**Project Report on**

**Cab Rental Management System**



Submitted by

**ARGHYA BANERJEE**

**ADITYA LAHIRI**

***Bachelors of Technology***

***(CSE-AIML)***

**Batch 2022-2026**

**ABSTRACT**

This project report outlines the development of a web-based cab rental management system using HTML, CSS, PHP, and MySQL. The system is designed with two main user interfaces: an admin portal and a user portal. The admin interface allows for the management of cab records, including adding, viewing, and managing cab details and bookings. The user interface enables customers to browse available cabs, make bookings, and receive booking confirmations. The MySQL database is used to store user data, cab information, and booking records, ensuring efficient data handling and retrieval. This project demonstrates the integration of a front-end user experience with a back-end server and database, providing a complete and functional rental service solution. The report details the system architecture, database schema, and implementation process, as well as future improvements for enhanced functionality and security.

**ACKNOWLEDGEMENT**

We would like to express my special thanks to “Dr. DEEPSUBHRA GUHA ” as well as “Mrs Bipasha Mahato”, for their able guidance and support in completing our project. We would also extend our gratitude to the HOD “PROF. AMARTYA MUKHERJEE” for providing us with all the facilities that was required.

Aditya Lahiri

* B.Tech
* V-Semester

Arghya Banerjee

* B.Tech
* V-Semester

**CONTENTS**

**Sl No. Topics Page No. Remarks**

1. Introduction 5
2. System Description 10
3. Theoretical Analysis 12
4. Conclusion 21

**CHAPTER: 1 INTRODUCTION**

**Problem Definition:**

Traditional booking systems or manual processes are inefficient, prone to errors, and offer limited flexibility to customers. This results in issues like double bookings, poor inventory management, limited customer access to real-time availability, and reduced operational efficiency.

**Report Overview:**

This report provides a comprehensive overview of the development and design of a car rental system. The system is built to streamline car rental operations by automating tasks such as car inventory management, booking schedules, availability checks, and customer interactions, using a full-stack approach (HTML, CSS, JavaScript, PHP and MySQL).

1. **Database Design:**

The **car\_rental** database is designed to manage cars for a rental system. It primarily includes a **cars** table, which stores information about each car, such as:

**id**: Unique identifier for each car.

**car\_model**: Model of the car (e.g., "Toyota Corolla").

**year**: Year the car was manufactured.

**daily\_rate**: Cost per day to rent the car.

**is\_available**: Indicates if the car is currently available for rent.

This table allows for basic operations like checking if a car model is available and retrieving details about each car. Additional tables, like **bookings** and **customers**, could be added to manage rental reservations, track customer information, and record payments for a fully functional car rental system.

The data structures and relationships prioritize **data normalization**, ensuring that data is stored logically and efficiently. Relationships between cars and bookings help manage availability, and linking bookings to customers supports tracking of individual rental histories, setting a foundation for managing a scalable car rental system.

1. **System Functionality:**
   * Detailed description of the core functionalities:

The core functionalities of the car rental system include car availability checks, rental booking, customer management, and administrative features. Each feature is designed to streamline the car rental process from both customer and admin perspectives.

### Car Availability Check

**Description**: Allows customers to check if a specific car model is available for rental.

* **Input**: Car model name (e.g., "Toyota Corolla").
* **Process**: The system queries the cars table to find the car model and checks the is\_available field.
* **Output**: Returns a message indicating whether the car is available or unavailable.

**Purpose**: This functionality quickly informs the customer about car availability before booking, improving the user experience by reducing unnecessary booking attempts.

* Rental Booking

**Description**: Enables customers to reserve an available car for a specific period.

* **Input**: Car ID, customer details, rental start date, and end date.
* **Process**:
  + Confirms car availability during the specified period by cross-referencing the bookings table.
  + Creates a new entry in the bookings table with car\_id, customer\_id, start and end dates, and status as "pending" or "confirmed."
  + Updates the is\_available field in the cars table if the car is booked.
* **Output**: Confirms the booking with a booking ID and total rental cost (calculated by daily\_rate x rental duration).

**Purpose**: Allows customers to reserve a car, and the system can maintain a dynamic record of booked and available cars.

### 1. Database Connection and Query Execution

**Connection Setup**: To establish communication between the PHP application and the MySQL database, a database connection is necessary. The connection process allows the PHP code to send SQL queries to MySQL, which will execute the queries and return results or modify data.

$servername = "localhost";

$username = "root";

$password = ""; // MySQL password (usually empty in local setups)

$dbname = "car\_rental"; // Database name

// Create connection

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

// Check connection

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

}

echo "Connected successfully";

**Execution of Queries**: Once a database connection is established, we can execute SQL queries to retrieve, insert, update, or delete data from the database. Below are common database operations performed in a car rental system.

#### a. SELECT Query (Fetching Data)

A **SELECT** query is used to retrieve data from the database, such as car listings, customer information, or booking details. Here's how you can fetch and display car information:

$sql = "SELECT \* FROM cars"; // Replace 'cars' with your table name

$result = mysqli\_query($conn, $sql);

if (mysqli\_num\_rows($result) > 0) {

while($row = mysqli\_fetch\_assoc($result)) {

echo "Car ID: " . $row["id"] . " - Name: " . $row["name"] . " - Model: " . $row["model"] . "<br>";

}

} else {

echo "No cars found";

}

* **Benefits**:
  + The use of parameterized queries (? placeholders) helps prevent SQL injection attacks, enhancing security.
  + The MySQL2 library is a fast, reliable library for executing queries in Node.js and supports asynchronous callbacks for handling query results.

### 2. Data Insertion, Retrieval, and Modification

**Data Retrieval**: Retrieving data allows you to view the records stored in the database. For example, when a user views available cars for rental, you need to fetch data from the cars table. This operation involves using a SELECT SQL query.

#### Example: Retrieve All Cars

**Data Insertion and Modification**: Inserting data into the database allows new records to be added. For example, when a new car is added to the rental fleet, its details need to be stored in the database. This operation typically involves using an INSERT INTO SQL statement.

#### Example: Insert a New Car into the Database

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "car\_rental";

// Create connection

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

// Check connection

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

}

// Data to be inserted

$name = "Toyota";

$model = "Corolla";

$year = 2024;

// Insert query

$sql = "INSERT INTO cars (name, model, year) VALUES ('$name', '$model', $year)";

if (mysqli\_query($conn, $sql)) {

echo "New car added successfully";

} else {

echo "Error: ". $sql . "<br>" . mysqli\_error($conn);

}

mysqli\_close($conn);

?>

### 3. Error Handling and Exception Management

In PHP, error handling can be done using both **traditional error handling** methods (using functions like trigger\_error(), set\_error\_handler(), etc.) and **exception handling**. For modern PHP applications, **exception handling** is preferred because it allows for better control over errors, especially with operations that might fail, such as database interactions.

#### Basic Error Handling

* **Basic Error Handling**Error handling is typically done by checking for the existence of errors after acting. For example, when a database connection fails or a query doesn’t execute properly, you need to capture the error and handle it accordingly
* **Error Responses**: HTTP status codes and JSON responses are used to communicate error states to the client. For example, if a car model is not found, the code returns a message indicating "Car not found" instead of leaving it unhandled.
* **Limitations**:
  + The error handling is minimal, without detailed exception management. Errors are only logged in the response to the client; additional logging (e.g., to an external logging service) would provide better debugging and error tracking.
  + Throw in db.connect() might cause the application to crash if the database connection fails, making the application less resilient. A better approach would be to retry the connection or send a graceful error response, allowing the server to remain operational.

**CHAPTER 2:**

**SYSTEM DESCRIPTION**

The car rental system is a web-based application developed to manage car rental operations efficiently by automating and centralizing key tasks. It consists of two main interfaces: one for customers and another for administrators (admins). Built using HTML, CSS, JavaScript, PHP, and MySQL, this system provides functionalities for car search, booking, inventory management, and user account handling, delivering a seamless experience for users and operational control for admins.

The system allows users to:

1. **Browse available cars:**
   * Users can view a list of all available cabs, including details such as the car model, number plate, and rental price per day.
2. **Cab Booking:**
   * Users have the option to book a cab for specific dates by selecting the desired vehicle and inputting the required booking and return dates.
3. **Booking Confirmation:**
   * After successfully booking a cab, users receive a booking confirmation that summarizes their reservation details.

**Key Components:**

* **Frontend Interface:**
  + **HTML Pages**: Provide the structure and content of the website, including the home page, cab listings, and booking forms.
  + **CSS Styling**: Ensures a visually appealing and responsive design, improving user experience across devices.
  + **Forms and Navigation**: Includes input forms for booking, search features, and navigation bars for seamless movement between pages.
* **Backend Logic:**
  + **PHP Scripts**: Handle server-side logic for form submissions, database interactions, session management, and user authentication.
  + **Validation**: Includes server-side input validation to ensure data integrity and security before processing user data.
* **Database (MySQL):**
  + **Users Table**: Stores user information such as usernames, passwords (hashed), and roles (e.g., admin or user).
  + **Cabs Table**: Contains details about each cab, including model, number plate, price per day, and availability status.
  + **Bookings Table**: Tracks booking records, linking users to specific cabs and storing dates for rental periods.

This system offers a convenient way to organize and book, making it a valuable tool for home cooks and food enthusiasts.

**CHAPTER 3:**

**THEORETICAL ANALYSIS**

### Core Functionalities

The system primarily revolves around four core functionalities:

* **Cab Browsing and Search:**
  + **Overview:** Allows users to view available cabs, including relevant details such as model, number plate, rental price, and availability.
  + **Elaboration:** Users can filter and search for cabs based on criteria such as price range or vehicle type. This feature ensures that users can quickly find a cab that meets their specific needs.
  + **Implementation:** Data is fetched from the cabs table in the MySQL database and displayed dynamically using PHP.
* **Cab Booking:**
* **Overview**: Enables users to reserve a cab for a specific period.
* **Elaboration**: Users can select a cab and specify booking and return dates. The system checks the availability of the selected cab and confirms the booking if the vehicle is available.
* **Implementation**: A form captures user input, and PHP scripts process the booking by inserting a new record into the bookings table and updating the cab’s availability status.
* **Booking Confirmation:**
* **Overview:** Provides users with confirmation details after successfully booking a cab.
* **Elaboration:** Displays the booking ID, cab details, and rental period to ensure users have all the necessary information for their rental.
* **Implementation:** The confirmation page retrieves booking details from the database using PHP after the booking is successfully saved.
* **Cab management(Admin):**
* **Overview**: Allows the admin to manage the fleet of cabs available for rent.
* **Elaboration**: Admins can add new cabs, update existing cab details (e.g., price or availability status), and remove cabs that are no longer in service. This ensures the database reflects the current fleet.
* **Implementation**: PHP forms and backend logic interact with the cabs table to perform updates, deletions, and insertions.

### Technical Implementation

**Cab Browsing and Search:**

* Use **HTML** and **CSS** to create a page that lists available cabs.
* Implementation of **search and filter forms** using basic HTML.

<div class="leftContent">

<b><form action="available.php" method="post" >

<h1> BOOK A CAR </h1><br><br>

Enter Customer ID: <input type="text" name="cbid"/><br><br>

Enter date yyyy-mm-dd: <input type="text" name="Sdate"/><br><br>

Select Car type: <select name = "Ctype">

<option>Compact</option>

<option>Medium</option>

<option>Large</option>

<option>SUV</option>

<option>Van</option>

<option>Truck</option>

</select><br><br>

Rent type: <select name = "Rtype">

<option>Daily</option>

<option>Weekly</option>

</select><br><br>

No of days : <select name = "Days">

<option>0</option><option>1</option><option>2</option><option>3</option><option>4</option><option>5</option><option>6</option><option>7</option>

</select><br><br>

No of weeks : <select name = "Weeks">

<option>0</option><option>1</option><option>2</option><option>3</option>

</select><br><br>

<input type="submit" name="check" value="Check Availability"><br><br>

Enter Vehicle id: <input type="text" name="Vehicleid"/><br><br>

<input type="submit" name="book" value="Book">

</form></b>

<br>

<b><form action="viewrental.php" method="post" >

<h2> VIEW ACTIVE & SCHEDULED RENTALS </h2>

<input type="submit" name="submit1" value="View">

</form></b>

</div>

**PHP Backend:**

* Use PHP to query the cabs table in the MySQL database.
* Construct SQL queries to display all available cabs or filter results based on user input.

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<body>

<?php

$database\_host = "localhost";

$database\_user = "rajat";

$database\_pass = "pizza";

$database\_name = "carrental";

$connection = mysqli\_connect($database\_host, $database\_user, $database\_pass, $database\_name);

if(mysqli\_connect\_errno()){

die("Failed connecting to MySQL database. Invalid credentials" . mysqli\_connect\_error(). "(" .mysqli\_connect\_errno(). ")" ); }

$Ctype=$\_POST["Ctype"];

$res="SELECT Vehicle\_id,License\_no,Model,Year,Drate AS Daily\_Rate, Wrate AS Weekly\_Rate FROM car AS C JOIN rates AS R

WHERE C.Ctype=R.Ctype AND C.Ctype='$Ctype'";

$result=mysqli\_query($connection,$res);

echo "<h1><center>".$Ctype."&nbsp;Cars</h1><br><br>";

?>

<center>

<table border='1'>

<tr>

<th>Vehicle ID</th>

<th>License No</th>

<th>Model</th>

<th>Year</th>

<th>Daily Rate</th>

<th>Weekly Rate</th>

</tr>

<?php

if (mysqli\_num\_rows($result) > 0) {

while($row = mysqli\_fetch\_assoc($result))

{

echo "<tr>";

echo "<td>" . $row["Vehicle\_id"] . "</td>";

echo "<td>" . $row["License\_no"] . "</td>";

echo "<td>" . $row["Model"] . "</td>";

echo "<td>" . $row["Year"] . "</td>";

echo "<td>" . $row["Daily\_Rate"] . "</td>";

echo "<td>" . $row["Weekly\_Rate"] . "</td>";

echo "</tr>";

}

}

?>

</table>

</body>

</html>

### Security Considerations

* **Database Security:**
  + **Minimal Privileges**: Configured database user accounts with minimal privileges needed for the operations they perform.
  + **Regular Backups**: Implemented regular database backups and ensured they are securely stored and encrypted.
* **Data Encryption:**
  + **Encrypt Sensitive Data**: Used encryption for sensitive data stored in the database, such as personal user information.
  + **Transport Layer Encryption**: Used SSL/TLS to encrypt data in transit between the client and server.
* **Error Handling and Logging:**
  + **Error Messages**: Avoid exposing detailed error messages to users that might reveal sensitive information about the database or server.
  + **Logging**: Log critical events (e.g., failed login attempts, unauthorized access) to monitor for potential security breaches. Ensure logs are stored securely and only accessible by authorized personnel.

### Potential Enhancements

### 1. Mobile App Integration

* **Development of a Mobile App**: Create iOS and Android applications that sync with the website for seamless booking and management on the go.
* **Push Notifications**: Notify users of booking confirmations, updates, promotions, and reminders.

### 2. Advanced Booking Features

* **Dynamic Pricing**: Implement algorithms to adjust pricing based on demand, peak hours, or special days.
* **Multi-City Booking**: Allow users to book cabs across multiple cities with one reservation.
* **Ride Scheduling**: Enable users to schedule rides for a future date and time.

### 3. AI-Powered Features

* **Recommendation Engine**: Use machine learning to recommend cab options based on past user behavior and preferences.
* **Chatbots**: Implement AI-driven chatbots for 24/7 customer support to handle common queries, bookings, and cancellations.

### 4. Payment Enhancements

* **Multiple Payment Options**: Integrate various payment gateways including digital wallets (e.g., PayPal, Google Pay, Apple Pay).
* **Partial Payments and Deposits**: Allow users to pay a deposit and settle the remaining balance later.
* **Split Payments**: Enable users to split fare payments with friends or family.

### 5. Real-Time Tracking

* **Live GPS Tracking**: Integrate real-time GPS tracking of the cab during the rental period.
* **ETA Updates**: Provide estimated time of arrival and live updates to users during the trip.

### 6. User Feedback System

* **Rating and Review System**: Enable users to leave ratings and reviews for vehicles and drivers.
* **Driver Feedback Dashboard**: Provide feedback directly to drivers for service improvement.

By addressing these aspects, the Cab rental management system can evolve into a robust and user-friendly tool for daily commuters.

**Chapter 4:**

**CONCLUSION**

The development of the cab rental management website has successfully addressed the primary objectives of providing an efficient, user-friendly platform for booking and managing cab rentals. Through the integration of key functionalities such as user and admin routes, secure authentication, real-time booking management, and seamless payment processing, the system offers an effective solution for both end-users and administrators.

This project not only showcases the potential for automating and streamlining cab rental services but also highlights the importance of adopting robust technical implementations and security measures to ensure data safety and user trust. The website’s design, built using HTML, CSS, PHP, and MySQL, strikes a balance between functionality and simplicity, facilitating ease of use while maintaining a high level of performance.

Potential enhancements outlined in the report, such as mobile app integration, advanced booking features, and AI-powered tools, lay the foundation for future scalability and increased user engagement. By continuing to evolve and implement these suggested improvements, the platform can maintain its competitive edge and adapt to changing user expectations.

In conclusion, this project demonstrates a comprehensive approach to creating a digital cab rental system that not only meets current needs but is designed with scalability and future advancements in mind. The development process, lessons learned, and strategic enhancements outlined serve as a roadmap for continuous improvement and long-term success in the cab rental industry.