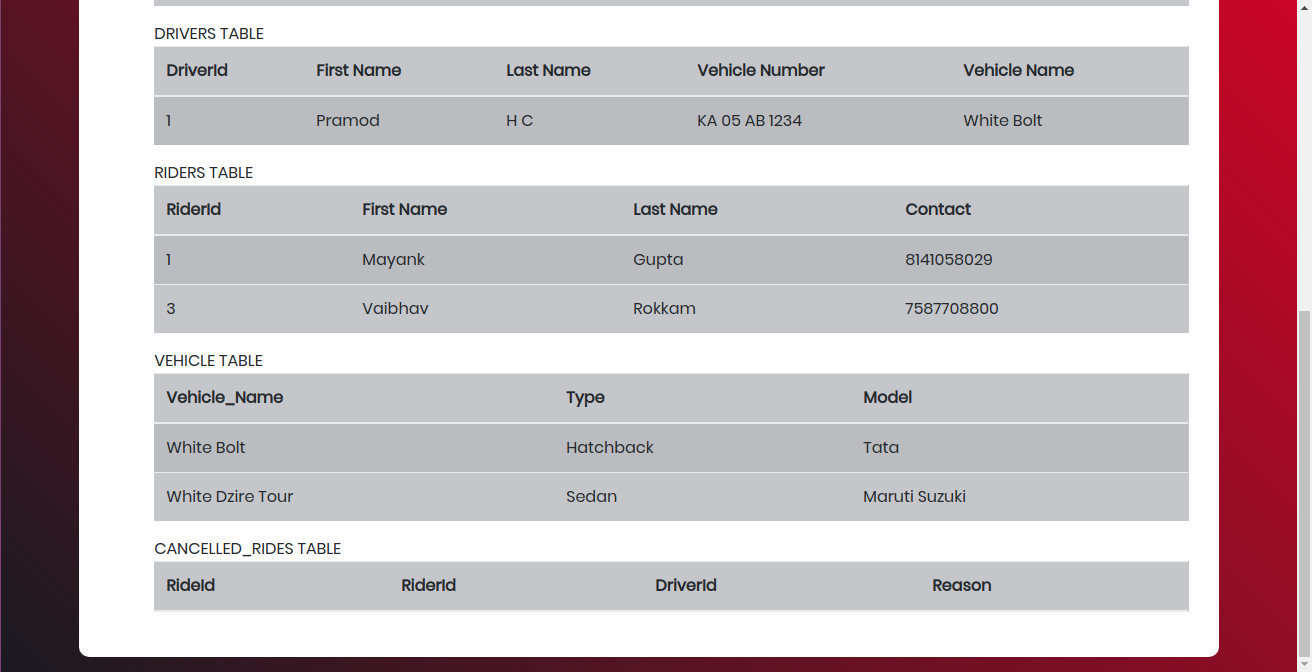
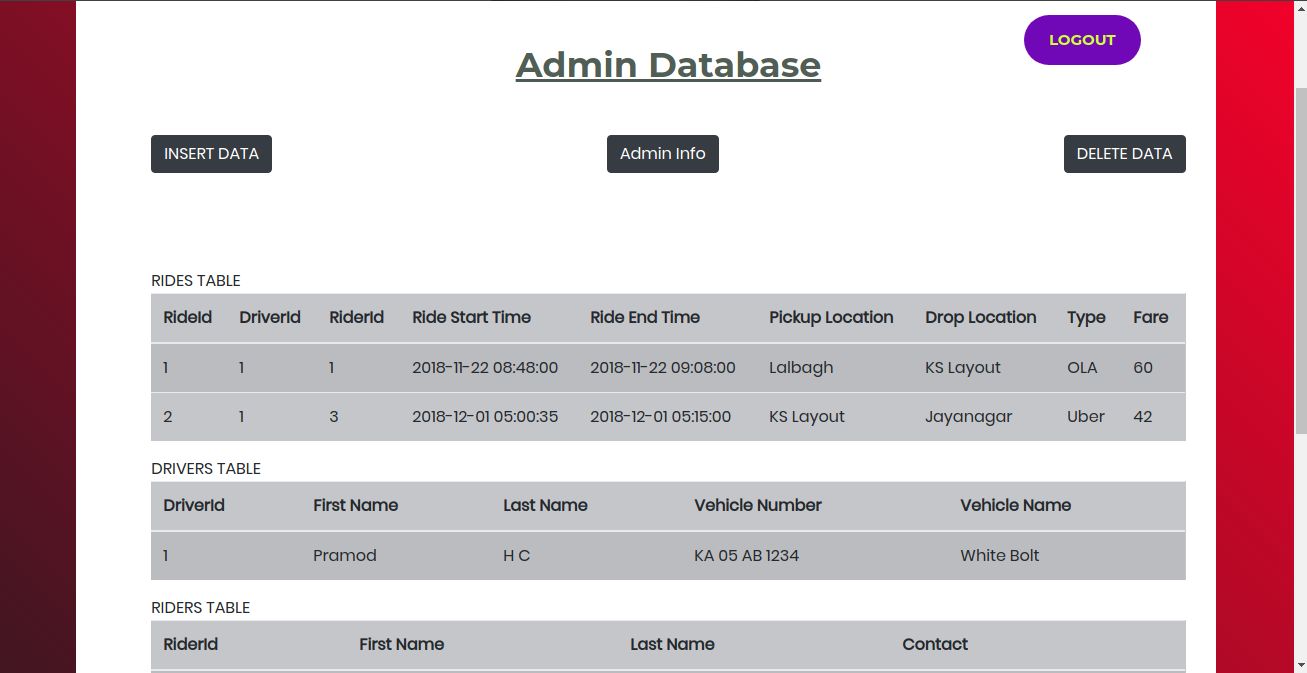
****

FIG 6.4 ADMIN HOMEPAGE

5. INSERT INTO TABLE PAGE

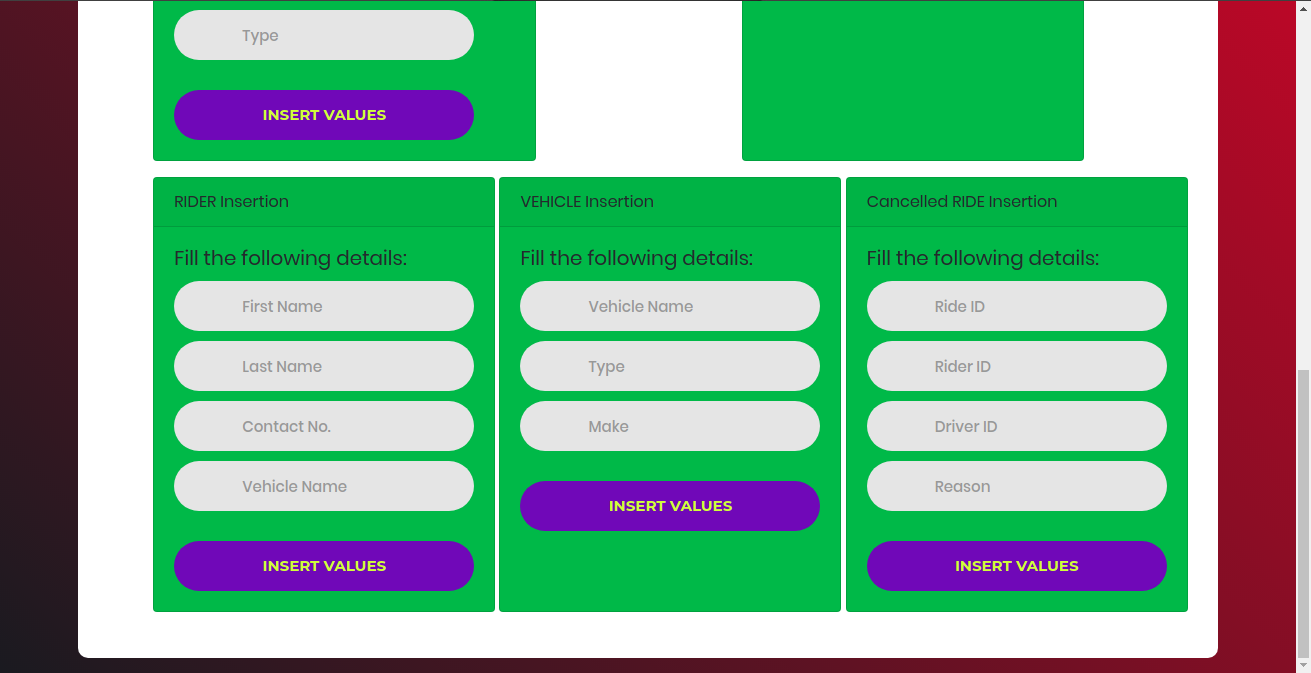
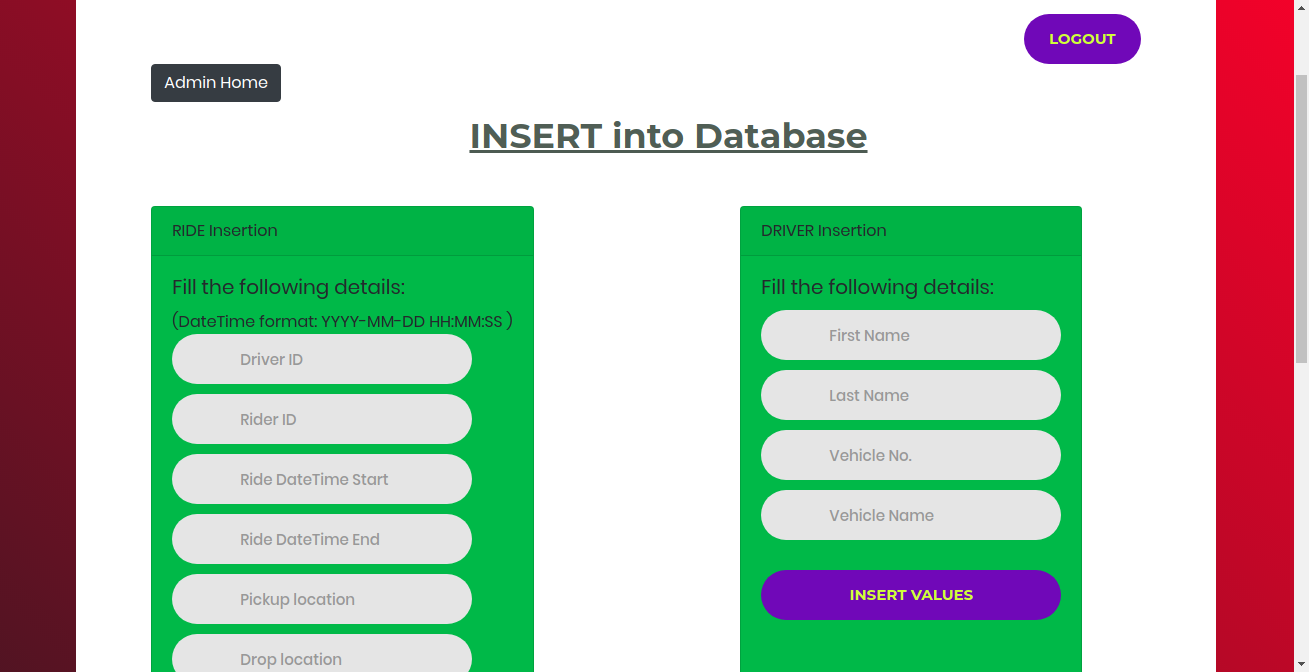
****

FIG 6.5 INSERT INTO TABLE PAGE

6. DELETE FROM TABLE PAGE

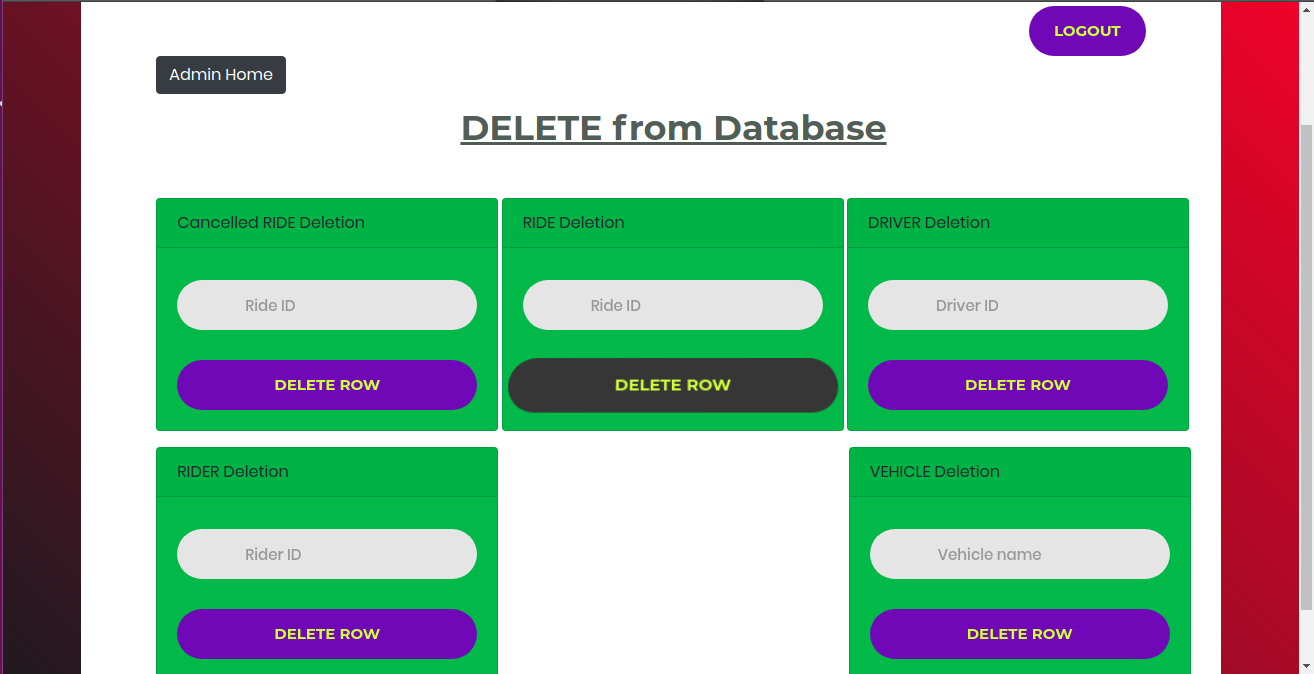


FIG 6.6 DELETE FROM TABLE PAGE

7. SIGNUP PAGE

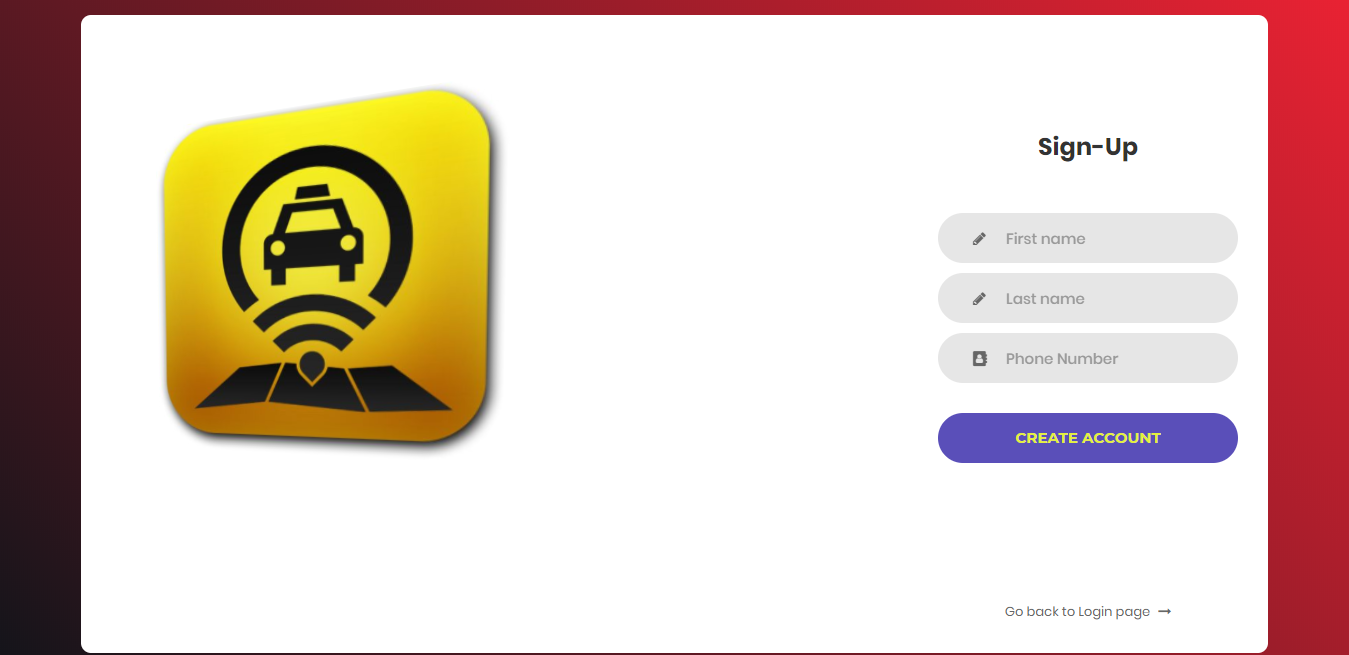


FIG 6.7 SIGNUP PAGE

**CHAPTER 7**

**REFERENCES**

1. Elmasri &Navathe : Fundamental Database System ,7th edition 2017
2. [www.w3schools.com](http://www.w3schools.com)
3. [www.stackoverflow.com](http://www.stackoverflow.com)
4. [MySQL :: MySQL 5.7 Reference Manual :: 13.2.9 SELECT Syntax](https://dev.mysql.com/doc/refman/5.7/en/select.html) dev.mysql.com
5. www.tutorialspoint.com