

Governors State University

OPUS Open Portal to University Scholarship

All Capstone Projects

Student Capstone Projects

Fall 2022

Rent-A-Car

Roosevelt Jackson

Follow this and additional works at: <https://opus.govst.edu/capstones>

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to http://www.govst.edu/Academics/Degree_Programs_and_Certifications/

Visit the [Governors State Computer Science Department](#)

This Capstone Project is brought to you for free and open access by the Student Capstone Projects at OPUS Open Portal to University Scholarship. It has been accepted for inclusion in All Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact opus@govst.edu.

RENT A CAR

By

Roosevelt Jackson

Northeastern Illinois University,
Bachelor's Degree in Computer Science, 2005

GRADUATE CAPSTONE SEMINAR PROJECT

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science



Governors State University
University Park, IL 60484

2022

ABSTRACT

Our goal is to build a data management system for a car rental company. This enables the admin to rent a vehicle to a customer. This system increases customer retention while simplifying vehicle and personnel management. The interface of this software car Rental System is very user-friendly. As a result, users will find it extremely simple to work on. Administrators can use this system to manage customer confirm and cancel booking requests, customer testimonials, and customer issues. The vehicle data can be entered into the system. Administrators can also modify or delete existing vehicle data. There is no delay in the availability of any car information; it can be captured quickly and easily whenever required.

In this application, we had three primary responsibilities: admin, renter, and owner. Renters can locate their vehicles and communicate with the owners, owners can upload vehicle information, and administrators can manage accounts and vehicles in this application. We chose the PHP Codeigniter framework to build our car rental website, which includes HTML5, CSS3, Bootstrap, and Javascript for the user interface and MySql for the database.

Table of Content

| | | |
|-----------|---|----|
| 1 | <i>Project Description</i> | 1 |
| 1.1 | Competitive Information | 1 |
| 1.2 | Relationship to Other Applications/Projects | 1 |
| 1.3 | Assumptions and Dependencies | 1 |
| 1.4 | Future Enhancements | 2 |
| 1.5 | Definitions and Acronyms..... | 2 |
| 2 | <i>Project Technical Description</i> | 2 |
| 2.1 | Application Architecture | 2 |
| 2.2 | Application Information flows | 3 |
| 2.3 | Capabilities | 4 |
| 2.4 | Risk Assessment and Management | 4 |
| 3 | <i>Project Requirements</i> | 5 |
| 3.1 | Identification of Requirements | 5 |
| 3.2 | Operations, Administration, Maintenance, and Provisioning (OAM&P) | 5 |
| 3.3 | Security and Fraud Prevention | 5 |
| 3.4 | Release and Transition Plan | 5 |
| 4 | <i>Project Design Description</i> | 6 |
| 5 | <i>Internal/external Interface Impacts and Specification</i> | 11 |
| 6 | <i>Design Units Impacts</i> | 11 |
| 6.1 | Functional Area A/Design Unit A..... | 11 |
| 6.1.1 | <i>Functional Overview</i> | 11 |
| 6.1.2 | <i>Impacts</i> | 11 |
| 6.1.3 | <i>Requirements</i> | 12 |
| 6.2 | Functional Area B/Design Unit B | 12 |
| 6.2.1 | <i>Functional Overview</i> | 12 |
| 6.2.2 | <i>Requirements</i> | 12 |
| 7 | <i>Open Issues</i> | 12 |
| 8 | <i>Acknowledgements</i> | 12 |
| 9 | <i>References</i> | 13 |
| 10 | <i>Appendices</i> | 13 |

1 Project Description

To hire a vehicle at a fixed price, we created this initiative. All booking activity is done manually, and keeping the information about bookings and vehicles in the current system is quite labor-intensive. Finding out which vehicles are available for reservation requires much time. The procedure becomes more complex and more challenging as a result. The main objective of development is to automatically generate daily reservations, records of vehicles available for booking, recordings of available routes, car rental rates for each route, and customer store data. The RENT A CAR system is a car rental software that offers a solution to your day-to-day car rental business operating demands. Users may manage customer information online with the aid of this solution. Using this method, you may verify your customer information whenever you want. This aids in keeping track of your revenue in a given month or over an entire year. You may make decisions on expanding your business based on this information. This project is designed to be utilized by consumers that hire cars. Customers may see available vehicles, register, view profiles, and reserve vehicles using this online system.

1.1 Competitive Information

The car rental market is fiercely competitive, especially regarding cost and quality. A number of the big car rental firms' recent ownership changes may also have increased competitiveness. Franchisees may face competition from national, regional, and local businesses in any given place, many of which, notably those owned by the big cars, have more financial resources. In a similar vein, the leasing and management of vehicles are both quite competitive. Along with the big fleet management service providers, hundreds of other local, regional, and specialized rivals concentrate on only one or two items. The leading car rental firms have occasionally experienced pricing pressure throughout the whole sector, whether it was due to overcapacity or low demand. A recurrence of oversupply or a significant decrease in overall demand could negatively impact our ability to maintain or raise our rental rates.

1.2 Relationship to Other Applications/Projects

Construct a web-based system that enables customers to register and book cars online while enabling the company to run its vehicle rental operations effectively. To simplify the procedure of renting a car for customers. Based on the kind of vehicles that are hired, most businesses in the market earn a profit. The rental vehicles are divided into four categories: economy, compact premium, compact premium, and luxury. Customers are allowed to select any vehicle of their choosing, subject only to the availability of such a car at the time of reservation and their financial situation.

1.3 Assumptions and Dependencies

- Each reservation is only ever linked to one car reservation at a time.
- At some point, vehicles that are a member of the system should be accessible.
- The rental insurance may or may not be included in the reservation because the renter may have his insurance.

1.4 Future Enhancements

- We intend to rent vehicles daily in the near future so that the consumer may provide the client with their vehicles every day.
- We intend to introduce a new function called "pay after the journey."
- To significantly improve user experience, we aim to boost system automation.

1.5 Definitions and Acronyms

- MVC - Model View Controller
- HTML - Hyper Text Markup Language
- CSS - Cascading Style Sheets
- PHP - Hypertext Preprocessor
- MySQL - My Structured Query Language
- XAMPP - X-operating system, Apache, Mysql, Php, Perl

2 Project Technical Description

The goals of the Rent a Car project must be accomplished via the completion of various research that spans a wide variety of themes. The PHP technology we utilized to construct the application was incorporated within the CodeIgniter framework (Learn Php n.d.). Create a web-based system that will enable customers to register and book cars online and help the organization manage its vehicle rental business successfully. The Tools in this Application were used for the development below.

1. XAMPP: Apache (Application Server) An open-source Java Servlet Container called Apache, sometimes referred to as Server, was created by the Apache Software Foundation (Harwani n.d.).

2. MySQL Server: It handles large databases more quickly than earlier techniques. It includes a multi-threaded SQL server that accommodates a range of back ends, client applications and libraries, administrative tools, and APIs (APIs). MySQL Server's connectivity, speed, and security make it ideal for accessing databases via the Internet (Learn Database n.d.).

3. Visual Studio Code is a potent text editor that can work with prose, markup, and code. You will be impressed by the slick user interface, superb features, and fantastic performance (Visual Studio Code n.d.).

4. Web browsers: We may use any web browser.

2.1 Application Architecture

The tools for online vehicle reservations are offered by Rent a Car. It has a number of the features listed below.

Rent Vehicle Management: It offers an online car reservation service. Customers may check out the website to view a variety of autos. Bookings can be made if they meet the requirements.

Checking Availability: The user can check the car's availability. He or she keeps up the vehicle database. If no cars are available, it indicates that you have already reserved a vehicle.

System of payment: After receiving confirmation of their reservation, renters can only pay for the vehicle.

Vehicle Booking: The consumer must be able to register for a reservation via the system. The technology must enable customers to examine detailed descriptions of specific vehicles. The system must have an advanced search feature that allows users to restrict their vehicle search to particular car search categories. The system must enable clients to choose a specific vehicle from various search categories when making a reservation. During a reservation, the system must inspect a list of available cars. The system notifies all successfully committed reservations. For successfully committed reservations, the system must be able to show the reservation summary.

Log in: The system must permit administrators to log in using their username and password. Users should be able to access the system by entering their usernames and password.

Owner: The system needs to provide free posting of new vehicles. The system must enable personnel to search for vehicles using a particular record. The system must enable updating of information about the vehicle that requires change. The system must show all available vehicles.

Renter: The system must let the renter choose from available vehicles. The technology will enable surveys and vehicle input. The technology displays the address on a Google Map for a particular car.

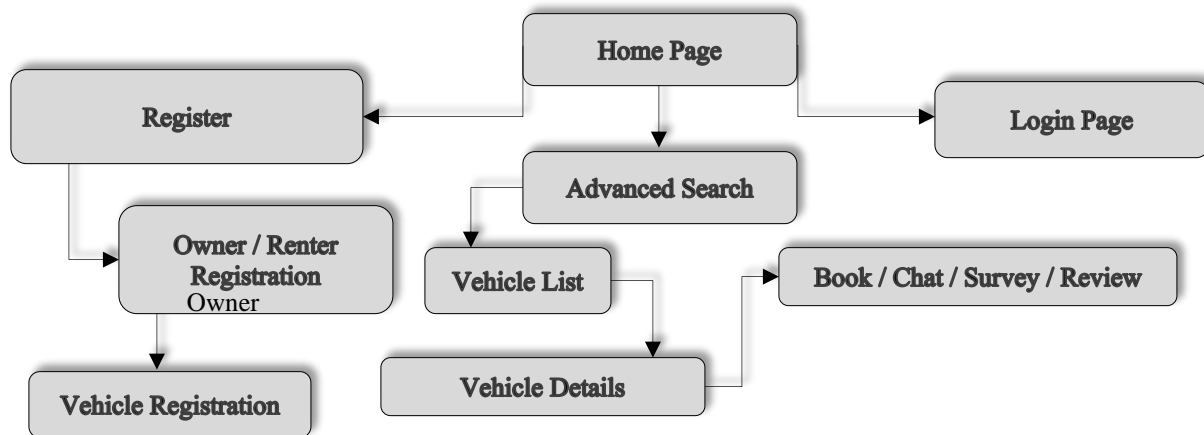


Fig1 Application Architecture

2.2 Application Information flows

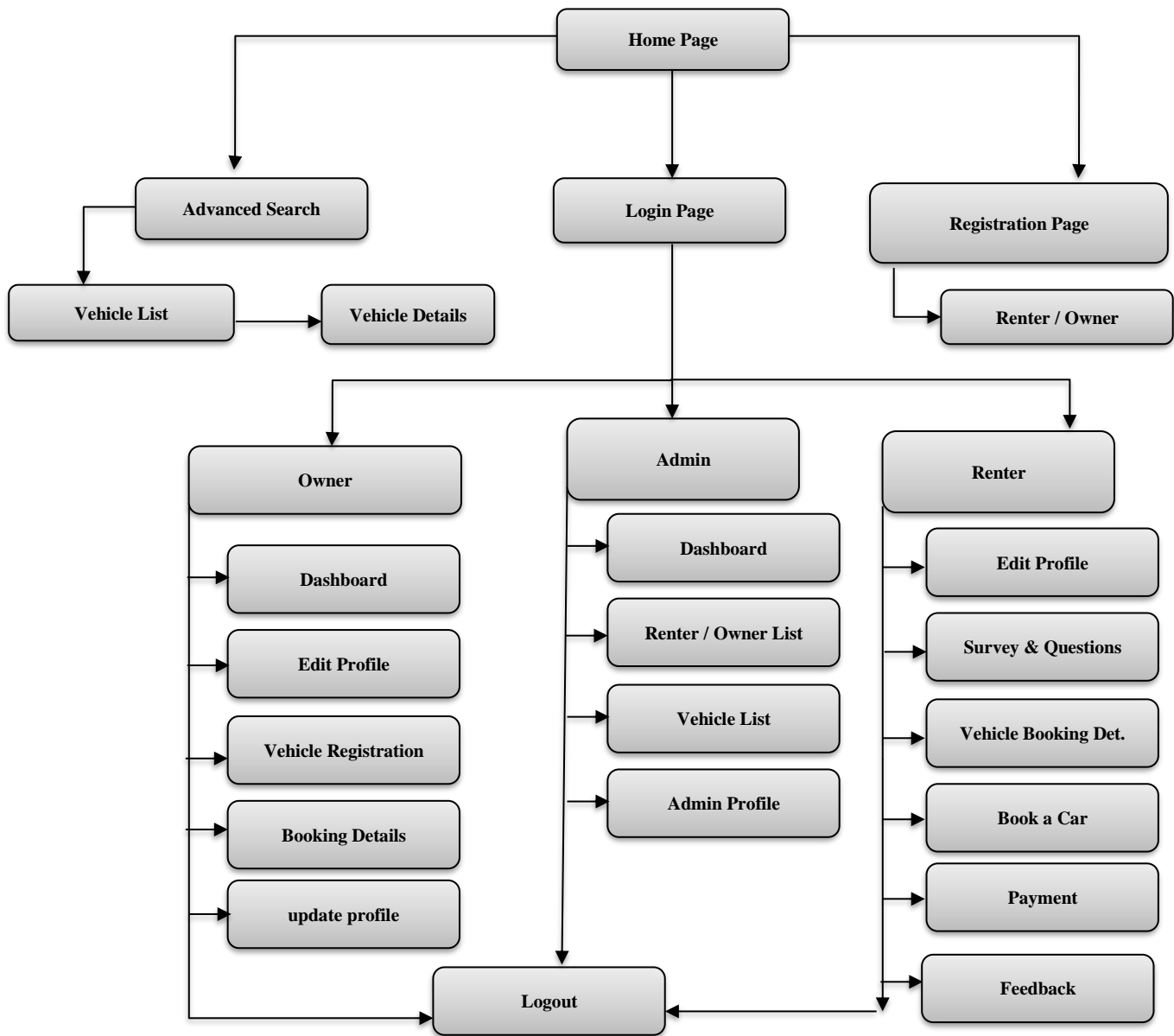


Fig 2 WorkFlow

2.3 Capabilities

The renter reserves the vehicles. Renter and owner data is kept in a database. Information on the renter, including personal data, travel arrangements, and the status of the vehicle's availability, is kept on file. Communication between owners and renters is allowed.

2.4 Risk Assessment and Management

The types of hazards connected to the project have been listed. Identifying and ranking these hazards have helped us understand that each risk has a different level of potential for damage and a different frequency of occurrence. In this instance, we have dealt with the risk that has a tremendous potential to hurt first to control the risks associated with each strategy.

3 *Project Requirements*

3.1 *Identification of Requirements*

<GSU-RC_CPSC8985_V1 User- 1>

The user should have the option to register as a Renter.

Implementation: The user should be able to register as a renter by providing the required information, such as their first and last names, email addresses, passwords, and confirmed passwords.

<GSU-RC_CPSC8985_V2 Owner- 2>

Implementation: The owner can add, amend, or remove the car.

<GSU-RC_CPSC8985_V3 admin- 3>

Implementation: Only the owner, renter, and cars can be active or inactive for the administrator.

3.2 *Operations, Administration, Maintenance, and Provisioning (OAM&P)*

Operations

- Car-booking websites should be simple to use and secure to operate.
- The website must be dependable and user-friendly.
- The program enables legitimate, signed-up users to access safe transactions.
- Access to the website should need both registration and login.
- The search process needs to go quickly and realistically.
- The search option's results must be accurate.

Administration

- The admin can authorize renter and owner access to the website.

Maintenance

- The program has to be kept up and protected.
- The program has to be secure and up to date.
- The website database provides data security, backup, and fault recovery procedures.

Provisioning

- The program ensures a trustworthy configuration
- To make the program user-friendly, the system displays the workflow that the user directs.

3.3 *Security and Fraud Prevention*

The user may confidently upload their car with photographs and vehicle details in the online application because users cannot see the data until they have registered for our program. Additionally, the registration process is supported by an email verification procedure, which enhances the security of our application. Even so, the admin can promptly disable/delete the post or the user if any fraudulent activity occurs.

3.4 *Release and Transition Plan*

Right now, we are utilizing XAMPP to deploy directly on local workstations (Friends, Apache 2017). However, in the near future, we intend to put it on a server that anybody on the globe may access.

4 Project Design Description

Rent-a-Car is created with the CodeIgniter (Learning Codeigniter n.d.) Framework and has a user-friendly interface that aids users in selecting their automobiles so they may enjoy their travels. Three user roles — Owner, Renter, and admin—comprise Rent-a-Car entirely. Users' and owners' uploaded cars are subject to the admin's control.

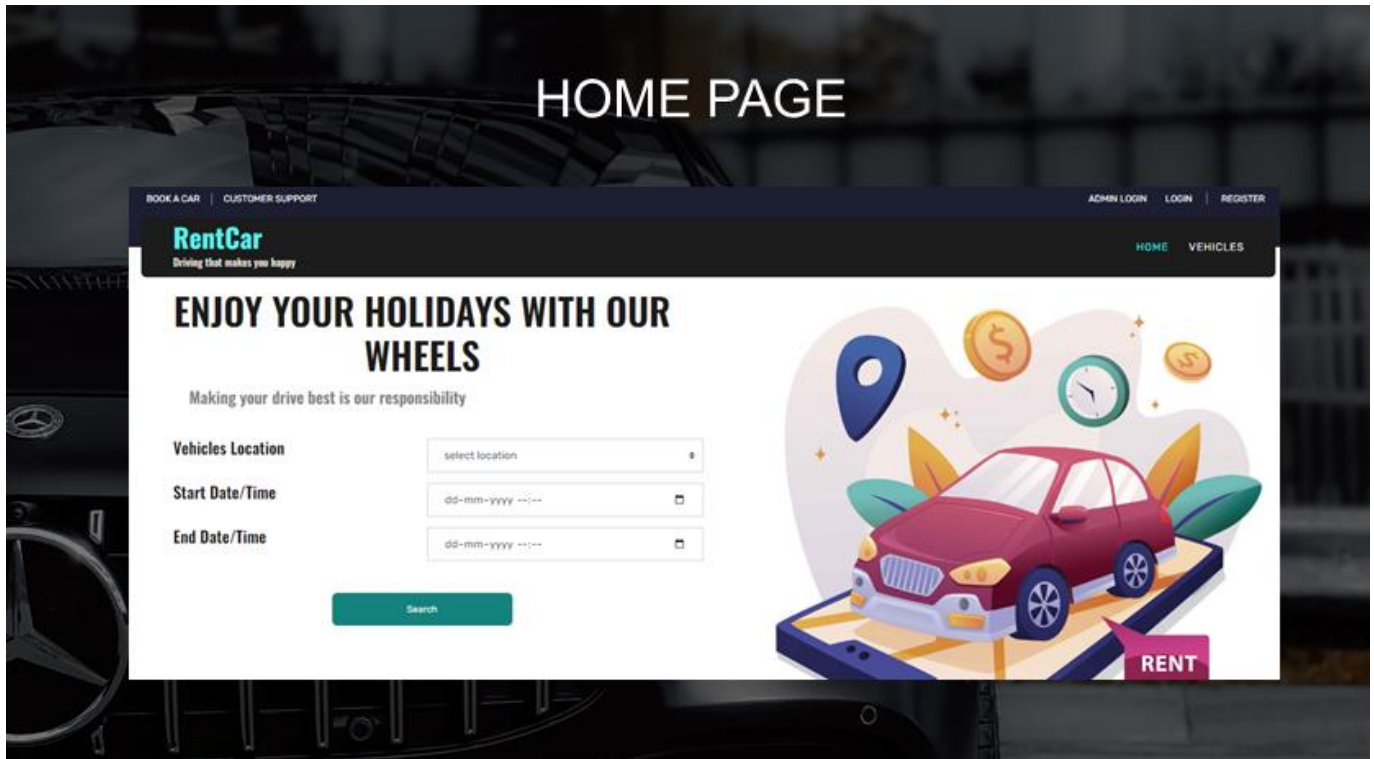


Fig 3 Home Page

The first page of our application is the home page. On this page, we may search for the cars based on their starting and ending points. On this page, we have listed the cars that were highlighted. When a user selects any of the highlighted vehicles, a page with all the facts about that particular vehicle is brought up. The availability of the vehicle on this specific day may be seen on the vehicle information page. Moreover, we could use Google Maps to locate the place. In addition, we may also see information on the owners of specific vehicles and customer reviews.

LOGIN TO RENTER / OWNER




Illustration of a person in a blue shirt and orange pants working on an orange car. The car is on a platform, and there are industrial buildings in the background.

User Name

User Password

User Type

DON'T HAVE AN ACCOUNT?

Fig 4 User Login Page

WELCOME TO ADMIN DASHBOARD

LOGIN TO RENTCAR



Illustration of a red car on a smartphone screen. The screen shows a 'RENT' button. Above the car are icons for a location pin, a clock, and coins.

User Name

User Password

Fig 5 Admin Login Page

For this application, we utilized two login pages: one for admin login and the other for renter and owner login. Only the admin may log in to their page. Owner or Renter logs in using their user login page. The admin can access the admin dashboard after successfully entering their credentials. They manage only the renter, owner, and cars. Users and cars may be activated or deactivated by the administrator. Admin can

edit their profile using the settings menu. We can see on the admin dashboard how many renters, owners, and vehicles are available across the entire application.

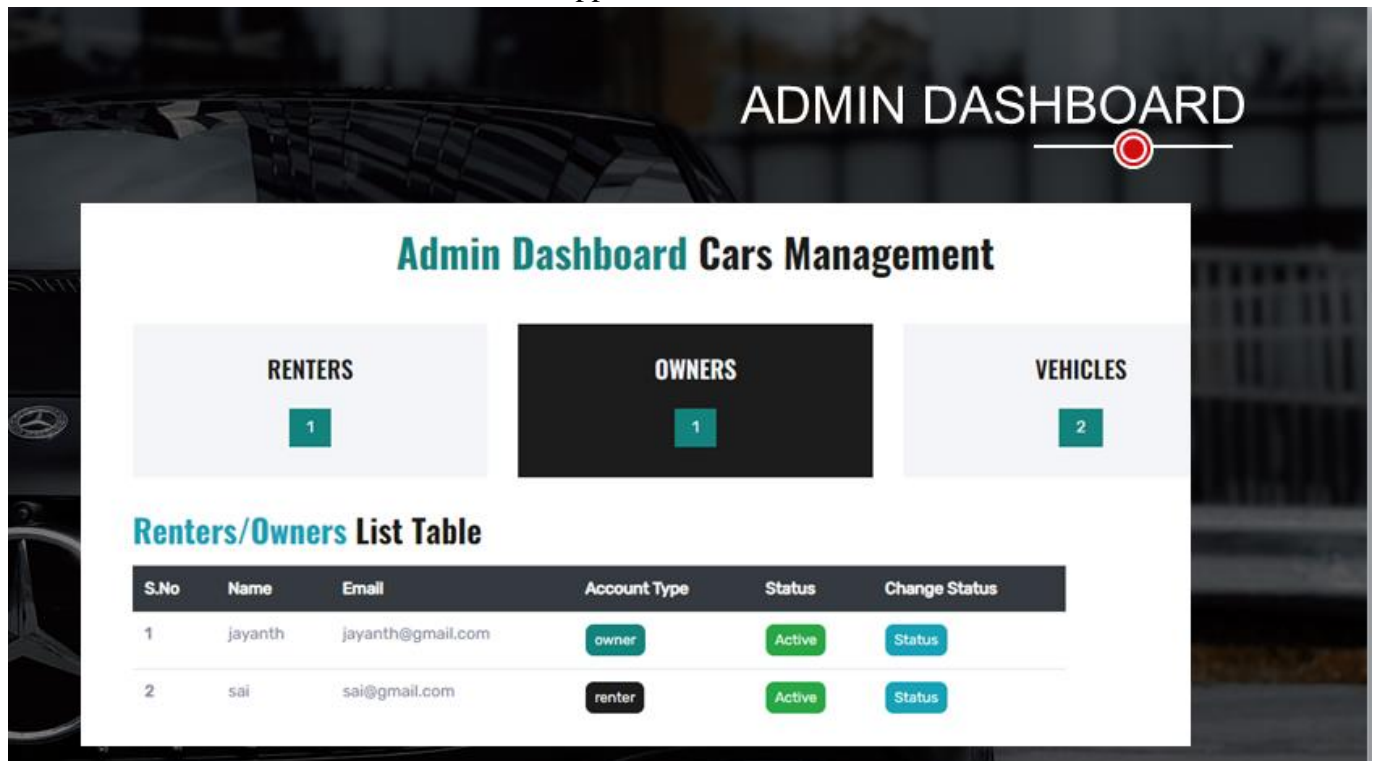


Fig 6 Admin Dashboard

The owner dashboard is redirected when the owner enters their correct credentials on the login page. They can amend their personal information using the edit profile option on the owner dashboard. Only the owner posts the vehicles, and the admin must approve them. The owner sets the cost of the car on an hourly, daily, or monthly basis. The owner dashboard also displays a list of vehicles. The owner wants to talk with the renter to discuss using the chat option.

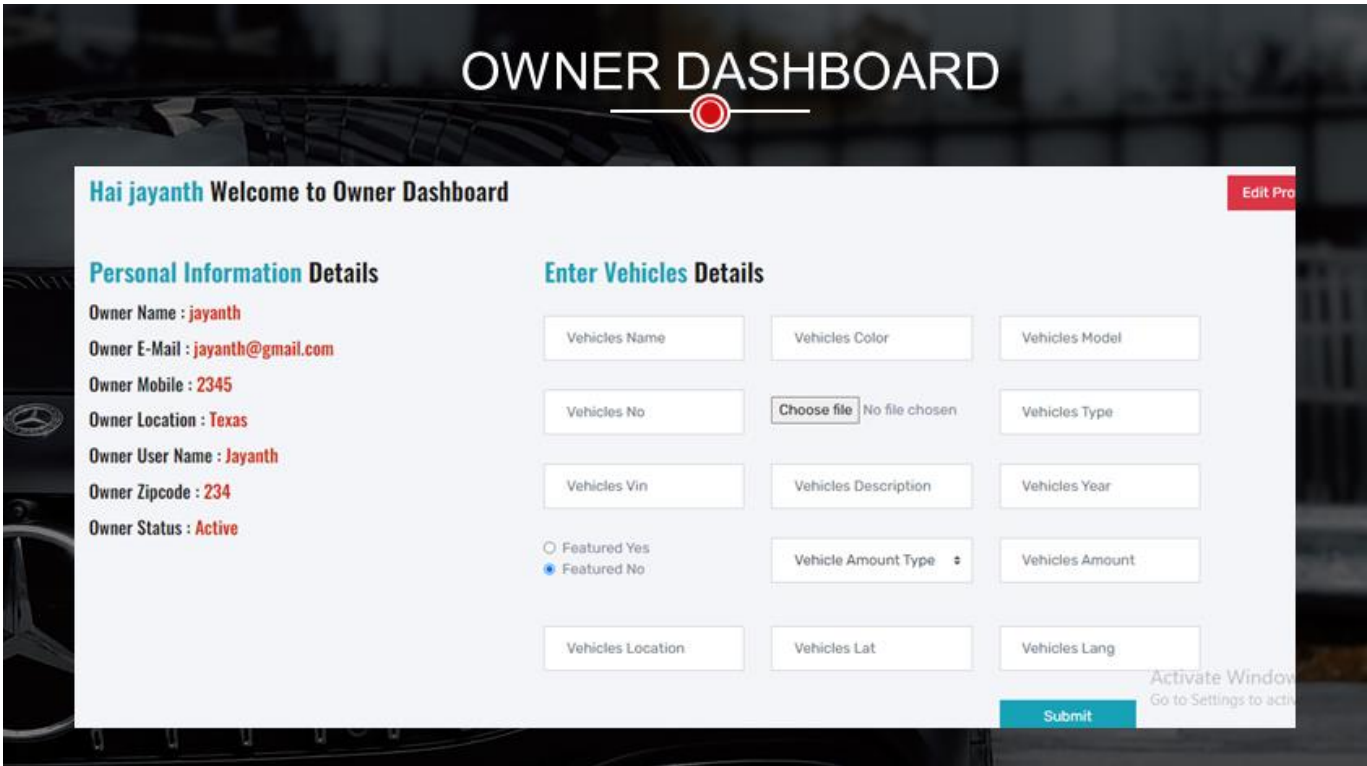


Fig 7 Owner Dashboard

The renter dashboard is redirected when they use the correct login information on the login page. Utilizing the edit profile feature on the renter dashboard, they may change their personal information. Only the renter may reserve vehicles and examine information about the reservations. They can also leave feedback comments and surveys for the vehicles and the whole renting experience. The owner and renter can both interact with one another via a chat feature.

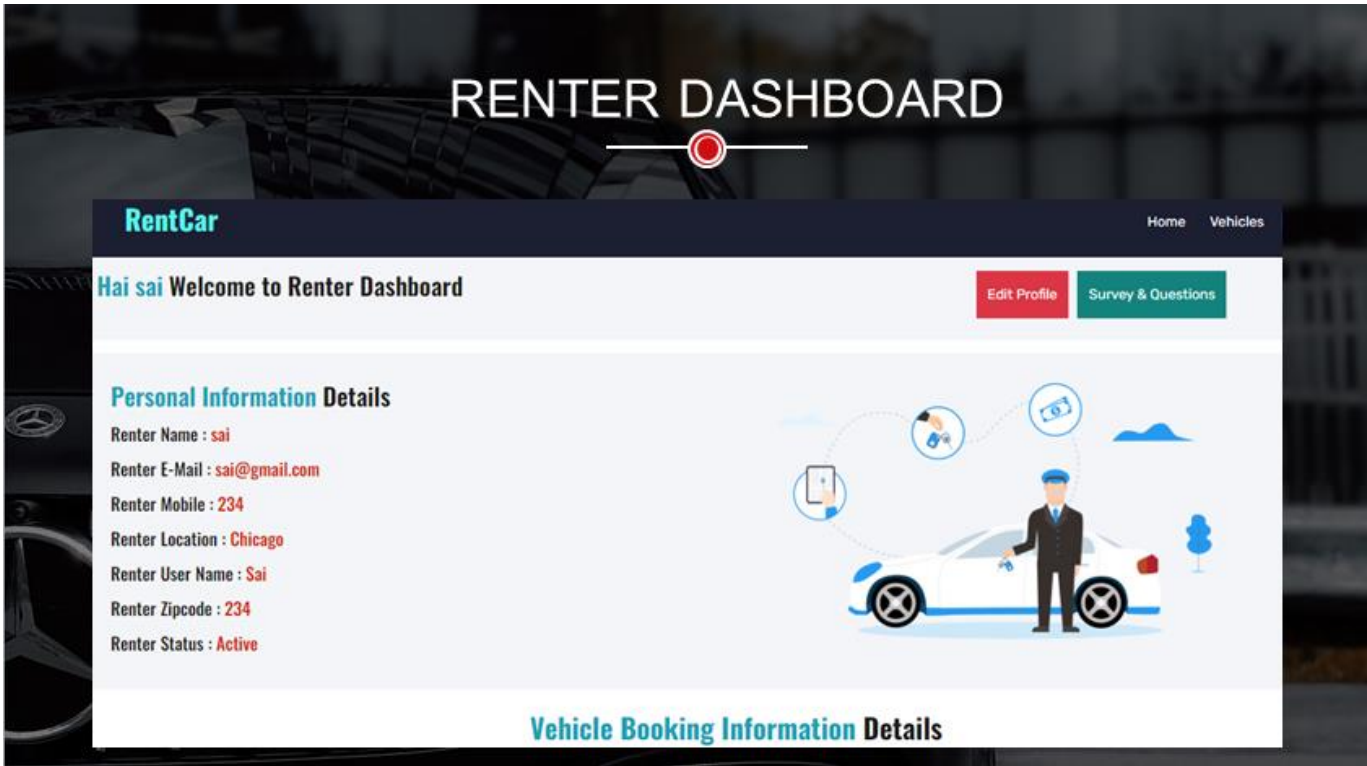


Fig 8 Renter Dashboard

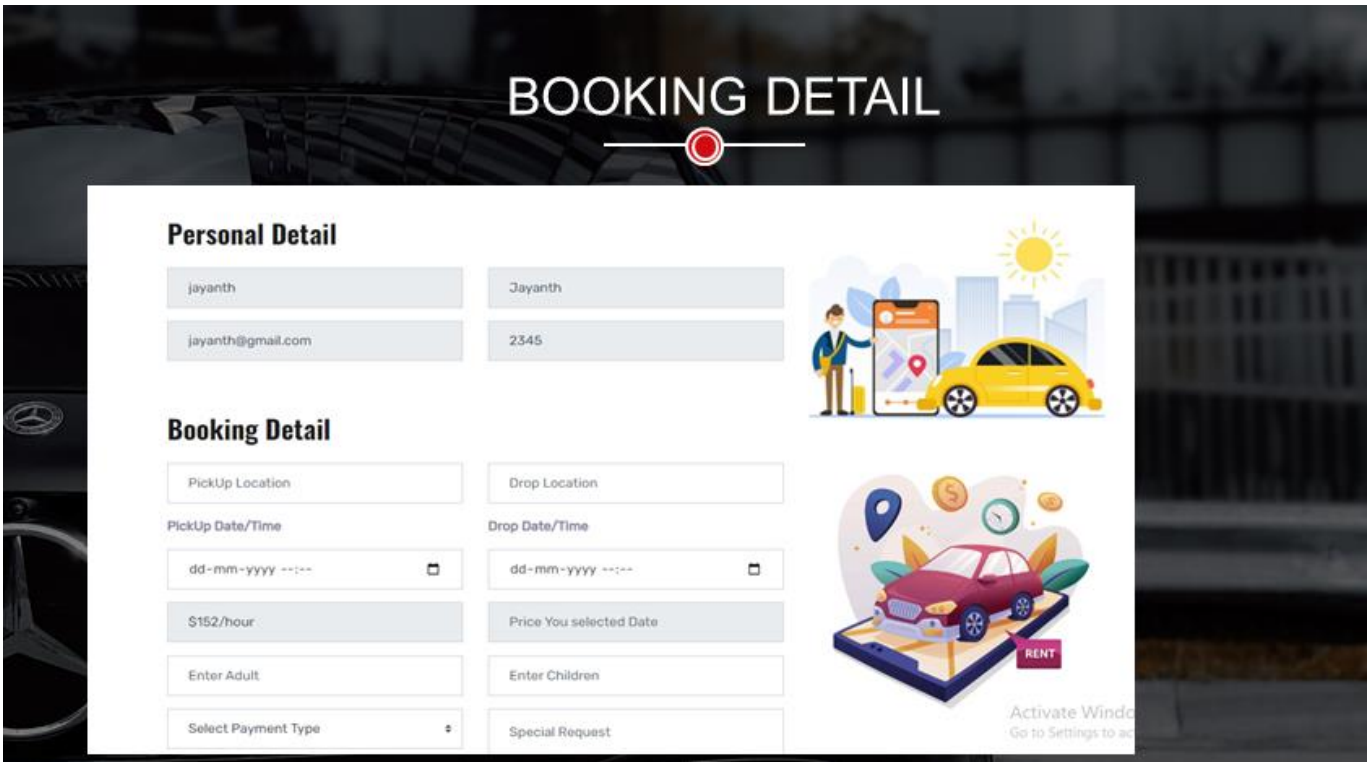


Fig 9 Vehicle Booking Detail

5 *Internal/external Interface Impacts and Specification*

The internal interfaces of renting a car allow us to govern both endpoints of the created connection. If something goes wrong, we can immediately examine all the parts and infrastructure involved in the execution of a module, troubleshoot the underlying issue, work on a long-term solution, and ensure that it does not happen again. Most of the time, your web server's internal interfaces are with your application server's configuration property files, SQL databases, or caches. External interfaces, on the other hand, limit our influence to a single connecting point. In cases when third-party plugins or email servers are used, the code might malfunction if the email server is not responding. If the email server stops responding at this stage, the problem occurs at that moment and prevents the registered user from using the program. The email has to be validated at the time of registration in order to login into the application. The problem with using the exterior point of connection is that you frequently function without a set administration level and have to deal with obtuse accessibility or execution. Having excellent and direct private communication with the other connection point administrators also seems acceptable so that, should the need arise, they may be persuaded to implement sensible modifications on their end. Furthermore, external connection points may change or disappear at times beyond your control or even be obfuscated by you.

6 *Design Units Impacts*

Code Igniter Framework is used to create the rent-a-car application in a user-friendly way. We used an MVC architecture that adheres to the Composite, Observable, and Strategy design principles.

6.1 *Functional Area A/Design Unit A*

6.1.1 *Functional Overview*

The process of establishing the criteria or conditions that must be met for a new or updated product while considering the possibility of conflicting requirements from various users is known as requirement analysis. It is a software engineering method. Functional requirements are used to illustrate the system's internal operation and describe and explain each subsystem inside the system. It consists of the job that the system must do, the associated processes, the necessary data, and the user interfaces. After the vehicle rental system is finished, a requirement analysis will be performed using a software engineering approach to identify the requirements and conditions for the system to function effectively. All of the potentially incompatible requirements of entities with various dimensions must be considered in this evaluation. Functional requirements show how the underlying system design functions to achieve the overall system goals. Along with the core vehicle rental system, this also shows its supporting systems and provides details of each step. This includes the duties that systems must carry out, the data that systems must store, the procedures involved, and the appropriate user interface.

6.1.2 *Impacts*

The dynamic front-end design of Rent a Car makes it user-friendly and enables users to discover the desired results by utilizing UI filters. Since there is very little database latency, web pages load quickly, allowing users to utilize the program without experiencing response delays. A framework is a collection

of specified ideas, methods, and standards for handling a certain kind of issue that may be used as a manual for tackling and resolving related issues in the future.

6.1.3 Requirements

1. Online user registration for new users should be possible.
2. Online car reservations - Users of the system should be able to book and reserve cars online.
3. The system should automatically update the database whenever a new reservation is made, or a new client is registered.

6.2 Functional Area B/Design Unit B

6.2.1 Functional Overview

Admin has access to both owner and renter management. The administrator can also control the vehicles that the owner posts. The content presented throughout the program is within the admin's control. The application's administrator may shield it against harmful information, fake vehicles, and user accounts.

6.2.2 Requirements

Hardware Requirements:

Processor: Intel Pentium Dual Core

RAM: 512 MB

Hard Disk: 160 GB Space

Software Requirements:

Operating System: Windows /iOS/Unix

Web Browser: IE/Google Chrome/Firefox

Framework Technology: Code Igniter Framework

Tools: XAMPP

Web Design: HTML, CSS, JAVASCRIPT

Back End: MYSQL

Scripting Language: PHP

7 Open Issues

Currently, it is nearly difficult for any user to log into the application if they forget their password. Because of this, we want to introduce the "Forgot Password" function, which enables users to change their passwords and keep using the application without encountering any problems.

8 Acknowledgments

The Almighty God deserves all the praise for His mercies, unwavering goodness, and abounding love throughout my life. Professor Xin Chen has my sincere gratitude for her guidance and assistance throughout the project development. My teammates helped me accomplish the assignment. Therefore I want to thank them for their contributions.

9 References

Friends, Apache. 2017. <https://codebriefly.com/how-to-setup-apache-php-mysql-on-windows-10/>.

Harwani, Bintu. *Installing XAMPP*. n.d. <https://www.ionos.com/digitalguide/server/tools/xampp-tutorial-create-your-own-local-test-server/> (accessed 11 30, 2022).

Learn Database. n.d. https://www.w3schools.com/php/php_mysql_create.asp.

Learn Php. n.d. https://www.w3schools.com/php/php_forms.asp.

Learning Codeigniter. n.d. https://www.tutorialspoint.com/codeigniter/installing_codeigniter.htm.

Visual Studio Code. n.d. <https://adamtheautomator.com/visual-studio-code-tutorial/>.

10 Appendices

```
<?php
class Admin_model extends CI_model{

    function __construct(){
        parent::__construct();
    }

    function loginuser($user,$pass,$table){
        $query = $this->db->query("SELECT * FROM $table WHERE user = '". $user. "' AND password_encode='".
        $result = $query->result_array();
        return $result;
    }
    function checkUserloginstatus($user,$pass,$table){
        $query = $this->db->query("SELECT * FROM $table WHERE user = '". $user. "' AND password_encode='".
        $result = $query->result_array();
        return $result;
    }
    function checkUser($checkarray,$table){

        $query = $this->db->query("SELECT * FROM $table WHERE email = '". $checkarray['email']. "'");
        $result = $query->result_array();
        return $result;
    }
    function create_car_account($insert_array,$table){

        $this->db->insert($table, $insert_array);
        return true;
    }
}
```

Fig 10 Admin Model

```

function loginuser($user,$pass,$type){
    $table='carflexi_account';
    //echo "SELECT * FROM $table WHERE user_name = '$user.'" AND password='$pass.'" AND type='";
    $query = $this->db->query("SELECT * FROM $table WHERE user_name = '$user.'" AND password='$pass.'" AND type='";
    $result = $query->result_array();
    return $result;
}
function checkUserloginstatus($user,$pass,$type){
    $table='carflexi_account';
    $query = $this->db->query("SELECT * FROM $table WHERE user_name = '$user.'" AND password='$pass.'" AND type='";
    $result = $query->result_array();
    return $result;
}
function checkUser($checkarray,$table){
    $query = $this->db->query("SELECT * FROM $table WHERE email = '$checkarray['email'].'" AND ty
    $result = $query->result_array();
    return $result;
}
function checkUserCar($checkarray,$table){
    $query = $this->db->query("SELECT * FROM $table WHERE email = '$checkarray['email'].'" AND ty
    $result = $query->result_array();
    return $result;
}

```

Fig 11 User Model

```

public function login(){

    $data=array();

    $username = $this->input->post('username');
    $password = md5($this->input->post('password'));
    $logintype = $this->input->post('logintype');

    $table='admin_login';
    if($logintype!='login')
    {
        $this->load->view('admin/admin_login');
    }else{

        $checkUserlogin = $this->admin_model->loginuser($username,$password,$table);

        $data['userlogin_details']=$checkUserlogin;
        if(count($checkUserlogin) > 0){
            $user_data = array(
                'user'=> $checkUserlogin[0]['user'],
                'userid'=> $checkUserlogin[0]['id'],
                'logged_in'=> TRUE,
                'email' => $checkUserlogin[0]['email'],
                'type'=>'Admin'
            );
            $this->session->set_userdata('UserAdminlogin', $user_data);
            $data_result = array("result"=>true,"message"=>"done","url"=>base_url('admin/dashboard'));
            echo json_encode($data_result);
        }
    }
}

```

Fig 12 Admin Controller

```

public function index(){

    $data=array();
    $data['dashboardlist'] = $this->admin_model->getAllDatalist();
    $this->load->view('dashboard',$data);
}
public function vehicles(){

    $data=array();
    $data['vehilces_list'] = $this->admin_model->getVehiclesList();
    $this->load->view('vehicles',$data);
}
public function vehicles_view($id){
    $data['vehilcesDetails_list'] = $this->admin_model->getVehiclesList($id);
    $data['vehilces_feddbackList'] = $this->admin_model->getVehiclesFeedaBack($id);
    $data['checkavailable'] = $this->admin_model->checkAvailability($id);
    $data['getLocations'] = $this->admin_model->getLocations();
    $data['vid']=$id;
    //print_r($data['vehilces_feddbackList']);die;
    $this->load->view('vehicles_details',$data);
}
public function search(){

    $data=array();

```

Fig 13 Dashboard Controller

```

$data=array();

$username = $this->input->post('username');
$password = md5($this->input->post('password'));
$logintype = $this->input->post('logintype');
$usertype = $this->input->post('usertype');

$table='carflexi_account';
if($logintype!='login')
{
    $this->load->view('users/login',$data);
}else if($this->input->post('usertype')=='' ){
    $data_result = array("result"=>false,"message"=>"typeerror","url"=>'');
    echo json_encode($data_result);
}else{

    $checkUserlogin = $this->users_model->loginuser($username,$password,$usertype);

    $data['userlogin_details']=$checkUserlogin;
    if(count($checkUserlogin) > 0){
        $user_data = array(
            'name'=> $checkUserlogin[0]['name'],
            'email' => $checkUserlogin[0]['email'],
            'mobile'=> $checkUserlogin[0]['mobile'],
            'user_name'=> $checkUserlogin[0]['user_name'],
            'type'=> $checkUserlogin[0]['type'],

```

Fig 14 User Controller

```

$caruser_session_data = $this->session->userdata('Userlogin');
//print_r($caruser_session_data);?>

<!-- Detail Start -->
<div class="container-fluid pt-5">

    <div class="container pt-5 pb-3">
        <h1 class=" text-uppercase text-danger">make your journey safer and more pleasant </h1>

        <h1 class="display-4 text-uppercase mb-5"><?php echo $vehilcesDetails_list[0]['vehicle_name']>
        <a href="<?php echo base_url('users/dashboard/'. $vehilcesDetails_list[0]['id'].'');?>" clas

        <div class="row align-items-center pb-2">
            <div class="col-lg-5 mb-4">
                
            </div>
            <div class="col-lg-7 mb-4">
                <h2 class="mb-4">Car Information Detail</h2>

                <h4 class="mb-2"><?php echo $vehilcesDetails_list[0]['vehicle_rentamt']; ?></?php echo $vehilcesDetails_list[0]['vehicle_rentamt']; ?></h4>
                <div class="d-flex mb-3">
                    <h6 class="mr-2">Rating:</h6>
                    <div class="d-flex align-items-center justify-content-center mb-1">
                        <small class="fa fa-star text-primary mr-1"></small>
                        <small class="fa fa-star text-primary mr-1"></small>
                        <small class="fa fa-star text-primary mr-1"></small>
                        <small class="fa fa-star text-primary mr-1"></small>
                        <small class="fa fa-star-half-alt text-primary mr-1"></small>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

Fig 15 Car Booking View Code

```


<div class="row">
    <div class="col-lg-10 col-md-6 mb-2">
      <h3 class="text-info mb-3">Hai <?php echo $aruser_session_data['name'];?>
      <span class="text-secondary">Welcome to Owner Dashboard</span></h3>
    </div>
    <div class="col-lg-2 col-md-6 mb-2">
      <a class="btn btn-danger btn-md px-3" href="<?php echo base_url('users/profile/'. $userDetails[0]
    </div>
  </div>
</div>



<div class="row">
    <div class="col-lg-12 col-md-6 mb-2 form-group">
      <div class="row">
        <div class="col-lg-4 col-md-6 mb-2 form-group">
          <h3 class="text-info mb-3">Personal Information <span class="text-secondary">Details</span>
          <h5 class="mb-3">Owner Name : <span style="color:#d12509"><?php echo $userDetails[0]
          <h5 class="mb-3">Owner E-Mail : <span style="color:#d12509"><?php echo $userDetails[
          <h5 class="mb-3">Owner Mobile : <span style="color:#d12509"><?php echo $userDetails[
          <h5 class="mb-3">Owner Location : <span style="color:#d12509"><?php echo $userDetail
          <h5 class="mb-3">Owner User Name : <span style="color:#d12509"><?php echo $userDetai
          <h5 class="mb-3">Owner Zipcode : <span style="color:#d12509"><?php echo $userDetails
          <h5 class="mb-3">Owner Status : <span style="color:#d12509"><?php echo ($userDetails
        </div>
      </div>
    </div>
  </div>


```

Fig 16 Owner Dashboard View

```
application > views > vehicles_details.php
1  <?php defined('BASEPATH') OR exit('No direct script access allowed');?>
2  <?php $this->load->view('header');
3  $caruser_session_data = $this->session->userdata('Userlogin');
4
5  $lang = $vehilcesDetails_list[0]['vehicles_lang'];
6  $lat = $vehilcesDetails_list[0]['vehicles_lat'];
7
8  // print_r($getLocations['pickup_location']);die;
9  ?>
10
11 <?php $carname = $vehilcesDetails_list[0]['vehicle_name'].' '.$vehilcesDetails_list[0]['vehicle_model'];?>
12 <!-- Detail Start -->
13 <div class="container-fluid pt-5">
14     <div class="container pt-5">
15         <div class="row">
16             <div class="col-lg-8 mb-1">
17                 <h1 class="display-4 text-uppercase "><?php echo $vehilcesDetails_list[0]['vehicle_name']
18                 <div class="row mx-n2 mb-3">
19                     <div class="col-md-3 col-6 px-2 pb-2">
20                         Vehicles Description</h3>
25
26             <p><?php echo $vehilcesDetails_list[0]['vehicle_description'];?></p>
27             <div class="row pt-2">
28                 <div class="col-md-3 col-6 mb-2">
29                     <i class="fa fa-car text-primary mr-2"></i>
```

Fig 17 Vehicle Details View