ONLINE PEER-TO-PEER CAR RENTAL SYSTEM

PROJECT REPORT 2025-2026

Submitted By

24500430 SHALINI G

24590071 DHEIVANAI E

24590073 INDIRA V

24590075 MEGALA V

Under the guidance of

Mrs.N.MOIDHEEN BEE M.Sc.,B.Ed

LECT-COMPUTER

Diploma in Information Technology

of the directorate of Technical Education, Government of Tamil Nadu

DEPARTMENT OF INFORMATION TECHNOLOGY

AKT MEMORIAL POLYTECHNIC COLLEGE

KALLAKURICHI-606202.

AKT MEMORIAL POLYTECHNIC COLLEGE

KALLAKURICHI-606202

Department of Information Technology

BONAFIDE CERTIFICATE

This is certified that this project work entitled **Online Peer-To-Peer Car Rental System** has been submitted **SHALINI G, DHEIVANAI E, INDIRA V, MEGALA V** in the partial fulfilment of the requirements for the award of Diploma in Information Technology during the academic year 2025-2026, who carried out the project work under our supervision.

Project Guide	**Head of the Department**
Mrs.N.MOIDHEEN BEE M.Sc.,B.Ed	**Mrs. C.SRIDEVI B.E.,(CSE)**
LECT-COMPUTER	**HOD-COMPUTER & IT**

This is to certify that_____ was examined for the project work viva-voce held on_____

INTERNAL EXAMINER EXTERNAL EXAMINER

Acknowledgement

We express our sincere and profound thanks to our chairman **Mr.A.K.T.Mahendiran B.Com.,** for creating this magnificent edifice of learning by providing all necessary facilities to complete diploma engineering course successfully.

We express our gratitude to our principal **Mr.P.K.Kabilar M.E.,MBA.,** for constant encouragement and facilities provided towards the successful completion of our project work.

At the same time we would like to express our heartfelt thanks to our head of the department **Mrs.C.Sridevi.,B.E.,(CSE).,** and also our project guide **Mrs. N.Moidheen Bee M.Sc., B.Ed.,** for guiding and needful advices and time to complete this project.

We would like to show our gratitude to our department staffs for all instructions and guidance given throughout.

Our sincere thanks and affection to our parents and friends who gave hand in all our steps.

CONTENTS

ABSTRACT

The Online Peer-to-Peer Car Rental System is a web-based platform developed to connect private car owners and renters through a secure, user-friendly, and efficient digital marketplace. The system enables vehicle owners to list their cars with complete details such as model, brand, price, availability, location, images, and rental conditions, while customers can search, compare, and book vehicles online based on their requirements. Built using Laravel, PHP 8.2+, and MySQL, the platform offers a modern and responsive interface for managing rentals, vehicle listings, user accounts, and booking transactions.

This proposed system is designed to simplify the traditional vehicle rental process by automating core operations such as registration, vehicle management, booking requests, payment tracking, approval workflows, and rental history maintenance. It supports multiple user roles including admin, car owner, and renter, ensuring proper access control and efficient management of the platform. The admin panel provides full control over users, cars, bookings, pricing, and reports, while owners can manage their listed vehicles and renters can easily make reservations from any device.

The main objective of this project is to create a reliable peer-to-peer rental environment that improves accessibility, reduces manual work, and increases asset utilization for vehicle owners. By integrating features such as availability calendars, booking status updates, document verification, secure authentication, and responsive design, the system enhances both user convenience and operational efficiency. This project is suitable for deployment on VPS or cloud hosting environments using Apache or Nginx, making it scalable, secure, and ready for real-world implementation in the automobile rental industry.

1. INTRODUCTION

The Online Peer-to-Peer Car Rental System is an innovative web-based application designed to revolutionize the traditional vehicle rental process by enabling direct interaction between car owners and renters. In conventional rental systems, users depend on centralized rental agencies, which often involve higher costs, limited vehicle options, and time-consuming procedures. This system eliminates such limitations by providing a decentralized platform where individuals can list their vehicles for rent and customers can easily browse and book cars according to their preferences. By leveraging modern web technologies like Laravel, PHP, and MySQL, the platform ensures a seamless, secure, and responsive user experience.

With the increasing demand for flexible transportation solutions, peer-to-peer rental systems have gained significant popularity worldwide. This project addresses that need by offering a scalable and efficient solution that allows vehicle owners to generate income from idle assets while providing renters with affordable and diverse vehicle choices. The system incorporates essential functionalities such as user registration, vehicle listing, advanced search filters, booking management, and payment tracking. It also includes role-based access control for administrators, owners, and customers, ensuring proper system governance and data security.

Furthermore, the platform is designed to operate in a cloud-based or VPS hosting environment using web servers like Apache or Nginx, ensuring high availability and performance. It supports multiple PHP extensions and modules required for secure transactions, data processing, and smooth application execution. Overall, this system aims to enhance convenience, transparency, and efficiency in the car rental ecosystem by integrating technology-driven solutions into everyday transportation needs.

2. LITERATURE SURVEY

. The development of an Online Peer-to-Peer Car Rental System is influenced by the rapid growth of digital transportation services, sharing economy platforms, and web-based booking applications. Earlier vehicle rental systems were mainly designed for large rental companies, where customers had to visit offices or rely on phone-based reservations. These systems provided only basic functionalities such as vehicle availability checking and manual booking confirmation. Over time, researchers and developers introduced online rental platforms that improved accessibility by allowing customers to browse vehicles, compare prices, and make reservations through web interfaces. However, many traditional systems were still limited to company-owned vehicles and lacked flexibility for individual car owners.

The concept of peer-to-peer rental platforms emerged as part of the sharing economy, where privately owned resources could be monetized through digital applications. Similar to property-sharing and ride-sharing models, peer-to-peer car rental systems allow vehicle owners to rent out their idle cars to others for short-term or long-term use. Several studies have highlighted that these systems improve resource utilization, reduce transportation costs, and offer users greater choice and convenience. Existing commercial platforms have demonstrated the success of this model by incorporating features such as secure authentication, vehicle verification, real-time booking, location-based search, and online payment integration. These studies emphasize that trust, transparency, and system usability are critical factors in the success of such applications.

From the software engineering perspective, modern car rental platforms increasingly adopt framework-based web development, especially using technologies like Laravel, Node.js, or Django, to ensure scalability, modularity, and maintainability. Literature also shows the importance of integrating responsive design, role-based access control, booking workflows, and secure database management for improving overall system performance. In addition, cloud hosting and VPS deployment have become common practices to achieve better availability, security, and scalability for online

rental platforms. Based on the existing literature, it is clear that a well-designed peer-to-peer car rental system should combine ease of use, efficient booking management, secure transactions, and reliable hosting infrastructure to meet the needs of both vehicle owners and renters in a competitive digital market.

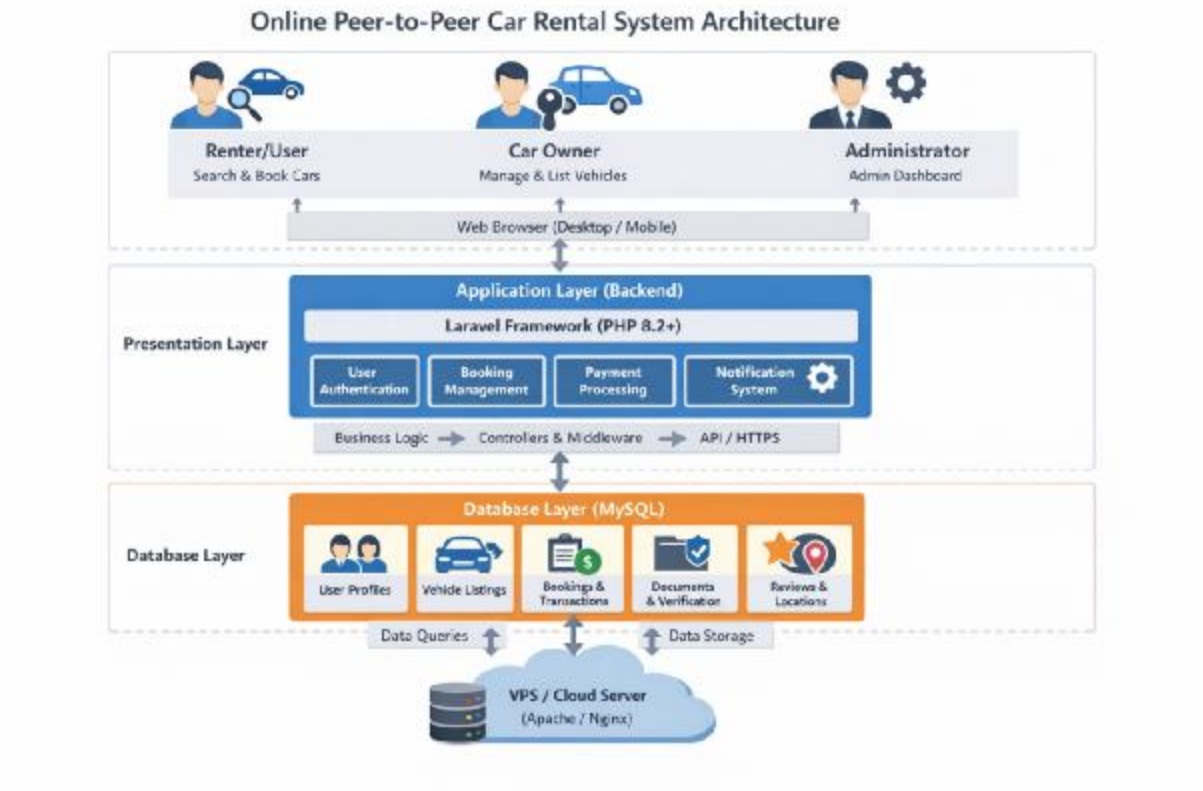
3. PROPOSED SYSTEM

The proposed Online Peer-to-Peer Car Rental System is a modern web-based application developed to provide a secure and efficient platform where private car owners can list their vehicles for rent and customers can book them online with ease. The system is designed using Laravel framework, PHP 8.2+, and MySQL, offering a robust architecture for handling multiple users, vehicle listings, reservations, and administrative controls. Unlike traditional rental systems that are limited to company-owned fleets, this platform allows individuals to participate directly in the rental ecosystem, thereby maximizing vehicle utilization and creating an additional source of income for car owners.

The proposed system includes three major user roles: Administrator, Car Owner, and Renter/User. The Administrator manages the complete platform, including user accounts, listed vehicles, booking approvals, pricing controls, reports, and dispute handling. The Car Owner can register, add vehicle details, upload car images, set rental prices, manage availability, and monitor booking requests. The Renter/User can search available cars based on location, type, price, and features, then place booking requests, upload required documents, and complete the rental process online. This role-based functionality ensures better control, data security, and smooth coordination among all participants.

To improve operational efficiency, the system automates important tasks such as vehicle listing approval, booking confirmation, rental status tracking, payment record management, and notification updates. It also supports responsive web design so that the platform can be accessed from desktops, tablets, and mobile devices. Hosting on VPS or cloud-based servers with Apache or Nginx ensures scalability, performance, and reliability in real-world deployment. Overall, the proposed system offers a practical and user-friendly solution for transforming the traditional car rental process into a digital peer-to-peer business model that is transparent, flexible, and convenient.

3.1 ARCHITECTURE



The architecture of the Online Peer-to-Peer Car Rental System is designed as a multi-user web-based client–server architecture that supports secure communication between renters, car owners, administrators, and the central application server. The system is developed using the Laravel framework as the backend application layer, MySQL as the database layer, and a responsive frontend built with HTML, CSS, JavaScript, and Blade templates. The architecture ensures smooth handling of vehicle listing, booking workflows, payment management, and administrative control through a centralized platform hosted on a VPS or cloud server using Apache or Nginx.

At the presentation layer, users interact with the system through a web browser on desktop or mobile devices. This layer provides separate dashboards and interfaces for renters, car owners, and administrators. Car owners can add and manage their vehicles, renters can search and book cars, and admins can monitor users, listings, bookings, and transactions. The responsive user interface communicates with the application server through secure HTTP/HTTPS requests, ensuring accessibility and user-friendly operation across multiple devices.

The application layer contains the main business logic of the platform. Laravel controllers, middleware, service classes, and validation mechanisms process user requests, authenticate users, manage bookings, check vehicle availability, calculate rental charges, and update booking statuses. This layer also handles document verification, notifications, and role-based access control. The system uses Laravel routing to direct user actions to the appropriate modules, while middleware secures restricted pages such as admin panels and owner dashboards. All important business operations are validated at this layer before data is stored or retrieved.

The database layer stores all essential information such as user profiles, car details, vehicle images, booking records, payment history, reviews, location details, and verification documents. MySQL tables are structured to maintain relationships among renters, owners, vehicles, and bookings. The application server interacts with the database using Laravel's Eloquent ORM or query builder, ensuring efficient and secure data operations. This layered architecture improves maintainability, scalability, and security, making the system suitable for real-time deployment in a peer-to-peer car rental business environment.

3.2 MODULES DESCRIPTION

1. User Registration and Authentication Module

This module manages the registration and login process for all types of users in the system, including **admin, car owner, and renter**. New users can create accounts by entering personal details such as name, email, phone number, password, and address. The module validates input data, encrypts passwords, and stores user credentials securely in the database. It also supports login authentication, password reset, profile update, and logout functionality. Role-based access control is implemented so that each user can access only the features assigned to their role. This module is essential for maintaining security and ensuring that only authorized users can use the system.

2. Car Owner Management Module

The car owner management module allows vehicle owners to create and manage their profiles and rental business details within the platform. Owners can update contact information, upload identity proof, submit verification documents, and manage their account settings. This module also helps the admin verify owner authenticity before allowing them to publish cars for rental. It improves trust and transparency in the platform by ensuring that only genuine vehicle owners participate in the peer-to-peer rental process. Through this module, owners can also view booking requests, earnings, rental history, and customer interactions.

3. Vehicle Listing Management Module

This module enables car owners to add, edit, update, and delete vehicle details from the system. Owners can upload car images, enter specifications such as brand, model, fuel type, seating capacity, transmission type, registration number, rental price per day, location, and availability status. The module ensures that complete and valid vehicle information is stored so renters can make informed decisions. It also supports vehicle categorization, featured listings, and listing approval by the administrator. This module is one of the core components of the system, as it directly handles the presentation of rentable vehicles on the platform.

4. Vehicle Search and Filter Module

The vehicle search and filter module helps renters quickly find suitable cars based on their needs. Users can search by city, pickup location, drop-off location, price range, car type, model, seating capacity, transmission, and fuel type. This module improves user experience by reducing search time and displaying only relevant vehicles. It may also include sorting options such as lowest price, newest listing, popular vehicles, or top-rated owners. A user-friendly search system increases booking opportunities and helps the platform operate more efficiently.

5. Booking and Reservation Management Module

This module handles the complete rental booking workflow from car selection to reservation confirmation. A renter can choose an available car, select rental dates, view pricing details, and place a booking request. The car owner or system can approve or reject the request depending on system design. Once approved, the booking status is updated and stored in the database. The module also manages booking history, cancellation requests, rental duration, booking modifications, and status tracking such as pending, confirmed, active, completed, or cancelled. This is one of the most important modules because it automates the entire reservation process.

6. Payment and Transaction Management Module

The payment module records rental charges, booking payments, deposits, refunds, and transaction history. It calculates the total rental cost based on duration, vehicle rate, taxes, service charges, and other applicable fees. Even if live payment gateway integration is not initially implemented, this module can still maintain payment records for manual or offline verification. It allows the admin and owner to track completed payments and pending dues. For future upgrades, this module can be integrated with online payment gateways such as Razorpay, Stripe, or PayPal. This module improves financial transparency and supports proper revenue management.

7. Document Verification Module

The document verification module is used to collect and validate important user and vehicle documents. Renters may need to upload a driving license, ID proof, and address proof, while car owners may upload RC book, insurance certificate, pollution certificate, and vehicle images. The administrator can review these documents before approving users or listings. This module increases trust and legal compliance within the system by ensuring that both parties meet rental eligibility requirements. It also reduces the risk of fraud, unauthorized bookings, and invalid vehicle listings.

8. Review and Rating Module

This module allows renters to provide feedback and ratings after completing a booking. Users can rate the car condition, owner behavior, booking experience, cleanliness, and overall satisfaction. Similarly, owners may also rate renters based on vehicle handling, punctuality, and communication. The review system helps future users make better rental decisions and improves transparency across the platform. It also motivates owners to maintain vehicle quality and service standards. Admins can monitor reviews and remove inappropriate or fake feedback if necessary.

9. Admin Dashboard and Control Module

The admin dashboard is the central control panel of the entire system. Through this module, the administrator can manage all users, vehicle listings, bookings, payments, reviews, verification requests, and reports. Admins can approve or reject car listings, verify documents, monitor suspicious activity, disable accounts, and view system-wide statistics. The dashboard may also include charts, booking summaries, earnings reports, and active user counts for better decision-making. This module ensures smooth platform operation, policy enforcement, and complete administrative control over the peer-to-peer rental application.

10. Notification and Communication Module

This module is responsible for sending important alerts and communication messages to renters, owners, and administrators. Notifications can be triggered for registration confirmation, booking requests, booking approval, payment updates, cancellation alerts, document verification status, and rental completion. Alerts may be delivered through email, SMS, or in-app notifications depending on the system configuration. This module improves real-time communication and keeps all users informed about important actions and updates. Effective notification handling reduces confusion, improves service quality, and enhances the overall user experience.

4. IMPLEMENTATION

The implementation of the **Online Peer-to-Peer Car Rental System** is carried out using the **Laravel framework**, which provides a structured and secure environment for developing modern web applications. The system is built with **PHP 8.2+** on the server side, **MySQL** as the database, and standard frontend technologies such as **HTML, CSS, JavaScript, Bootstrap, and Blade templates** for designing a responsive user interface. The project is hosted on a **VPS or cloud server** using **Apache or Nginx**, which ensures reliability, scalability, and real-time accessibility for users. Laravel's MVC architecture helps in separating the presentation layer, business logic, and database operations, making the application easier to develop, test, and maintain.

The implementation process begins with **database design and migration creation**, where tables are developed for users, car owners, vehicle listings, bookings, payments, documents, reviews, notifications, and admin controls. Laravel migrations and seeders are used to define and initialize the database structure efficiently. After this, authentication and role-based access control are implemented to provide separate dashboards and permissions for **admin, car owners, and renters**. Middleware is configured to secure restricted routes and protect sensitive modules from unauthorized access. Each major function such as vehicle management, booking handling, and payment tracking is developed as an individual module so that the system remains modular and maintainable.

On the frontend side, responsive pages are designed for home, login, registration, car listings, booking forms, owner dashboard, and admin dashboard. Car owners can add, edit, update, and remove vehicle records, while renters can search cars, filter results, place booking requests, and view booking history. Administrators can monitor the entire platform through a centralized dashboard, verify owners and documents, approve or reject listings, and manage transactions. File upload functionality is implemented for storing vehicle images and verification documents, while form validation and error handling ensure that only correct and complete data is submitted into the system. Notifications and booking status messages are also integrated to improve communication between users.

Finally, the system is deployed in a production environment by configuring the web server, database connection, environment variables, and required PHP extensions. Apache or Nginx handles incoming requests, while Laravel processes them and communicates with the MySQL database. Proper testing is conducted at each stage, including unit testing, form validation testing, booking workflow testing, and user acceptance testing to ensure the application performs correctly. Security measures such as password hashing, CSRF protection, input validation, and secure session handling are applied to safeguard user data and system operations. Thus, the implementation transforms the proposed design into a fully functional, real-time online peer-to-peer car rental platform suitable for practical deployment.

4.1 SOFTWARE REQUIEMENTS

he **Online Peer-to-Peer Car Rental System** is developed using modern web technologies and requires a well-configured software environment to ensure smooth performance, security, and scalability. The system is primarily based on the **Laravel framework**, which runs on **PHP 8.2 or higher**, and uses **MySQL** as the database management system. It is designed to operate on a **VPS or cloud hosting environment** with a web server such as **Apache or Nginx**, enabling real-time access and high availability for multiple users.

1. Operating System

- Windows 10 / 11 (for development)
- Linux (Ubuntu/CentOS) recommended for production server

2. Web Server

- Apache Server (with mod_rewrite enabled)

  OR
- Nginx Server

3. Backend Technology

- PHP >= 8.2
- Laravel Framework (Latest Version)

4. Database

- MySQL Server (5.7 or higher)

  OR
- MariaDB

5. Frontend Technology

- HTML5
- CSS3
- JavaScript
- Bootstrap (for responsive UI design)
- Blade Template Engine (Laravel)

6. Required PHP Extensions

- PDO PHP Extension

- OpenSSL PHP Extension
- Mbstring PHP Extension
- Exif PHP Extension
- Fileinfo Extension
- XML PHP Extension
- Ctype PHP Extension
- JSON PHP Extension
- Tokenizer PHP Extension
- cURL PHP Extension

7. Additional Tools & Software

- Composer (Dependency Manager for PHP)
- Git (Version Control System)
- Node.js & NPM (for frontend asset compilation, optional)

8. Server Configuration

- Enable **mod_rewrite** (Apache)
- Enable **HTTPS (SSL Certificate)** for secure communication
- Configure .env file for database and app settings
- Set proper file permissions for storage and cache

9. Browser Compatibility

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Safari

10. Hosting Environment

- VPS Server / Cloud Hosting (AWS, DigitalOcean, etc.)
- Minimum recommended configuration:
    - 2+ CPU Cores
    - 4 GB RAM
    - 50 GB Storage

4.2 SYSTEM REQUIREMENTS

1. Hardware Requirements

Development System Requirements

These are the minimum hardware requirements for developing and testing the project:

- **Processor:** Intel Core i3 / i5 or higher
- **RAM:** 8 GB minimum
- **Hard Disk:** 256 GB SSD or higher
- **Monitor:** 14-inch or above
- **Keyboard and Mouse:** Standard input devices
- **Internet Connection:** Stable broadband connection

These specifications are sufficient for coding, database handling, local server testing, and UI development.

Server Requirements (VPS Hosting)

For live deployment, the application requires a **Virtual Private Server (VPS)** with the following configuration:

Component	Specification
Server Type	Virtual Private Server (VPS)
CPU	8 Core Processor
RAM	32 GB
Storage	300 GB NVMe SSD
Bandwidth	High-speed / Unlimited preferred
Operating System	Ubuntu / CentOS / AlmaLinux
Web Server	Apache or Nginx
Database Server	MariaDB
Control Panel	WHM / cPanel
Backup Support	Daily / Weekly Backup Recommended
SSL Certificate	Required for secure access

4.3 SAMPLE CODING

```
@php
   $layout = BaseHelper::stringify(request()->query('layout'));

   if (!in_array($layout, ['list', 'grid'])) {
      $layout = $defaultLayout ?? 'grid';
   }

   $col = BaseHelper::stringify(request()->query('col'));

   if (empty($col)) {
      $col = (int) ($layoutCol ?? 4);
   }

   if(empty($enableFilter)) {
      $enableFilter = BaseHelper::stringify(request()->query('filter'));

      if (empty($enableFilter)) {
         $enableFilter = 'no';
      }
   }

@endphp

<div class="content-right cars-listing">
   @if($enableFilter === 'no')
      {!! Form::open(['url' => route('public.ajax.cars'), 'method' => 'GET', 'id' => 'cars-filter-form', 'class' => 'sidebar-filter-mobile__content']) !!}
      <input type="hidden" name="page" value="{{ $cars->currentPage() ?: 1 }}" data-value="{{ $cars->currentPage() ?: 1 }}" />
      <input type="hidden" name="per_page" value="{{ $cars->perPage() }}" />
```

```
        <input type="hidden" name="layout" value="{{ $layout }}" />
        <input type="hidden" name="layout_col" value="{{ $col }}" />
        <input type="hidden" name="filter" value="{{ $enableFilter }}" />
        <input type="hidden" name="sort_by" value="{{ BaseHelper::stringify(request()->query('sort_by')) }}" />
    @endif

    <div class="box-filters mb-25 pb-5 border-bottom border-1">
        <div class="row align-items-center">
            <div class="col-xl-4 col-md-4 mb-10 text-start">
                <div class="box-view-type">
                    @if($enableFilter === 'yes')
                        <a type="button" class="d-lg-none me-3 d-inline-block btn-mobile-filter" data-bs-toggle="offcanvas" data-bs-target="#mobileFiltersOffcanvas" aria-controls="mobileFiltersOffcanvas">
                            <x-core::icon name="ti ti-filter" />
                            {{ __('Filters') }}
                        </a>
                    @endif
                    <a @class(['display-type display-grid layout-car', 'active' => $layout === 'grid']) href="#"
                        data-layout="grid">
                        <svg width="22" height="22" viewbox="0 0 22 22" fill="none"
                            xmlns="http://www.w3.org/2000/svg">
                            <path
                                d="M20 8V2.75C20 2.3375 19.6625 2 19.25 2H14C13.5875 2 13.25 2.3375 13.25 2.75V8C13.25 8.4125 13.5875 8.75 14 8.75H19.25C19.6625 8.75 20 8.4125 20 8ZM19.25 0.5C20.495 0.5 21.5 1.505 21.5 2.75V8C21.5 9.245 20.495 10.25 19.25 10.25H14C12.755 10.25 11.75 9.245 11.75 8V2.75C11.75 1.505 12.755 0.5 14 0.5H19.25Z"
                                fill=""></path>
```

<path
d="M20 19.25V14C20 13.5875 19.6625 13.25 19.25 13.25H14C13.5875 13.25 13.25 13.5875 13.25 14V19.25C13.25 19.6625 13.5875 20 14 20H19.25C19.6625 20 20 19.6625 20 19.25ZM19.25 11.75C20.495 11.75 21.5 12.755 21.5 14V19.25C21.5 20.495 20.495 21.5 19.25 21.5H14C12.755 21.5 11.75 20.495 11.75 19.25V14C11.75 12.755 12.755 11.75 14 11.75H19.25Z"
fill=""></path>
<path
d="M8 8.75C8.4125 8.75 8.75 8.4125 8.75 8V2.75C8.75 2.3375 8.4125 2 8 2H2.75C2.3375 2 2 2.3375 2 2.75V8C2 8.4125 2.3375 8.75 2.75 8.75H8ZM8 0.5C9.245 0.5 10.25 1.505 10.25 2.75V8C10.25 9.245 9.245 10.25 8 10.25H2.75C1.505 10.25 0.5 9.245 0.5 8V2.75C0.5 1.505 1.505 0.5 2.75 0.5H8Z"
fill=""></path>
<path
d="M8 20C8.4125 20 8.75 19.6625 8.75 19.25V14C8.75 13.5875 8.4125 13.25 8 13.25H2.75C2.3375 13.25 2 13.5875 2 14V19.25C2 19.6625 2.3375 20 2.75 20H8ZM8 11.75C9.245 11.75 10.25 12.755 10.25 14V19.25C10.25 20.495 9.245 21.5 8 21.5H2.75C1.505 21.5 0.5 20.495 0.5 19.25V14C0.5 12.755 1.505 11.75 2.75 11.75H8Z"
fill=""></path>
</svg>

<a @class(['display-type display-grid layout-car', 'active' => $layout === 'list']) href="#"
data-layout="list">
<svg width="21" height="21" viewbox="0 0 21 21" fill="none"
xmlns="http://www.w3.org/2000/svg">
<path
d="M4.788 0H1.09497C0.491194 0 0 0.486501 0 1.08456V4.74269C0 5.34075 0.491194 5.82729 1.09497 5.82729H4.788C5.39177 5.82729 5.88297 5.34075 5.88297 4.74269V1.08456C5.88297 0.486501 5.39177 0

4.788 0ZM4.80951 4.74273C4.80951 4.75328 4.79865 4.76404 4.788 4.76404H1.09497C1.08432 4.76404 1.07345 4.75328 1.07345 4.74273V1.08456C1.07345 1.07401 1.08432 1.06329 1.09497 1.06329H4.788C4.79865 1.06329 4.80951 1.07401 4.80951 1.08456V4.74273ZM7.53412 1.32686C7.53412 1.03321 7.77444 0.795211 8.07084 0.795211H20.4632C20.7596 0.795211 21 1.03321 21 1.32686C21 1.62046 20.7596 1.8585 20.4632 1.8585H8.07084C7.77444 1.8585 7.53412 1.62046 7.53412 1.32686ZM21 4.50043C21 4.79408 20.7597 5.03208 20.4633 5.03208H8.07084C7.77444 5.03208 7.53412 4.79408 7.53412 4.50043C7.53412 4.20683 7.77444 3.96879 8.07084 3.96879H20.4632C20.7597 3.96879 21 4.20683 21 4.50043ZM4.788 7.58633H1.09497C0.491194 7.58633 0 8.07283 0 8.67089V12.329C0 12.9271 0.491194 13.4136 1.09497 13.4136H4.788C5.39177 13.4136 5.88297 12.9271 5.88297 12.329V8.67089C5.88297 8.07288 5.39177 7.58633 4.788 7.58633ZM4.80951 12.3291C4.80951 12.3396 4.79865 12.3504 4.788 12.3504H1.09497C1.08432 12.3504 1.07345 12.3396 1.07345 12.3291V8.67094C1.07345 8.66039 1.08432 8.64967 1.09497 8.64967H4.788C4.79865 8.64967 4.80951 8.66039 4.80951 8.67094V12.3291ZM4.788 15.1727H1.09497C0.491194 15.1727 0 15.6592 0 16.2573V19.9154C0 20.5135 0.491194 21 1.09497 21H4.788C5.39177 21 5.88297 20.5135 5.88297 19.9154V16.2573C5.88297 15.6592 5.39177 15.1727 4.788 15.1727ZM4.80951 19.9154C4.80951 19.926 4.79865 19.9368 4.788 19.9368H1.09497C1.08432 19.9368 1.07345 19.926 1.07345 19.9154V16.2573C1.07345 16.2468 1.08432 16.236 1.09497 16.236H4.788C4.79865 16.236 4.80951 16.2468 4.80951 16.2573V19.9154ZM12.0868C21 12.3805 20.7597 12.6185 20.4633 12.6185H8.07084C7.77444 12.6185 7.53412 12.3805 7.53412 12.0868C7.53412 11.7932 7.77444 11.5552 8.07084 11.5552H20.4632C20.7597 11.5552 21 11.7932 21 12.0868ZM8.91328C21 9.20688 20.7597 9.44492 20.4633 9.44492H8.07084C7.77444 9.44492 7.53412 9.20688 7.53412 8.91328C7.53412 8.61963 7.77444 8.38163 8.07084 8.38163H20.4632C20.7597 8.38163 21 8.61963 21 8.91328ZM16.4996C21 16.7932 20.7597 17.0313 20.4633 17.0313H8.07084C7.77444 17.0313 7.53412

```
16.7932 7.53412 16.4996C7.53412 16.206 7.77444 15.968 8.07084
15.968H20.4632C20.7597 15.968 21 16.206 21 16.4996ZM21 19.6732C21 19.9668
20.7597 20.2048 20.4633 20.2048H8.07084C7.77444 20.2048 7.53412 19.9668
7.53412 19.6732C7.53412 19.3796 7.77444 19.1415 8.07084
19.1415H20.4632C20.7597 19.1415 21 19.3796 21 19.6732Z"
                    fill=""></path>
                </svg>
            </a>
            <span class="text-sm-bold neutral-500 number-found">{{ __(':total items
found', ['total' => $cars->total()]) }}</span>
        </div>
    </div>
    <div class="col-xl-8 col-md-8 mb-10 text-lg-end text-start">
        <div class="box-item-sort">
            <a class="btn btn-sort" href="#">
                <svg width="18" height="18" viewbox="0 0 18 18" fill="none"
                    xmlns="http://www.w3.org/2000/svg">
                    <path d="M8.25 6L5.25 3M5.25 3L2.25 6M5.25 3L5.25 15"
stroke=""
                        stroke-width="1.5" stroke-linecap="round" stroke-
linejoin="round"></path>
                    <path d="M9.75 12L12.75 15M12.75 15L15.75 12M12.75 15L12.75
3" stroke=""
                        stroke-width="1.5" stroke-linecap="round" stroke-
linejoin="round"></path>
                </svg>
            </a>
            <div class="item-sort border-1">
                <span class="text-xs-medium neutral-500 mr-5">{{ __('Show')
}}</span>
                <div class="dropdown dropdown-sort border-1-right">
```

```
                    <button class="btn dropdown-toggle" id="dropdownSort2"
type="button"
                        data-bs-toggle="dropdown" aria-expanded="false"
                        data-bs-display="static"><span>{{ $cars->perPage()
}}</span></button>
                    <ul class="dropdown-menu dropdown-menu-light" aria-
labelledby="dropdownSort2">
                        @foreach($perPages ?? CarListHelper::getPerPageParams() as
$value)
                            <li>
                                <a class="dropdown-item dropdown-sort-by per-page-item"
href="#"
                                    data-per-page="{{ $value }}">{{ $value }}</a>
                            </li>
                        @endforeach
                    </ul>
                </div>
            </div>
            <div class="item-sort border-1">
                @include(Theme::getThemeNamespace('views.car-rentals.car-
list.partials.sort-by-dropdown'))
            </div>
        </div>
    </div>
</div>
<div class="box-grid-hotels wow fadeIn car-items">
    <div class="row position-relative">
        @include(Theme::getThemeNamespace('views.car-rentals.car-
list.partials.loading-ajax'))
```

```
            @forelse($cars as $car)
                @include(Theme::getThemeNamespace('views.car-rentals.car-
list.partials.car-item-' . $layout), [
                    'car' => $car,
                    'layoutCol' => $col
                ])
            @empty
                @include(Theme::getThemeNamespace('views.car-rentals.car-
list.partials.car-item-empty'))
            @endforelse
        </div>
    </div>
{!!                                                    $cars->withQueryString()-
>links(Theme::getThemeNamespace('partials.pagination')) !!}

</div>

@if($enableFilter === 'no')
    {!! Form::close() !!}
@endif
```

Car List

```
@php
    $transmission = $car->transmission;
    $types = $car->types;
    $make = $car->make;
    $carUrl = $car->url;

    $query = [];

    if ($startDate = BaseHelper::stringify(request()->query('start_date'))) {
        $query['rental_start_date'] = $startDate;
    }

    if ($endDate = BaseHelper::stringify(request()->query('end_date'))) {
        $query['rental_end_date'] = $endDate;
    }

    if ($query) {
        $carUrl = $car->url . '?' . http_build_query($query);
    }
@endphp
<div class="col-xl-12 col-lg-12">
    <div class="card-flight card-hotel card-property background-card border">
        <div class="card-image">
            <a href="{{ $carUrl }}">
                {{ RvMedia::image($car->image , $car->name, 'medium-rectangle') }}
            </a></div>
        <div class="card-info p-md-40 p-3">
            @if($avgReview = $car->avg_review)
            <div class="tour-rate">
```

```
            <div class="rate-element">
                <span class="rating">
                    <x-core::icon name="ti ti-star" size="16" class="icon icon-tabler icons-tabler-filled icon-tabler-star" />

                    <span>
                        {{ $avgReview }}
                    </span>
                    @if($reviewsCount = $car->reviews_count ?? 0)
                        <span class="text-sm-medium neutral-500">
                            ({{ $reviewsCount }} {{ $reviewsCount > 1 ? __('reviews') : __('review') }})
                        </span>
                    @endif
                </span>
            </div>
        </div>
        @endif
        <div class="card-title">
            <a class="heading-6 neutral-1000 text-ellipsis-2-lines" href="{{ $carUrl }}">
                {{ $car->name }}
            </a>
        </div>
        <div class="card-program">
            <div class="card-location mb-25">
                @if($car->current_location)
                    <p class="text-location text-md-medium neutral-500 text-truncate" title="{{ $car->current_location }}">
                        <x-core::icon name="ti ti-map-pin" />
                        {{ BaseHelper::clean($car->current_location) }}
```

```
                </p>
            @endif
        </div>
        <div class="card-facilities">
            <div class="item-facilities">
                <p class="room text-md-medium neutral-1000">{{ $car->mileage_display }}</p>
            </div>

            @if($transmission && $transmission->name)
                <div class="item-facilities">
                    <p class="size text-md-medium neutral-1000">{{ $transmission->name }}</p>
                </div>
            @endif
            @if($types && $types->name)
                <div class="item-facilities">
                    <p class="parking text-md-medium neutral-1000">{{ $types->name }}</p>
                </div>
            @endif
            @if($numberOfSeat = $car->number_of_seats)
                <div class="item-facilities">
                    <p class="bathroom text-md-medium neutral-1000">{{ $numberOfSeat }} {{ $numberOfSeat == 1 ? __('seat') : __('seats') }}</p>
                </div>
            @endif
            @if($make && $make->name)
                <div class="item-facilities">
                    <p class="pet text-md-medium neutral-1000">{{ $make->name }}</p>
```

```
                    </div>
                @endif
            </div>
            <div class="endtime">
                @include(Theme::getThemeNamespace('views.car-rentals.price'),     ['car'
=> $car])
                @include(Theme::getThemeNamespace('views.car-rentals.book-now-
button'), ['car' => $car])
            </div>
        </div>
    </div>
</div>
```

Car Make

```
@php
    Theme::set('pageTitle', $carMake->name);

    $layout = BaseHelper::stringify(request()->query('layout'));

    if (!in_array($layout, ['list', 'grid'])) {
        $layout = $defaultLayout ?? 'grid';
    }

    $col = BaseHelper::stringify(request()->query('col'));

    if (empty($col)) {
        $col = (int) ($layoutCol ?? 4);
    }

    $enableFilter ??= 'no';
@endphp

<div class="content-right cars-listing">
    @if($enableFilter === 'no')
        {!! Form::open(['url' => route('public.ajax.car_makes'), 'method' => 'GET', 'id' =>
'cars-filter-form', 'class' => 'sidebar-filter-mobile__content']) !!}
        <input type="hidden" name="page" data-value="{{ $cars->currentPage() ?: 1 }}"
/>
        <input type="hidden" name="per_page" />
        <input type="hidden" name="layout" value="{{ $layout }}" />
        <input type="hidden" name="layout_col" value="{{ $col }}" />
        <input type="hidden" name="filter" value="{{ $enableFilter }}" />
        <input type="hidden" name="car_makes[]" value="{{ $carMake->getKey() }}"
/>
```

```
    <input type="hidden" name="sort_by" value="{{ BaseHelper::stringify(request()->query('sort_by')) }}" />
  @endif

  <div class="box-filters mb-25 pb-5 border-bottom border-1">
    <div class="row align-items-center">
      <div class="col-xl-4 col-md-4 mb-10 text-start">
        <div class="box-view-type">
          <a @class(['display-type display-grid layout-car', 'active' => $layout === 'grid']) href="#"
            data-layout="grid">
            <x-core::icon name="ti ti-grid-dots" size="22" />
          </a>
          <a @class(['display-type display-grid layout-car', 'active' => $layout === 'list']) href="#"
            data-layout="list">
            <x-core::icon name="ti ti-list" size="21" />
          </a>
          <span class="text-sm-bold neutral-500 number-found">{{ $cars->total() }} items found</span>
        </div>
      </div>
      <div class="col-xl-8 col-md-8 mb-10 text-lg-end text-start">
        <div class="box-item-sort">
          <a class="btn btn-sort" href="#">
            <x-core::icon name="ti ti-arrows-sort" size="18" />
          </a>
          <div class="item-sort border-1">
            <span class="text-xs-medium neutral-500 mr-5">Show</span>
            <div class="dropdown dropdown-sort border-1-right">
```

```
                    <button class="btn dropdown-toggle" id="dropdownSort2"
type="button"
                        data-bs-toggle="dropdown" aria-expanded="false"
                        data-bs-display="static"><span>{{ $cars->perPage()
}}</span></button>
                    <ul class="dropdown-menu dropdown-menu-light" aria-
labelledby="dropdownSort2">
                      @foreach($perPages ?? CarListHelper::getPerPageParams() as
$value)
                        <li>
                          <a class="dropdown-item dropdown-sort-by per-page-item"
href="#"
                            data-per-page="{{ $value }}">{{ $value }}</a>
                        </li>
                      @endforeach
                    </ul>
                  </div>
                </div>
                <div class="item-sort border-1">
                  @include(Theme::getThemeNamespace('views.car-rentals.car-
list.partials.sort-by-dropdown'))
                </div>
              </div>
            </div>
          </div>
        </div>
        <div class="box-grid-hotels wow fadeIn car-items">
          <div class="row position-relative">
            @include(Theme::getThemeNamespace('views.car-rentals.car-
list.partials.loading-ajax'))
```

```
        @forelse($cars as $car)
            @include(Theme::getThemeNamespace('views.car-rentals.car-list.partials.car-item-' . $layout), [
                'car' => $car,
                'layoutCol' => $col
            ])
        @empty
            @include(Theme::getThemeNamespace('views.car-rentals.car-list.partials.car-item-empty'))
        @endforelse
    </div>
  </div>
  {!! $cars->withQueryString()->links(Theme::getThemeNamespace('partials.pagination')) !!}

</div>
```

Care Export

```
@extends('packages/data-synchronize::export')

@php
  $totalItems = 0;
  $counters = $exporter->getCounters();
  foreach ($counters as $counter) {
    if (str_contains(strtolower($counter->getLabel()), 'total')) {
      $value = str_replace(',', '', $counter->getValue());
      if (is_numeric($value) && $value > $totalItems) {
        $totalItems = (int) $value;
      }
    }
  }
```

```
    $isLargeExport = $totalItems > 10000;
    $isMediumExport = $totalItems > 1000 && $totalItems <= 10000;
@endphp

@push('header')
    <style>
        .export-recommendation-grid {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
            gap: 1rem;
            margin: 1rem 0;
        }

        .export-recommendation-item {
            padding: 1rem;
            background: #f8f9fa;
            border-radius: 6px;
        }

        @media (max-width: 768px) {
            .export-recommendation-grid {
                grid-template-columns: 1fr;
            }
        }
    </style>
@endpush

@section('export_extra_filters_after')
    <div class="row mb-3">
        <div class="col-md-4">
            <x-core::form.select
```

```
            name="import_type"
            :label="trans('plugins/location::location.export.import_type')"
            :options="[
                '' => trans('plugins/location::location.export.all_types'),
                'country' => trans('plugins/location::location.import_type.country'),
                'state' => trans('plugins/location::location.import_type.state'),
                'city' => trans('plugins/location::location.import_type.city'),
            ]"
        />
    </div>
    <div class="col-md-4">
        <x-core::form.select
            name="status"
            :label="trans('core/base::forms.status')"
            :options="[
                '' => trans('plugins/location::location.export.all_status'),
                'published' => trans('core/base::enums.statuses.published'),
                'draft' => trans('core/base::enums.statuses.draft'),
                'pending' => trans('core/base::enums.statuses.pending'),
            ]"
        />
    </div>
</div>

<div class="row mb-3">
    <div class="col-md-6">
        <div class="mb-3">
            <x-core::form.checkbox
                name="use_chunked_export"
                :value="1"
                :label="trans('plugins/location::location.export.use_chunked_export')"
```

```
                :checked="true"
                :helper-
text="trans('plugins/location::location.export.use_chunked_export_helper')"
            />
        </div>

        <div class="mb-3">
            <x-core::form.checkbox
                name="optimize_memory"
                :value="1"
                :label="trans('plugins/location::location.export.optimize_memory')"
                :checked="true"
                :helper-
text="trans('plugins/location::location.export.optimize_memory_helper')"
            />
        </div>

        <div class="mb-3">
            <x-core::form.checkbox
                name="use_streaming"
                :value="1"
                :label="trans('plugins/location::location.export.use_streaming')"
                :checked="$isLargeExport"
                :helper-
text="trans('plugins/location::location.export.use_streaming_helper')"
            />
        </div>

        @if ($isLargeExport)
            <div class="alert alert-success d-block">
```

```
                <strong>{{
trans('plugins/location::location.export.streaming_enabled_title') }}</strong><br>
                <small>{{
trans('plugins/location::location.export.streaming_enabled_message') }}</small>
            </div>
        @endif
    </div>

    <div class="col-md-6">
        <x-core::form-group>
            <x-core::form.label
                for="chunk_size">{{ trans('plugins/location::location.export.chunk_size')
}}</x-core::form.label>
            <x-core::form.text-input
                type="number"
                name="chunk_size"
                id="chunk_size"
                value="{{ $isLargeExport ? 200 : 300 }}"
                min="50"
                max="1000"
            />
            <x-core::form.helper-text>
                {{ trans('plugins/location::location.export.chunk_size_helper') }}
            </x-core::form.helper-text>

            <div class="mt-2">
                <div class="d-flex justify-content-between small text-muted mb-1">
                    <span>{{
trans('plugins/location::location.export.recommended_range') }}</span>
                    <span id="chunk-recommendation">
                        @if ($isLargeExport)
```

```
                    {{ trans('plugins/location::location.export.range_large_export') }}
                @elseif($isMediumExport)
                    {{  trans('plugins/location::location.export.range_medium_export')
}}
                @else
                    {{ trans('plugins/location::location.export.range_small_export') }}
                @endif
            </span>
        </div>
        <div
            class="progress"
            style="height: 4px;"
        >
            <div
                class="progress-bar bg-success"
                role="progressbar"
                style="width: 60%"
            ></div>
        </div>
    </div>
</x-core::form-group>
</div>
</div>

@if ($isLargeExport)
<x-core::alert
    type="warning"
    class="mb-4"
>
    <div>
```

```
            <h5                                                    class="mb-2">{{
trans('plugins/location::location.export.large_dataset_warning_title') }}</h5>
            <p class="mb-3">
                {{
trans('plugins/location::location.export.large_dataset_specific_message',  ['count'  =>
number_format($totalItems)]) }}
            </p>

            <div class="export-recommendation-grid">
                <div class="export-recommendation-item bg-white">
                    <strong                                          class="d-block">{{
trans('plugins/location::location.export.format_label') }}</strong>
                    <span
                        class="text-muted                                        small">{{
trans('plugins/location::location.export.csv_recommended') }}</span>
                </div>

                <div class="export-recommendation-item bg-white">
                    <strong                                          class="d-block">{{
trans('plugins/location::location.export.chunk_label') }}</strong>
                    <span
                        class="text-muted                                        small">{{
trans('plugins/location::location.export.chunk_recommended') }}</span>
                </div>

                <div class="export-recommendation-item bg-white">
                    <strong                                          class="d-block">{{
trans('plugins/location::location.export.time_label') }}</strong>
                    <span
                        class="text-muted                                        small">{{
trans('plugins/location::location.export.time_estimate') }}</span>
```

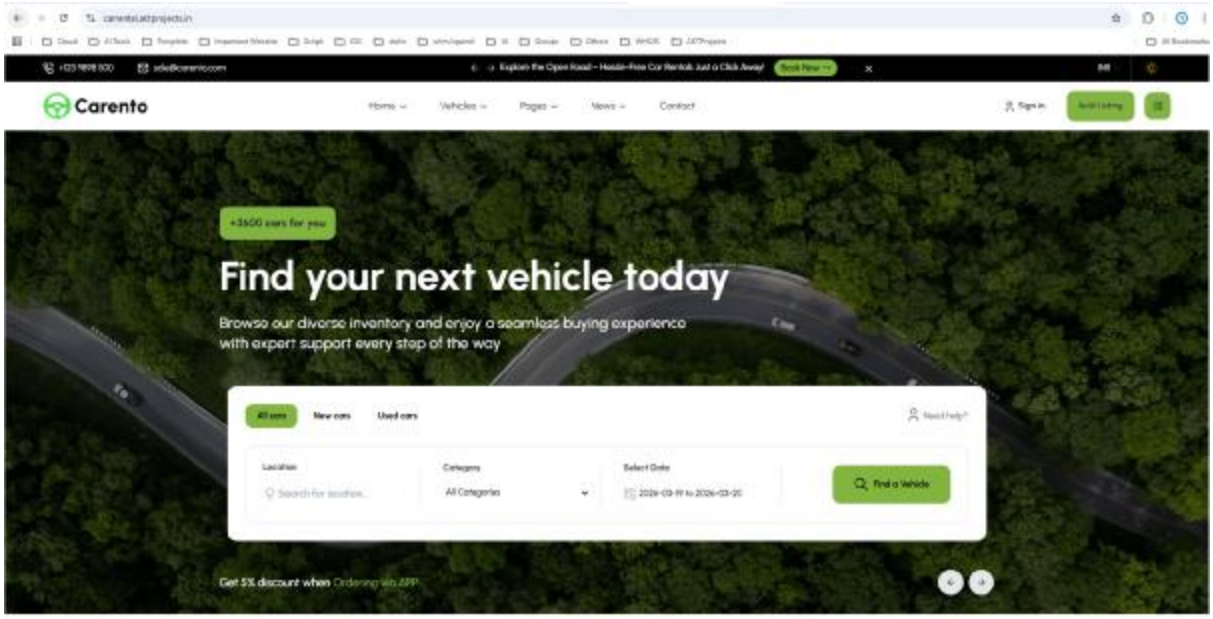
```
            </div>
        </div>

        <div class="alert alert-warning mt-3 mb-0">
            <strong>{{ trans('plugins/location::location.export.pro_tip') }}</strong>
            {{ trans('plugins/location::location.export.pro_tip_message') }}
        </div>
    </div>
</x-core::alert>
@elseif($isMediumExport)
<x-core::alert
    type="info"
    class="mb-4"
>
    <h6 class="mb-1">{{ trans('plugins/location::location.export.medium_dataset_detected') }}</h6>
    <p class="mb-0">
        {{ trans('plugins/location::location.export.medium_dataset_message', ['count' => number_format($totalItems)]) }}
    </p>
</x-core::alert>
@endif

@push('footer')
<script>
    document.addEventListener('DOMContentLoaded', function() {
        const chunkSizeInput = document.getElementById('chunk_size');
        const totalItems = {{ $totalItems }};
        const isLargeExport = {{ $isLargeExport ? 'true' : 'false' }};

        if (chunkSizeInput) {
```

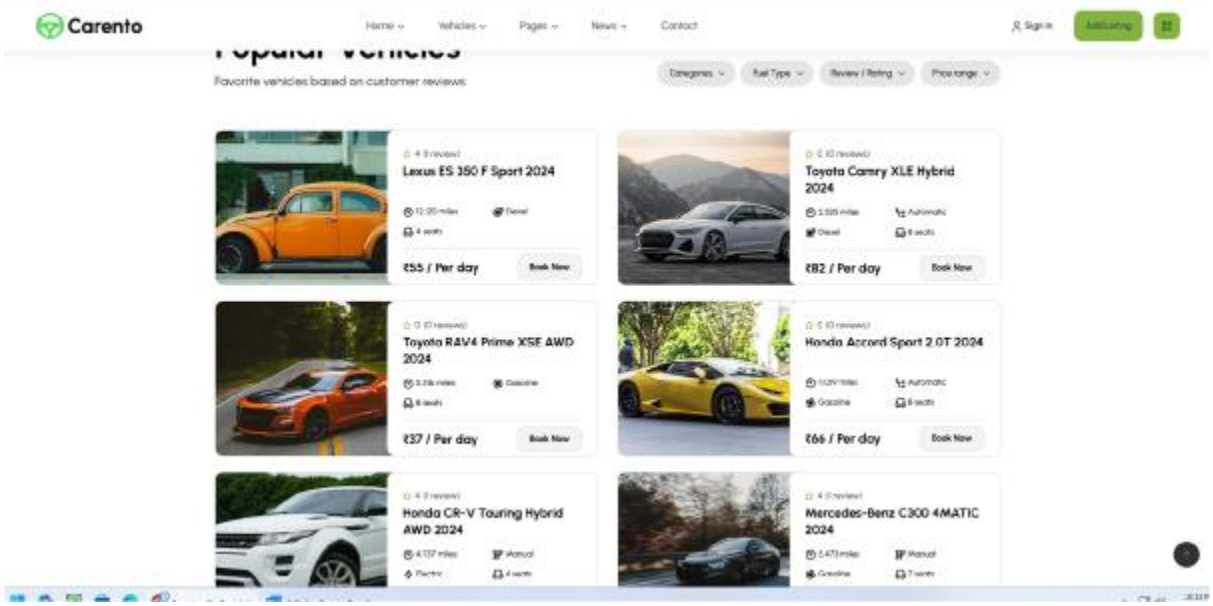
```
            chunkSizeInput.addEventListener('input', function() {
                const value = parseInt(this.value);
                const recommendation = document.getElementById('chunk-
recommendation');
                const progressBar = this.parentElement.querySelector('.progress-bar');

                let recommendedMin, recommendedMax, color, width;

                if (isLargeExport) {
                    recommendedMin = 150;
                    recommendedMax = 300;
                } else if (totalItems > 1000) {
                    recommendedMin = 200;
                    recommendedMax = 500;
                } else {
                    recommendedMin = 300;
                    recommendedMax = 700;
                }

                if (value >= recommendedMin && value <= recommendedMax) {
                    color = 'bg-success';
                    width = '80%';
                    recommendation.textContent =
                        '{{ trans('plugins/location::location.export.optimal_range') }}';
                } else if (value < recommendedMin) {
                    color = 'bg-warning';
                    width = '40%';
                    recommendation.textContent =
                        '{{ trans('plugins/location::location.export.too_small_slow') }}';
                } else {
                    color = 'bg-danger';
```

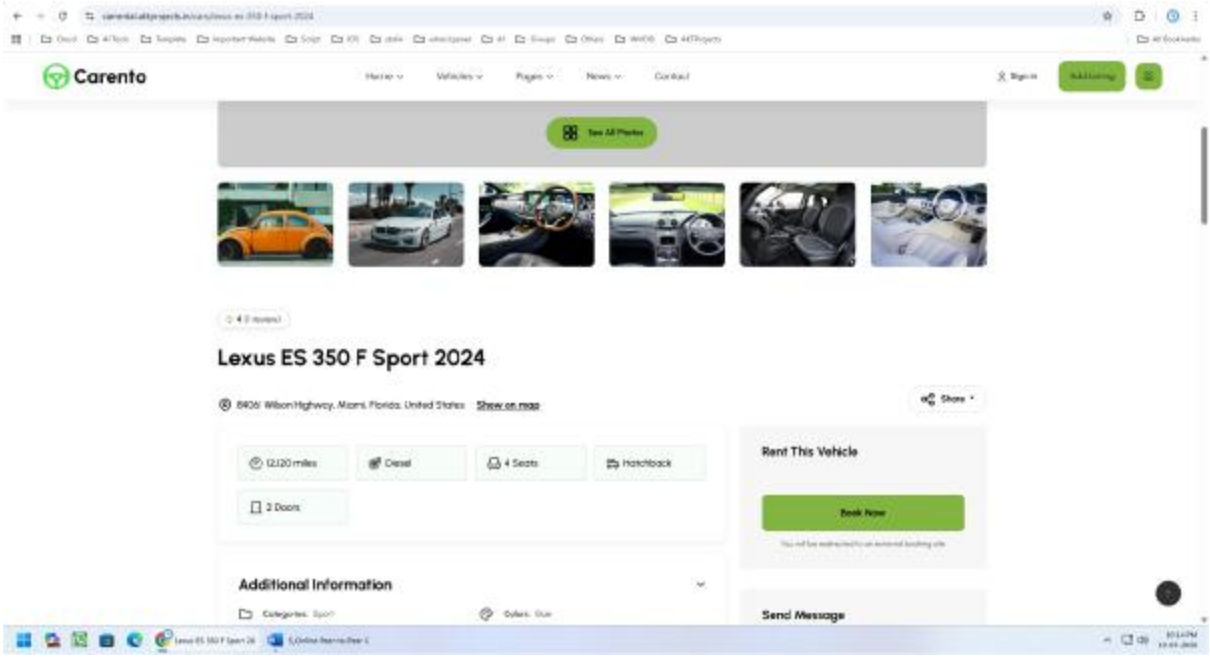
```
                width = '20%';
                recommendation.textContent =
                    '{{ trans('plugins/location::location.export.too_large_timeouts') }}';
            }

            progressBar.className = `progress-bar ${color}`;
            progressBar.style.width = width;
        });
    }

    @if ($isLargeExport)
        const excelRadio = document.querySelector('input[name="format"][value="xlsx"]');
        const csvRadio = document.querySelector('input[name="format"][value="csv"]');

        if (excelRadio && csvRadio) {
            excelRadio.disabled = true;

            const excelLabel = excelRadio.closest('label');
            if (excelLabel) {
                excelLabel.style.opacity = '0.8';
                excelLabel.style.cursor = 'not-allowed';

                const warningText = document.createElement('small');
                warningText.className = 'text-warning d-block mt-1';
                warningText.innerHTML =
                    '<i class="fas fa-exclamation-triangle me-1"></i>{{ trans('plugins/location::location.export.excel_disabled_warning', ['count' => number_format($totalItems)]) }}';
```

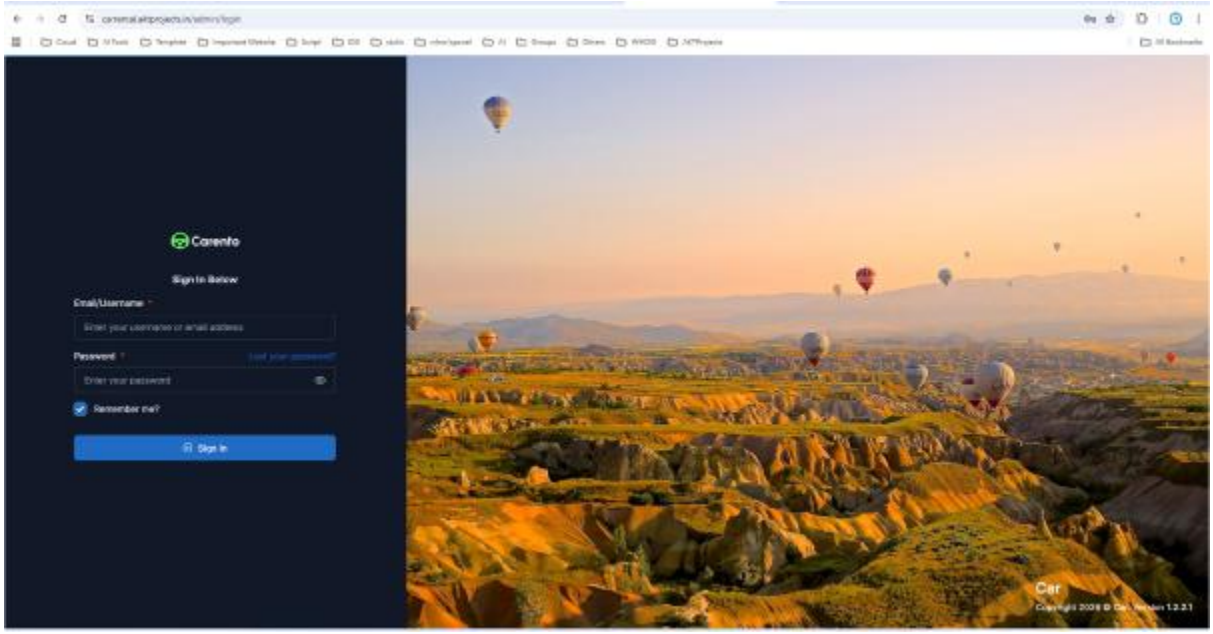
```
                excelLabel.appendChild(warningText);
            }

            csvRadio.checked = true;
            csvRadio.dispatchEvent(new Event('change', {
                bubbles: true
            }));

            const streamingCheckbox = document.querySelector('input[name="use_streaming"]');
            const chunkedCheckbox = document.querySelector('input[name="use_chunked_export"]');
            const memoryCheckbox = document.querySelector('input[name="optimize_memory"]');

            if (streamingCheckbox) streamingCheckbox.checked = true;
            if (chunkedCheckbox) chunkedCheckbox.checked = true;
            if (memoryCheckbox) memoryCheckbox.checked = true;
        }
    @endif
});
</script>
@endpush
@stop
```

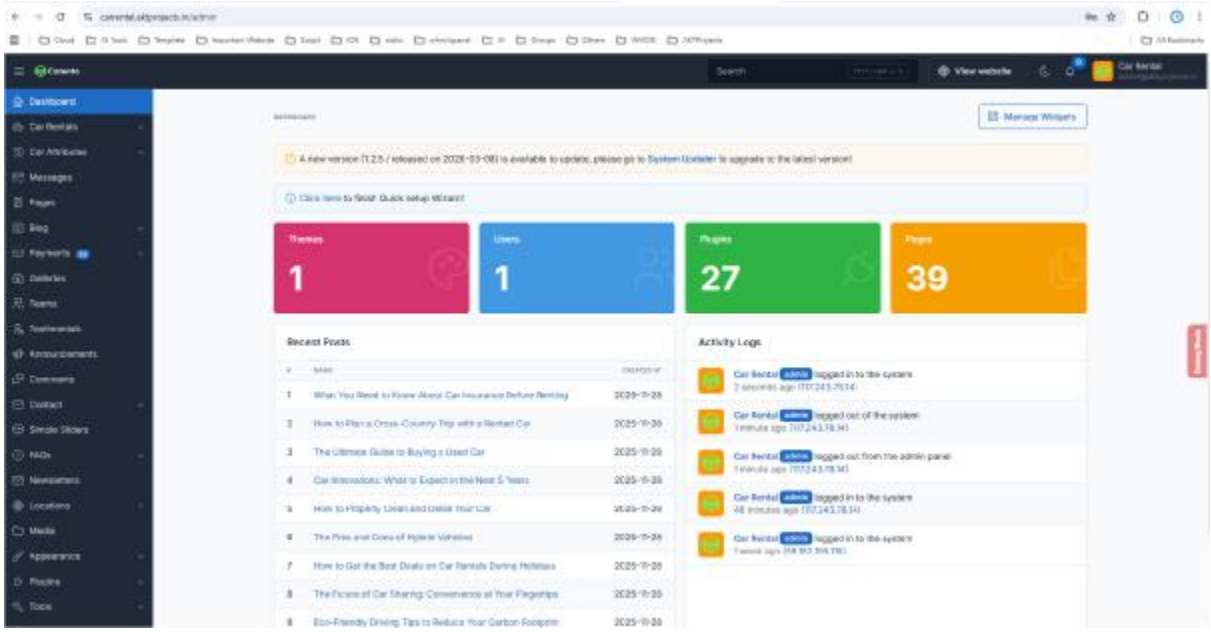
4.4. SCREEN SHOT

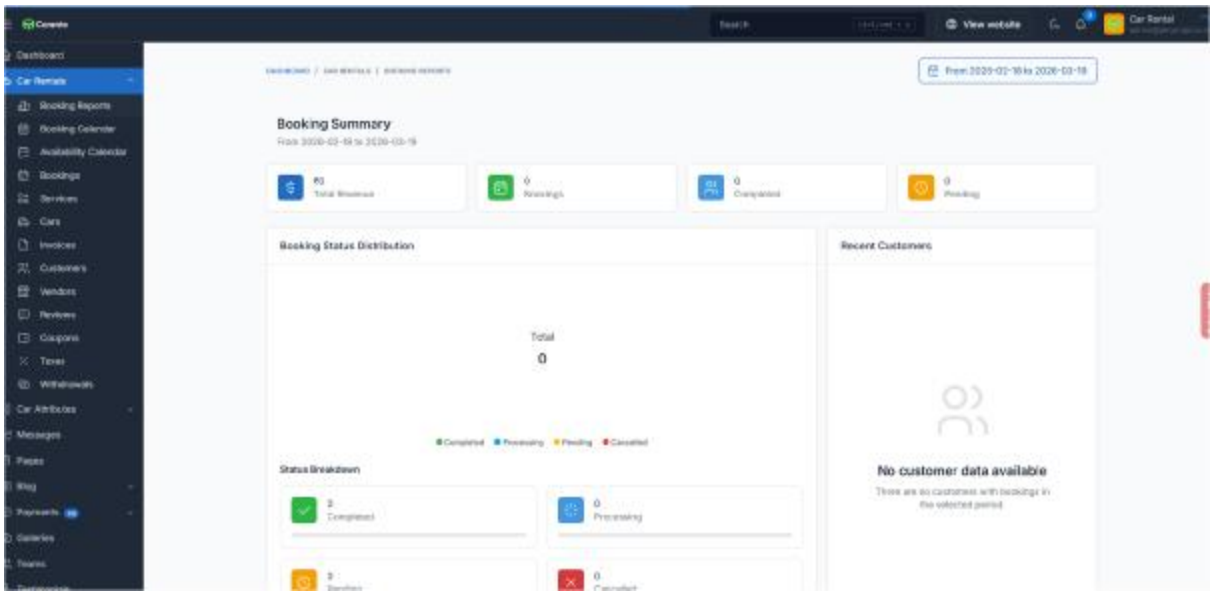


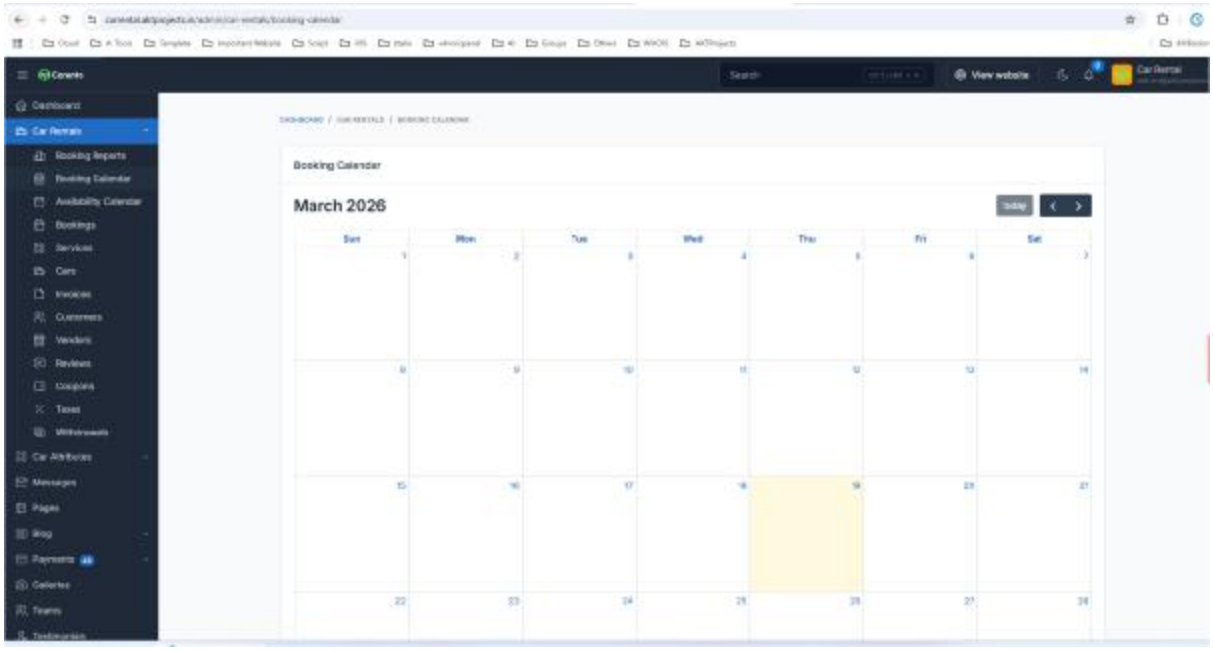


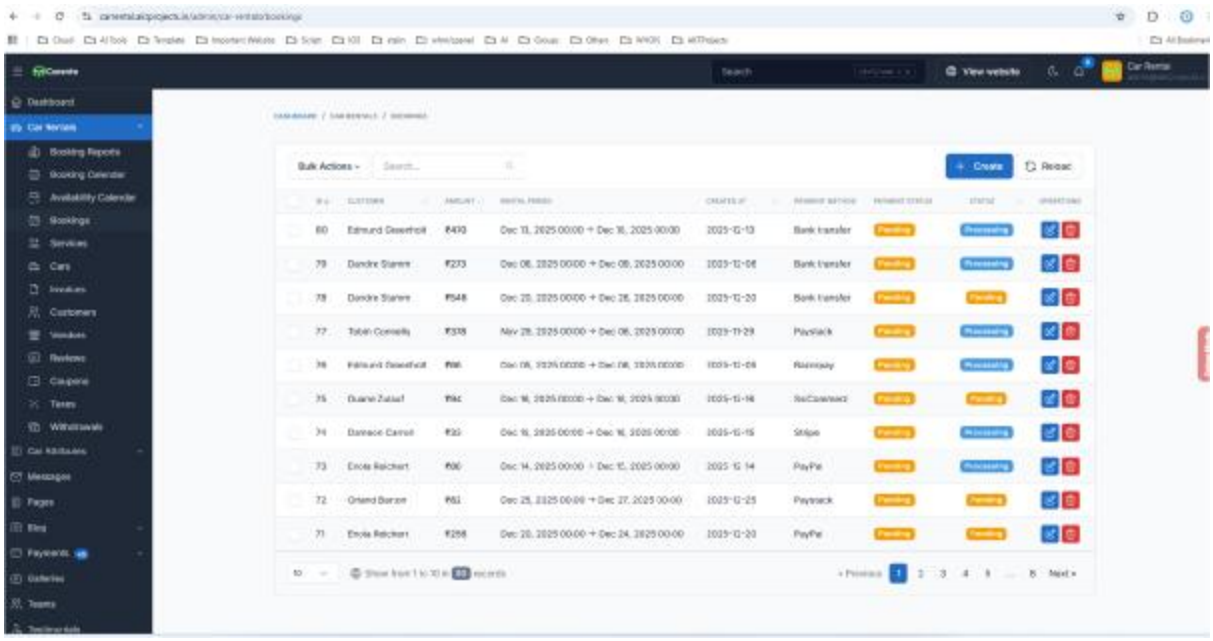

Carento
Lexus ES 350 F Sport 2024
Additional Information
Rent This Vehicle
Send Message

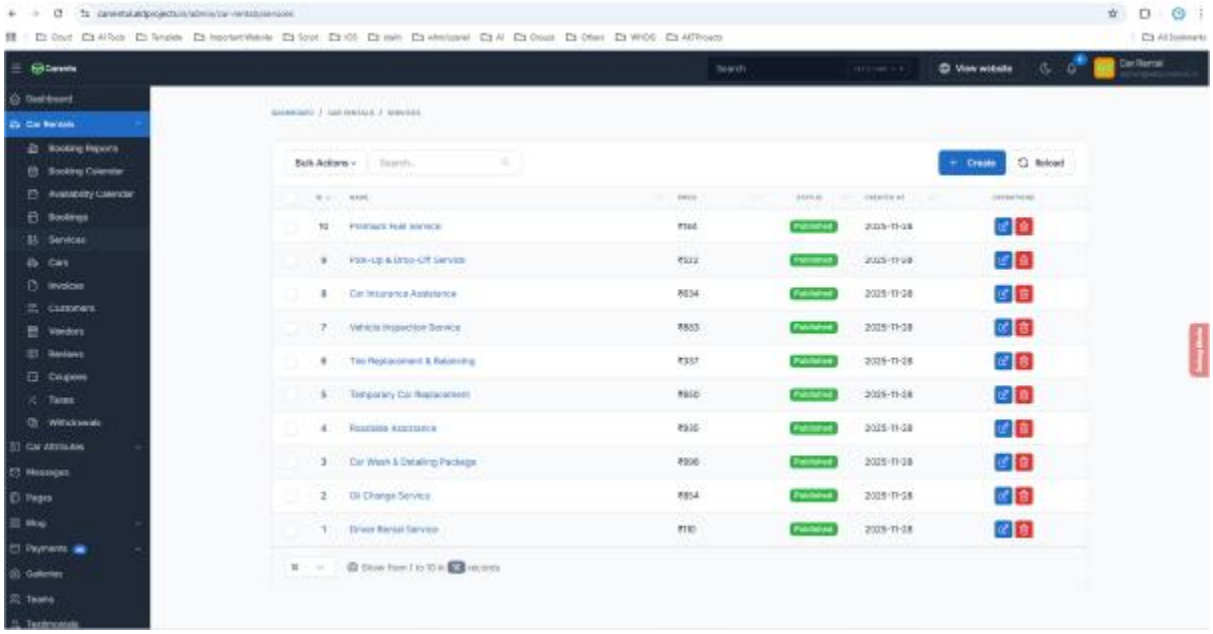

Carento
Sign In Below
Email/Username
Password
Remember me?
Sign In

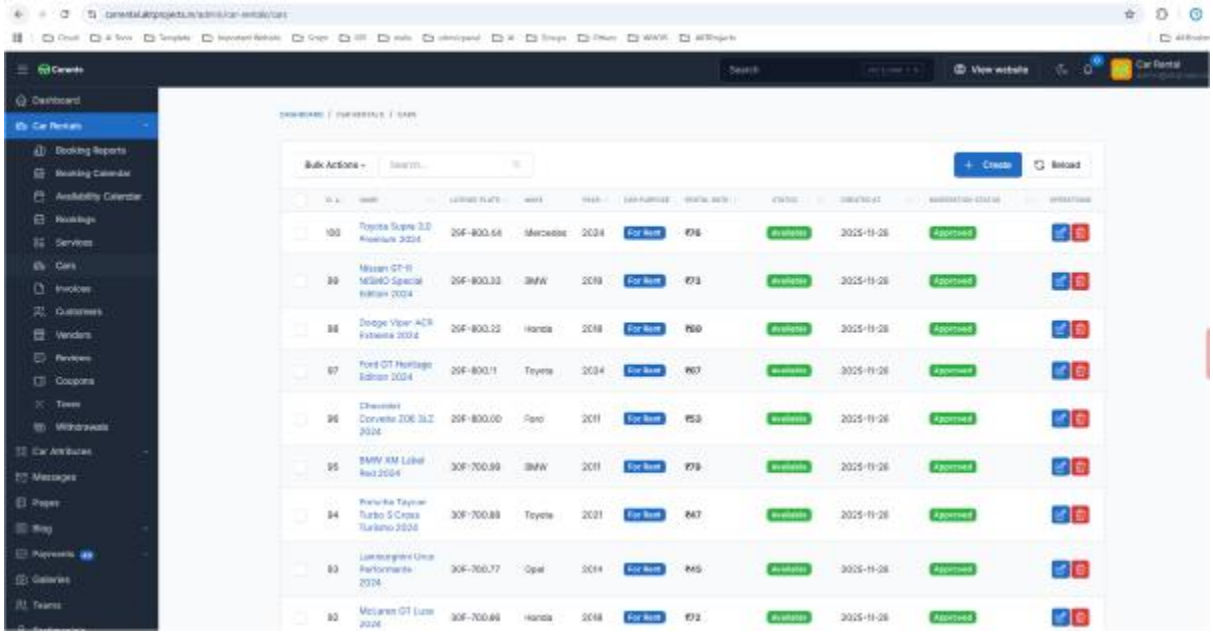

Dashboard
Car Rentals
Car Attributes
Messages
Pages
Blog
Payments
Galleries
Teams
Testimonials
Announcements
Comments
Contact
Simple Sliders
FAQs
Newsletters
Locations
Media
Appearance
Plugins
Tools
Search
View website
Car Rental
Manage Widgets
Themes
1
Users
1
Plugins
27
Pages
39
Recent Posts
Activity Logs


Dashboard
Car Rentals
Booking Reports
Booking Calendar
Availability Calendar
Bookings
Services
Cars
Invoices
Customers
Vendors
Reviews
Coupons
Taxes
Withdrawals
Car Attributes
Messages
Pages
Blog
Payments
Galleries
Teams
View website
Car Rental
Booking Summary
Booking Status Distribution
Recent Customers
Total
0
Status Breakdown
No customer data available


Booking Calendar
March 2026







5. CONCLUSION

The Online Peer-to-Peer Car Rental System successfully demonstrates how modern web technologies can be utilized to transform traditional vehicle rental services into a more flexible, efficient, and user-centric digital platform. By enabling direct interaction between car owners and renters, the system eliminates the limitations of centralized rental agencies and promotes better utilization of available resources. The integration of features such as vehicle listing management, booking automation, document verification, and role-based access control ensures a seamless and secure experience for all users.

The use of the Laravel framework, PHP 8.2+, and MySQL database, combined with deployment on a VPS or cloud server, provides a scalable and reliable infrastructure for real-world implementation. The system not only simplifies the rental process but also enhances transparency, reduces manual workload, and improves operational efficiency. Through a responsive and intuitive interface, users can easily access the platform from any device, making the system highly accessible and convenient.

In conclusion, this project offers a practical solution for the growing demand in the sharing economy by connecting vehicle owners with potential renters in a trusted environment. It lays a strong foundation for future enhancements such as online payment integration, GPS tracking, mobile applications, and AI-based recommendations. Overall, the system contributes to the advancement of digital transportation services and provides a sustainable and profitable model for peer-to-peer car rental businesses.

REFERENCES

1. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Doctoral Dissertation, University of California.
2. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
3. Pressman, R. S., & Maxim, B. R. (2019). Software Engineering: A Practitioner's Approach (9th Edition). McGraw-Hill.
4. Sommerville, I. (2016). Software Engineering (10th Edition). Pearson Education.
5. Laravel Documentation. (2025). Laravel Framework Official Guide. Available at: https://laravel.com/docs
6. PHP Documentation. (2025). PHP Manual. Available at: https://www.php.net/docs.php
7. MySQL Documentation. (2025). MySQL Reference Manual. Available at: https://dev.mysql.com/doc
8. Bootstrap Documentation. (2025). Bootstrap Framework Guide. Available at: https://getbootstrap.com
9. W3Schools. (2025). Web Development Tutorials. Available at: https://www.w3schools.com
10. AWS Documentation. (2025). Cloud Computing Services Guide. Available at: https://aws.amazon.com/documentation
11. DigitalOcean. (2025). VPS Hosting and Deployment Tutorials. Available at: https://www.digitalocean.com/community/tutorials
12. Stripe Documentation. (2025). Online Payment Integration Guide. Available at: https://stripe.com/docs
13. PayPal Developer Docs. (2025). Payment Gateway Integration. Available at: https://developer.paypal.com/docs
14. Open Web Application Security Project (OWASP). (2025). Web Security Guidelines. Available at: https://owasp.org
15. Zhang, T., & Chen, X. (2020). Peer-to-Peer Car Sharing Systems: Opportunities and Challenges. International Journal of Transportation Systems.