**BIKE AND SCOOTY RENTAL WEBSITE**

**ABSTRACT**

This website is a sophisticated web-based platform designed for convenient and efficient bike and scooty rental services. Built using HTML, CSS, Bootstrap, PHP, and MySQL, this website offers a user-friendly interface for customers to browse, select, and rent bike and scootys seamlessly.

The website's frontend is crafted using HTML, CSS, and Bootstrap, ensuring a responsive design across various devices. Users can explore a wide range of available vehicles, filtering by model, price range, or specific features. The intuitive interface allows for easy navigation and selection of desired bike and scootys.

The backend functionalities are powered by PHP and MySQL, enabling robust database management and dynamic content delivery. Users can create accounts, log in securely, and manage their profiles. The website employs secure authentication protocols to safeguard user data.

For bike and scooty rental transactions, This system integrates a booking system that allows users to select rental dates, view pricing details, and make secure payments online. The system tracks available bike and scootys in real-time, ensuring accurate availability for customers.

Administrators have access to a comprehensive dashboard to manage vehicle listings, rental schedules, user accounts, and payment processing. They can efficiently update bike and scooty information, track rental history, and generate insightful reports.

This website aims to provide a seamless and secure experience for both customers and administrators, ensuring hassle-free bike and scooty rentals while maintaining data integrity and security. This project showcases the integration of HTML, CSS, Bootstrap for the frontend, and PHP, MySQL for the backend, emphasizing functionality, usability, and reliability.

**INTRODUCTION**

In today's fast-paced world, accessibility to convenient and reliable transportation options is paramount. This website emerges as a solution, offering a sophisticated and user-friendly interface for individuals seeking efficient bike and scooty rental services.

Developed with a focus on seamless user experience and robust functionalities, this website harnesses the power of HTML, CSS, Bootstrap, PHP, and MySQL to create a comprehensive web-based solution. This platform caters to both customers looking for rental options and administrators managing the rental fleet.

The objective of this website is to provide an intuitive and secure online space where users can effortlessly browse, select, and rent bike and scootys according to their preferences. With an emphasis on responsive design and ease of navigation, the frontend design ensures accessibility across devices, allowing users to explore a diverse range of available vehicles with ease.

Furthermore, the backend functionalities, driven by PHP and MySQL, establish a solid foundation for managing the platform's dynamic content. Security measures are implemented to safeguard user data, ensuring a safe and reliable environment for transactions and interactions within the system.

This website facilitates a streamlined booking process, enabling users to select rental dates, view pricing details, and complete secure online payments. Real-time tracking of available vehicles ensures accurate availability information for customers, enhancing their overall experience.

Administrators are equipped with a comprehensive dashboard to efficiently manage vehicle listings, rental schedules, user accounts, and financial transactions. The system empowers them to update bike and scooty information, monitor rental history, and generate insightful reports for informed decision-making.

Through the integration of HTML, CSS, Bootstrap, PHP, and MySQL, this website aims to redefine the bike and scooty rental experience, placing emphasis on functionality, security, and user satisfaction. This introduction sets the stage for a detailed exploration of how this multifaceted platform operates and enhances the bike and scooty rental industry.

**SYSTEM ANALYSIS**

**EXISTING SYSTEM**

1. **Physical Rental Offices:**
   * Bike and scooty rental companies maintained physical offices or branches in various locations such as airports, downtown areas, or specific neighborhoods.
   * Customers interested in renting a bike and scooty visited these offices in person to make reservations, check vehicle availability, and complete rental agreements.
2. **Telephone Reservations:**
   * Prior to the widespread use of the internet, customers often made reservations by calling the rental agency's phone number.
   * Rental agents handled inquiries, checked availability, and booked vehicles over the phone. Customers provided their details and rental preferences during these conversations.
3. **Printed Catalogs and Flyers:**
   * Bike and scooty rental companies distributed printed catalogs or flyers showcasing their available vehicle models, rental rates, terms, and contact information.
   * Customers referred to these materials to explore available options and make rental decisions.
4. **Manual Paperwork and Agreements:**
   * Rental agreements, documentation, and paperwork were manually processed at the rental office.
   * Customers filled out forms detailing their personal information, rental duration, insurance coverage, and any additional services required.
5. **Walk-in Rentals:**
   * Some customers simply walked into rental offices without prior reservations, selecting available bike and scootys based on current availability.
6. **In-Person Payment and Pickup:**
   * Rental payments were made in person at the rental office using cash, checks, or credit/debit bike and scootyds.
   * Customers physically picked up the rented vehicle from the rental location after completing paperwork and payment.
7. **Manual Inventory Management:**
   * Rental companies managed their vehicle inventory manually, updating availability status based on in-person reservations and returns.
   * Records were kept using physical logbooks or systems involving paperwork.
8. **Customer Service and Disputes:**
   * Customer service and dispute resolution largely occurred in person or via phone calls, with customers contacting the rental office for inquiries, changes, or issues with their rentals.

**PROPOSED SYSTEM**

The system of this research work was web-based. PHP (Personal Home Page) is a scripting language and an interpreter that is used to develop the website.

**Benefits:**

1. **User Interface and Frontend Development:**
   * Develop a responsive and visually appealing frontend interface using HTML, CSS, and Bootstrap.
   * Ensure an intuitive and user-friendly design allowing customers to easily browse available bike and scootys, view details, and initiate rental requests.
2. **Backend Development with PHP:**
   * Implement PHP scripting for server-side functionalities, handling user authentication, managing sessions, and communicating with the database.
   * Utilize PHP to interact with MySQL database for storing and retrieving data related to users, vehicles, bookings, and transactions securely.
3. **Vehicle Listings and Rental Process:**
   * Create a comprehensive catalog displaying available vehicles with detailed information including make, model, pricing, features, and images.
   * Enable users to search, filter, and select bike and scootys based on preferences such as location, date range, vehicle type, and price.
4. **User Account Management:**
   * Provide user registration and login features for account creation and secure authentication.
   * Allow users to manage their profiles, view rental history, update personal information, and handle bookings.
5. **Booking and Reservation System:**
   * Develop a booking system enabling users to select rental dates, calculate pricing, and securely make reservations online.
   * Integrate secure payment gateways to process online transactions for rental fees.
6. **Administrative Dashboard:**
   * Design an admin panel for efficient management of the rental inventory.
   * Enable administrators to add, edit, or remove vehicle listings, update pricing, monitor reservations, and manage user accounts.
7. **Security Measures:**
   * Implement robust security protocols such as encryption, secure authentication mechanisms, and input validation to safeguard user data and prevent security breaches.
8. **Error Handling and Support:**
   * Incorporate error handling mechanisms to manage exceptions and errors effectively.
   * Provide customer support channels for user inquiries, feedback, and assistance.

**MODULE DESCRIPTION**

1. **User Authentication Module:**
   * Manages user registration, login, and logout functionalities.
   * Implements secure password storage and retrieval.
   * Differentiates between different user roles such as customers and administrators.
2. **Bike and scooty Listing and Search Module:**
   * Displays a catalog of available rental bike and scootys with details such as make, model, year, and rental rates.
   * Implements a search feature allowing users to filter and find specific bike and scootys based on criteria such as brand, type, or price range.
   * Provides an intuitive interface for users to view images and specifications of each bike and scooty.
3. **Booking and Reservation Module:**
   * Allows customers to select a rental bike and scooty and specify rental dates and times.
   * Validates and confirms reservations based on bike and scooty availability and rental policies.
   * Provides a calendar view to show bike and scooty availability and prevent overlapping reservations.
4. **User Profile and Booking History Module:**
   * Enables customers to create and manage their profiles with personal information and preferences.
   * Allows customers to view their booking history, upcoming reservations, and rental details.
   * Provides options for customers to update contact details or cancel reservations.
5. **Admin Dashboard Module:**
   * Provides administrators with a centralized dashboard for managing bike and scooty inventory and reservations.
   * Allows administrators to add, edit, or remove bike and scooty listings.
   * Enables administrators to view and manage customer reservations, including the ability to approve or reject bookings.
6. **Feedback and Review Module:**
   * Allows customers to submit feedback and reviews after completing their rental.
   * Displays customer testimonials and ratings on the website.
   * Enables administrators to respond to customer feedback and address concerns.
7. **Security Module:**
   * Implements robust security measures to protect user data and system integrity.
   * Manages access control to ensure users have appropriate permissions.
   * Regularly updates security protocols to address potential vulnerabilities.
8. **Database Management Module:**
   * Manages the storage and retrieval of bike and scooty rental-related data using MySQL databases.
   * Ensures data integrity, reliability, and security.
   * Implements backup and recovery mechanisms to prevent data loss.

This modular breakdown provides a foundation for the development of a comprehensive and user-friendly bike and scooty rental website. The use of HTML, CSS, Bootstrap, PHP, and MySQL ensures a responsive and dynamic web application that caters to the needs of both customers and administrators in the bike and scooty rental industry.

**SYSTEM SPECIFICATION**

**HARDWARE SPECIFICATION**

|  |  |
| --- | --- |
| System | HP 15s |
| Processor | Ryzen 5 2.1 GHz |
| Storage | 512 GB SSD |
| RAM | 16 GB |
| Monitor | Integrated Monitor |
| Mouse | Integrated Trackpad |
| Keyboard | Integrated Keyboard |

**OPERATING SYSTEM**

|  |  |
| --- | --- |
| Operating System | Windows 11 |
| Front End | HTML and Bootstrap |
| Back End | MySQL Version 8 |
| Server | XAMPP |

**SOFTWARE SPECIFICATION**

**SOFTWARE DESCRIPTION**

**XAMPP:**

XAMPP is an [open-source](https://en.wikipedia.org/wiki/Free_software) [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [web server](https://en.wikipedia.org/wiki/Web_server) [solution stack](https://en.wikipedia.org/wiki/Solution_stack) package developed by Apache Friends, consisting mainly of the [Apache HTTP Server](https://en.wikipedia.org/wiki/Apache_HTTP_Server), [Maria DB](https://en.wikipedia.org/wiki/MariaDB) [database](https://en.wikipedia.org/wiki/Database), and [interpreters](https://en.wikipedia.org/wiki/Interpreter_%28computing%29) for scripts written in the [PHP](https://en.wikipedia.org/wiki/PHP) and [Perl](https://en.wikipedia.org/wiki/Perl) [programming languages](https://en.wikipedia.org/wiki/Programming_language). XAMPP stands for Cross-Platform (X), Apache (A), Maria DB (M), PHP (P), and Perl (P). It is a simple, lightweight Apache distribution that makes it extremely easy for developers to create a local web server for testing and deployment purposes.

Everything needed to set up a web server – server application (Apache), database (Maria DB), and scripting language (PHP) – is included in an extractable file. XAMPP is also cross-platform, which means it works equally well on Linux, Mac, and Windows.

XAMPP's designers intended it for use only as a development tool, to allow website designers and programmers to test their work on their computers without any access to the Internet.

**CROSS-PLATFORM**

Cross-platform software is a type of software application that works on multiple operating systems or devices, which are often referred to as platforms. A platform means an operating system such as Windows, Mac OS, Android, or iOS. When a software application works on more than one platform, the user can utilize the software on a wider choice of devices and computers.

**BENEFITS OF CROSS-PLATFORM**

The benefit of a cross-platform software app or program is that you can use the same program whether you’re on a Windows PC or whether you’re logging in from your laptop or smartphone. The Microsoft Office suite of applications, which includes Word, Excel, and PowerPoint, is available on Windows, Mac OS, iOS (iPhone/iPad), and Android. While there are differences based on how the platforms work, you’ll have a similar experience within the application between all of your devices.

Having a similar experience across any platform means there’s a much smaller learning curve if one even exists at all, so you’ll be more productive and be able to use a software product you’re familiar with regardless of the operating system or device you choose. In addition, your files can be moved much more easily between your devices so you can use the software with whatever device you have with you at the time. And there’s a way to keep all of your work in sync across all of your devices, by using the cloud.

**EXAMPLES OF CROSS-PLATFORM**

**Unity 3D**

First, let’s talk about Unity3D. I think the game engine should be preferred by people who want to write mobile games.  
You can develop games on 17 platforms using multiple languages, including Linux. Of course, iOS, Android, and Windows Phone is also the most ideal game engine to develop games.

You can develop your application using C #, JS, and C ++.

Link to: [https://unity3d.com](https://unity3d.com/)

# Xamarin

Xamarin Some time ago, it was purchased by Microsoft and is a perfect fit for developers using C #.

Because it is a C # language, it has a lot of documentation, and because of Microsoft support, Xamarin is the choice for C # developers.

In addition, you can do everything you can do in Objective-C, Swift, and Java with the Xamarin library.

Link to: [https://xamarin.com](https://xamarin.com/)

# React Native

React Native is an open-source JavaScript library developed by the new generation of React — Facebook, which was open to Github in 2013. Native application creation means writing applications only for a specific operating system. React Native helps developers reuse their code over the web and on mobile. Developers will not have to create the same app from scratch for iOS and Android. They will be able to reuse the code in each operating system. The great thing about React Native is that there is little difference between a finished application in Objective-C or Java and an application built using React Native. Android and iOS code development environments are very different. So it takes time to remove the application to two different platforms. However, with React Native, only one developer can write on different mobile operating systems.

**APACHE:**

The Apache HTTP Server, colloquially called Apache is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [web server](https://en.wikipedia.org/wiki/Web_server) software, released under the terms of [Apache License](https://en.wikipedia.org/wiki/Apache_License) 2.0. Apache is developed and maintained by an open community of developers under the auspices of the [Apache Software Foundation](https://en.wikipedia.org/wiki/Apache_Software_Foundation).

The vast majority of Apache HTTP Server instances run on a [Linux distribution](https://en.wikipedia.org/wiki/Linux_distribution), but current versions also run on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [OpenVMS](https://en.wikipedia.org/wiki/OpenVMS),  and a wide variety of [Unix-like](https://en.wikipedia.org/wiki/Unix-like) systems. Past versions also ran on [NetWare](https://en.wikipedia.org/wiki/NetWare), [OS/2](https://en.wikipedia.org/wiki/OS/2), and other operating systems,  including ports to mainframes.

Originally based on the HTTP server, the development of Apache began in early 1995 after work on the NCSA code stalled. Apache played a key role in the initial growth of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), quickly overtaking NCSA HTTP as the dominant [HTTP](https://en.wikipedia.org/wiki/HTTP) server. In 2009, it became the first web server software to serve more than 100 million [websites](https://en.wikipedia.org/wiki/Website). As of January 2021, [Netcraft](https://en.wikipedia.org/wiki/Netcraft" \o "Netcraft) estimated that Apache served 24.63% of the million busiest websites, while [Nginx](https://en.wikipedia.org/wiki/Nginx) served 23.21% and Microsoft is in third place at 6.85% (for some of Netcraft's other stats Nginx is ahead of Apache), while according to W3Techs, Apache is ranked first at 35.0% and Nginx second at 33.0% and Cloudflare Server third at 17.3%.

**LANGUAGE SPECIFICATION**

**PHP**

**INTRODUCTION OF PHP**

PHP started as a small open-source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

* PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
* PHP is a server-side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, and even build entire e-commerce sites.
* It is integrated with several popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
* PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
* PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
* PHP is forgiving: PHP language tries to be as forgiving as possible.
* PHP Syntax is the same as C language.

**What is a PHP File?**

* PHP files can contain text, HTML, CSS, JavaScript, and PHP code.
* PHP code is executed on the server, and the result is returned to the browser as plain HTML.
* PHP files have the extension ".php".

**What Can PHP Do?**

* PHP can generate dynamic page content and it can create, open, read, write, delete, and close files on the server and it can collect form data.
* PHP can send and receive cookies it can add, delete, and modify data in your database and it can be used to control user-access and encrypt data.

**Why PHP?**

* PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.).
* PHP is compatible with almost all servers used today (Apache, IIS, etc.).
* PHP supports a wide range of databases.
* PHP is free.
* PHP is easy to learn and runs efficiently on the server side.

## **What is a Database?**

* A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching, and replicating the data it holds.
* Other kinds of data stores can be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.
* Nowadays, we use relational database management systems (RDBMS) to store and manage huge volumes of data. This is called a relational database because all the data is stored in different tables and relations are established using primary keys or other keys known as foreign keys.

**MySQL Database**

* MySQL is released under an open-source license. So you have nothing to pay to use it. MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language. MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc. MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development. MySQL supports large databases, up to 50 million rows or more in a table.
* The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB). MySQL is customizable.
* The open-source GPL license allows programmers to modify the MySQL software to fit their specific environments.

**TABLE CREATION**

* Name of the table
* Names of fields
* Definitions for each field
* Field Attribute **NOT NULL** is being used because we do not want this field to be NULL. So if the user tries to create a record with a NULL value, then MySQL will raise an error.
* Field Attribute **AUTO\_INCREMENT** tells MySQL to go ahead and add the next available number to the id field.
* Keyword **PRIMARY KEY** is used to define a column as the primary key. You can use multiple columns separated by a comma to define a primary key.

## **ADMINISTRATIVE MYSQL COMMAND**

* **USE DATABASE NAME**: This will be used to select a particular database in the MySQL work area.
* **SHOW DATABASES:** Lists the databases that are accessible by the MySQL DBMS.
* **SHOW TABLES:** Shows the tables in the database once a database has been selected with the use command.
* **SHOW COLUMNS FROM Table name:** Shows the attributes, types of attributes, key information, whether NULL is permitted, defaults, and other information for a table.
* **SHOW INDEX FROM Table name:** Presents the details of all indexes on the table, including the PRIMARY KEY

## **CREATING TABLES USING PHP SCRIPT:**

To create a new table in any existing database you would need to use PHP function **mysqli\_query()**.

## **Dropping Tables Using PHP Script:**

Drop an existing table in any database, you would need to use the PHP function **mysqli\_query()**.

## **INSERTING DATA USING PHP SCRIPT:**

**CREATE**

Create table statement is used to create a table in MySQL.

**SELECT**

The SELECT statement is used to select data from one or more tables.

**UPDATE**

The UPDATE statement is used to update existing records in a table:

## **DELETE**

The DELETE statement is used to delete records from a table:

**DATABASE DESIGN:**

The data in the system has to be stored and retrieved from the database. Designing the database is part of system design. Data elements and data structures to be stored have been identified at the analysis stage.

They are structured and put together to design the data storage and retrieval system. A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently.

The general objective is to make database access easy, quick, inexpensive, and flexible for the user. Relationships are established between the data items and unnecessary data items are removed.

Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data storage required, minimizing chances of data inconsistencies, and optimizing for updates.

**INPUT DESIGN**

The Input design is the main feature of the system. Input design determines the format and validation criteria for data entering the system. Inputs originate with end-users; human factors play a significant role in input design. The input design is designed to control the input, avoid delay, and errors in data, avoid extra steps, to keep the process simple. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps, and keeping the process simple. The input is designed in such a way that it provides security and ease of use while retaining privacy.

The following are the general principles, that are considered in designing inputs,

* + - Enter only variable data
    - Do not input data that can be calculated
    - List of values
    - Sequence entry

**OUTPUT DESIGN**

Designing the output is more important than working up with a few layout charts and reports. The outputs are designed based on the issue encountered. It will also take bike and scootye of who will receive the output, what for it is produced how many details are needed, when it is needed, and by what method.

The outputs designed in this system are easy to use and useful for their jobs. The outputs are simple to read and interpret. The outputs obtained from this system are designed by using a few guidelines, which are given below. The information should be clear and accurate, yet concise and restricted to relevant data. Reports should have titles, data, and descriptive headings for columns of data, numbered pages, and so on.

**SYSTEM TESTING**

System testing is the process of exercising software with the intent of finding and ultimately correcting errors. This fundamental philosophy does not change for web applications, because Web-based systems and applications reside on a network and interoperate with many different operating systems, browsers, hardware platforms, and communication protocols; the search for errors represents a significant challenge for web applications.

The distributed nature of client/server environments, the performance issues associated with transaction processing, the potential presence of several different hardware platforms, the complexities of network communication, the need to serve multiple clients from a centralized database, and the requirements imposed on the server all combine to make testing of client\server architectures.

Testing issues

* Client GUI considerations
* Target environment and platform diversity considerations
* Distributed database considerations
* Distributed processing considerations

**TYPES OF TESTING**

1. Unit Testing

2. Integration Testing

3. Validation Testing

4. User Acceptance Testing

5. System Testing

**Unit Testing**

All modules were tested and individually as soon as they were completed were checked for their correct functionality. Unit testing is bike and scootyried out by verifying and recovering errors within the boundary of the smallest unit or a module. In this testing step, each module was found to be working satisfactorily per the expected output of the module. In the package development, each module is tested separately after it has been completed and checked with valid data.

**Integration Testing**

The entire project was split into small programs; each of these single programs gives a frame as an output. These programs were tested individually; at last, all these programs were combined by creating another program where all these constructions were used. It causes a lot of problems by not functioning in an integrated manner.

The user interface testing is important since the user has to declare that the arrangements made in the frames are convenient and it is satisfied. When the frames are tested, the end user gives suggestions. Since they were much exposed to do the work manually.

**Validation Testing**

At the culmination of the black box testing software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of tests i.e., validation succeeds when the software functions in a manner that can be reasonably accepted by the customer.

**User Acceptance Testing**

User acceptance testing of the system is the key factor in the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective systems at the time of development and making changes whenever required. This is done concerning the input screen design and output screen design.

**System Testing**

This is to verify that all the system elements have been properly integrated and perform allocated functions. Testing is executing a program to test the logic changes made in it to find errors. Tests are also conducted to find discrepancies between the system and its original objective, current specifications, and documents.

**SYSTEM IMPLEMENTATION**

Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system & and giving the user confidence that the new system will work efficiently & and effectively in the implementation stage.

The stage consists of

* + - Testing the developed program with simple data.
    - Detections and correction of errors.
    - Creating whether the system meets user requirements.
    - Testing whether the system.
    - Making necessary changes as desired by the user.
    - Training user personnel.

**Implementation Procedures**

The implementation phase is less creative than the system design. A system project may be dropped at any time before implementation, although it becomes more difficult when it goes to the design phase.

The final report to the implementation phase includes procedural flowcharts, record layouts, report layouts, and a workable plan for implementing the candidate system design into an operational one. Conversion is one aspect of implementation.

**System Maintenance**

Maintenance is the implementation of the review plan. As important as it is, many programmers and analysts are to perform or identify themselves with the maintenance effort. There are psychological, personality, and professional reasons for this. Analysts and programmers spend far more time maintaining programs than they do writing them. Maintenance accounts for 50-80 percent of total system development.

Maintenance is expensive. One way to reduce maintenance costs is through maintenance management and software modification audits.

* Maintenance is not as rewarding or exciting as developing systems. It is perceived as requiring neither skill nor experience.
* Users are not fully cognizant of the maintenance problem or its high cost.
* Few tools and techniques are available for maintenance.
* A good test plan is lacking.
* Standards, procedures, and guidelines are poorly defined and enforced.
* Programs are often maintained without bike and scootye for structure and documentation.
* There are minimal standards for maintenance.
* Programmers expect that they will not be in their current commitment by the time their programs go into the maintenance cycle.

**SYSTEM DESIGN**

System design is "the process of studying a procedure or business to identify its goals, purposes and create systems and procedures that will efficiently achieve them". Another view sees system analysis as a problem-solving technique that breaks down a system into its component pieces for the study of how well those parts work and interact to accomplish their purpose.

The field of system analysis relates closely to requirements analysis or operations research. It is also "an explicit formal inquiry bike and scootyried out to help a decision maker identify a better course of action and make a better decision than they might otherwise have made."

* **DESIGN NOTATION**

Design notations are used when planning and should be able to communicate the purpose of a program without the need for formal code. Commonly used design notations are:

* DFD
* ERD
* **DFD (DATA FLOW DIAGRAM):**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an [information system](https://en.wikipedia.org/wiki/Information_system), modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the [visualization](https://en.wikipedia.org/wiki/Data_visualization) of [data processing](https://en.wikipedia.org/wiki/Data_processing) (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about the timing of the process or information about whether processes will operate in sequence or parallel, unlike a [flowchart](https://en.wikipedia.org/wiki/Flowchart) which also shows this information.

Data flow diagrams were popularized in the late 1970s, arising from the book Structured Design, by computing pioneers Ed Yourdon and Larry Constantine. They based it on the “data flow graph” computation models by David Martin and Gerald Estrin. The structured design concept took off in the software engineering field, and the DFD method took off with it. It became more popular in business circles, as it was applied to business analysis than in academic circles.

**DFD SYMBOLS**

The process that transforms data flow

Source or Destination of Data

Data Flow

Data source

**ENTITY RELATIONSHIP DIAGRAM**

The relation upon the system is structured through a conceptual ER-Diagram, which not only specifies the existential entities but also the standard relations through which the system exists and the bike and scootydinalities that are necessary for the system state to continue. The Entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the data modeling activity The attributes of each data object noted in the ERD can be described resign a data object description.

The set of primary components that are identified by the ERD are

* + Data object
  + Relationships
  + Attributes
  + Various types of indicators

The primary purpose of the ERD is to represent data objects and their relationships.

**ER-DIAGRAM SYMBOL**

Entity

Relationship

Flow

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specifications and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps, and keeping the process simple. The input is designed in such a way that it provides security and ease of use while retaining privacy.

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system. It is achieved by creating user-friendly screens for the data entry to handle large volumes of data.

The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulations can be performed. It also provides record viewing facilities. When the data is entered it will check for its validity. Data can be entered with the help of screens.

**DATABASE DESIGN**

The database is designed to manage large bodies of information. The management of data involves both the definitions of structures for the storage of information. In addition, the database system must provide for the safety of the information solved, despite system crashes or attempts at unauthorized access. For developing an efficient database users have to fulfill certain conditions such as controlled redundancy.

* Defining the data
* Inputting the data
* Locating the data
* Accessing the data
* Communicating the data

Revising the data

**Objectives of Database Design**

For designing a database design several objectives have to be met as follows:

* Ease of use
* Control of data integrity
* Control of redundancy
* Control of security
* Data independence (logical & physical)
* Data storage protection
* System performance

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information. In any system results of processing are communicated to the users and other systems through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source of information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

Output design generally refers to the results and information that are generated by the system for many end-users; output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application.

In this Online Repository System project output is to view customer details, employee lists, order tracking details, and attendance percentage results.

**SYSTEM FLOW DIAGRAM**

**DATA FLOW DIAGRAM**

Browses bike and scootys, posts queries and requests bookings

User

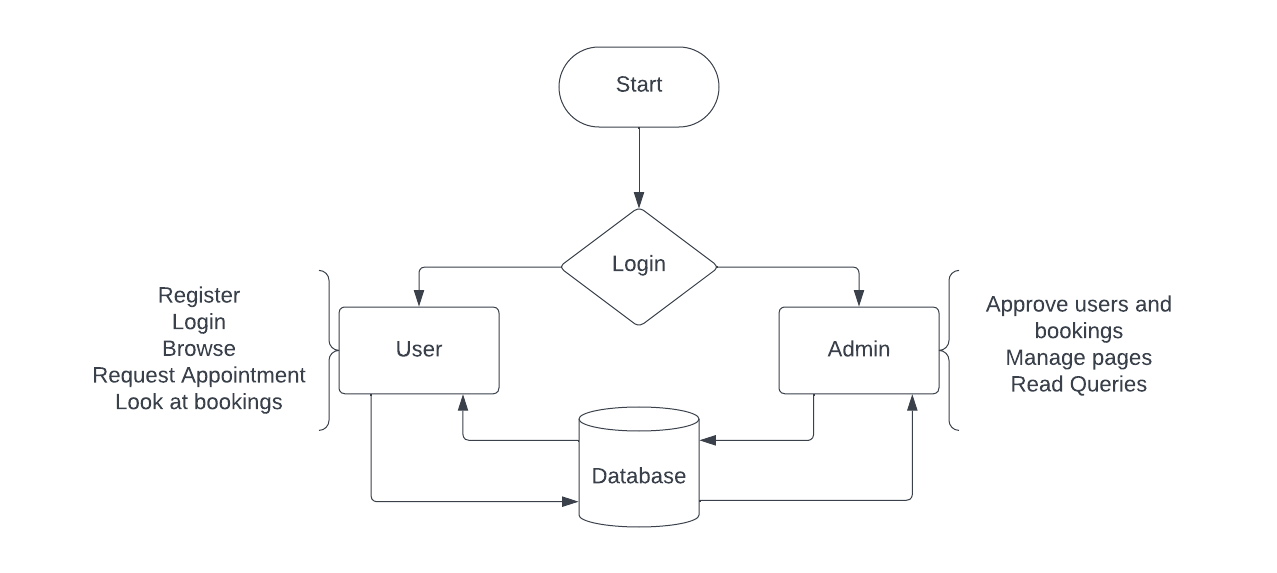
Admin

Approves users and bookings

Checks booking status

Oversees bookings and web pages

**ER DIAGRAM**



**DATABASE DESIGN**

**Table name: admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **LENGTH** | **DESCRIPTION** |
| id | int | 10 | unique id |
| UserName | varchar | 255 | Admin name |
| Password | varchar | 255 | Admin password |
| updationDate | varchar | 255 | Update date |

**Table name: tblusers**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **LENGTH** | **DESCRIPTION** |
| id | int | 10 | unique id |
| **FullName** | varchar | 255 | full name |
| **EmailId Index** | varchar | 255 | email |
| **Password** | varchar | 255 | password |
| **ContactNo** | varchar | 255 | Contact number |
| **dob** | varchar | 255 | Date of birth |
| **Address** | varchar | 255 | Address |
| **City** | varchar | 255 | city |
| **Country** | varchar | 255 | country |
| **RegDate** | varchar | 255 | Registration date |
| **UpdationDa** | varchar | 255 | Update date |

**Table name: tblbooking**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **LENGTH** | **DESCRIPTION** |
| Id | int | 11 | Unique ID |
| **BookingNumber** | varchar | 255 | booking number |
| **userEmail** | varchar | 255 | email |
| **VehicleId** | varchar | 255 | Vehicle id |
| **FromDate** | varchar | 255 | From date |
| **ToDate** | varchar | 255 | To date |
| **message** | varchar | 255 | message |
| **Status** | varchar | 255 | status |

**CONCLUSION**

In conclusion, developing a used bike and scooty selling website using HTML, CSS, Bootstrap, PHP, and MySQL offers a powerful, cost-effective, and customizable solution for both website owners and users. This technology stack combines the strengths of front-end and back-end development, resulting in a user-friendly, responsive, and feature-rich platform that can significantly benefit the automotive industry.

HTML, CSS, and Bootstrap empower the creation of an intuitive and visually appealing user interface, enhancing the overall user experience. Their responsive design ensures accessibility across a variety of devices, from smartphones to desktops, a critical feature in today's mobile-centric world.

PHP and MySQL provide the website with robust functionality and data management capabilities. PHP allows for real-time interactions, secure user authentication, and integration with payment gateways. MySQL, as a reliable database management system, efficiently handles large datasets, ensuring a seamless experience for both sellers and buyers.

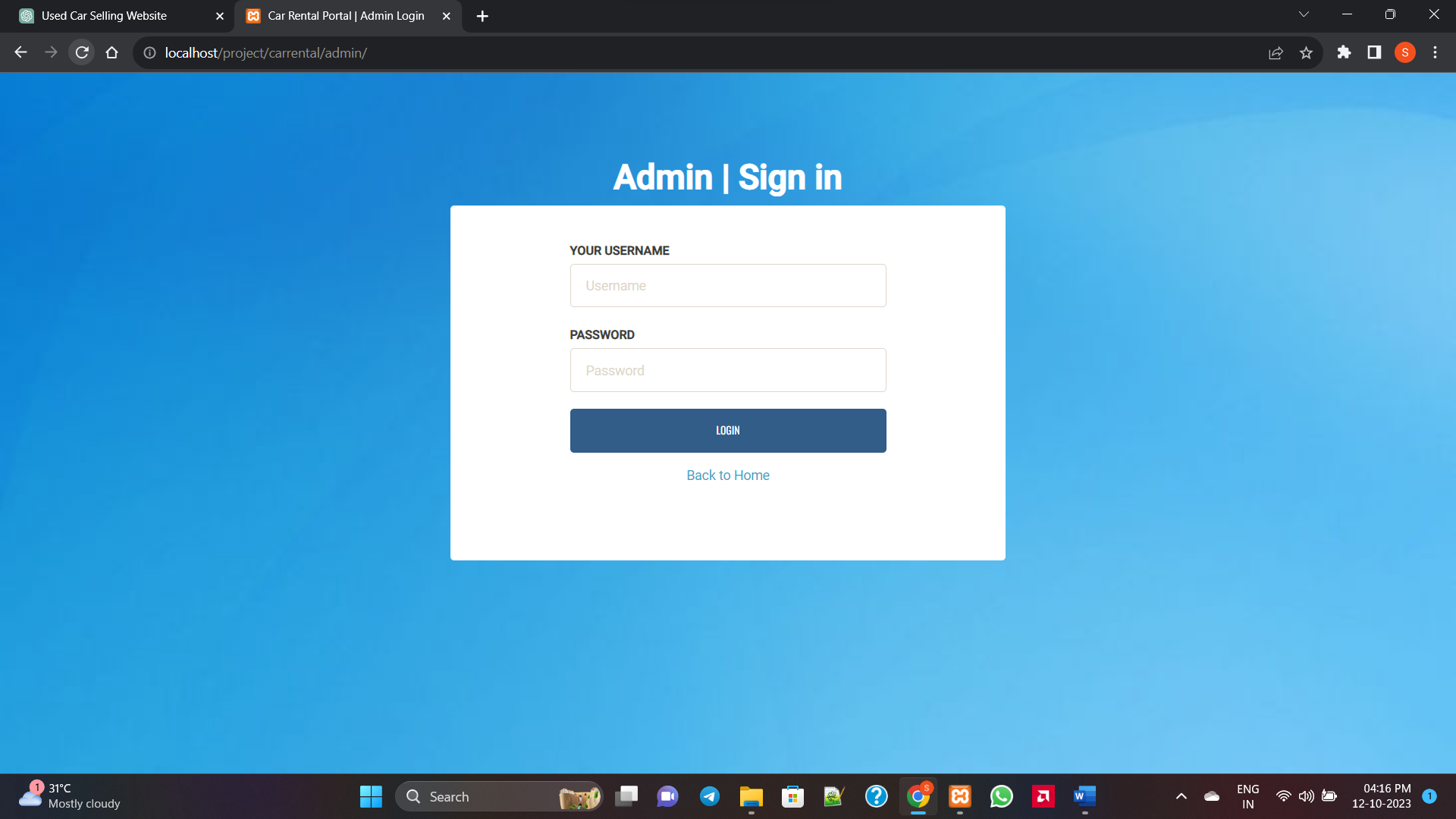
The website's advantages extend to inventory management, security, SEO-friendliness, and scalability. It can accommodate growth and is adaptable to evolving industry needs, making it an ideal choice for businesses and entrepreneurs looking to establish their presence in the used bike and scooty market.

Moreover, the control and ownership of the entire website, along with the ability to implement data privacy compliance features, offer peace of mind to website owners.

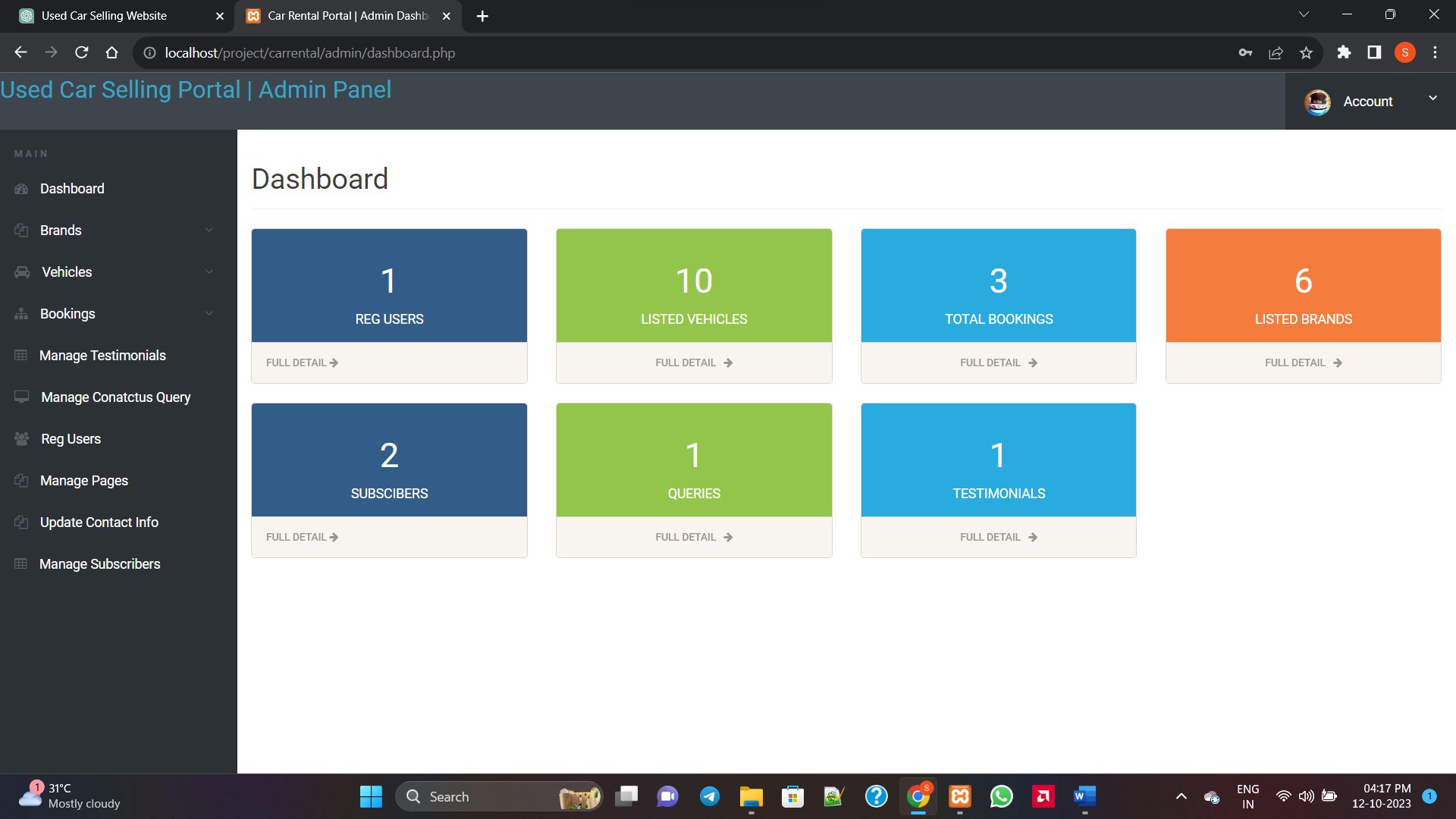
Overall, a used bike and scooty selling website developed with HTML, CSS, Bootstrap, PHP, and MySQL is a practical and sustainable solution. It leverages the strengths of these technologies to provide a platform that is responsive, secure, and cost-effective, catering to the needs of both buyers and sellers in the dynamic world of used bike and scootys.

**SCREENSHOTS**

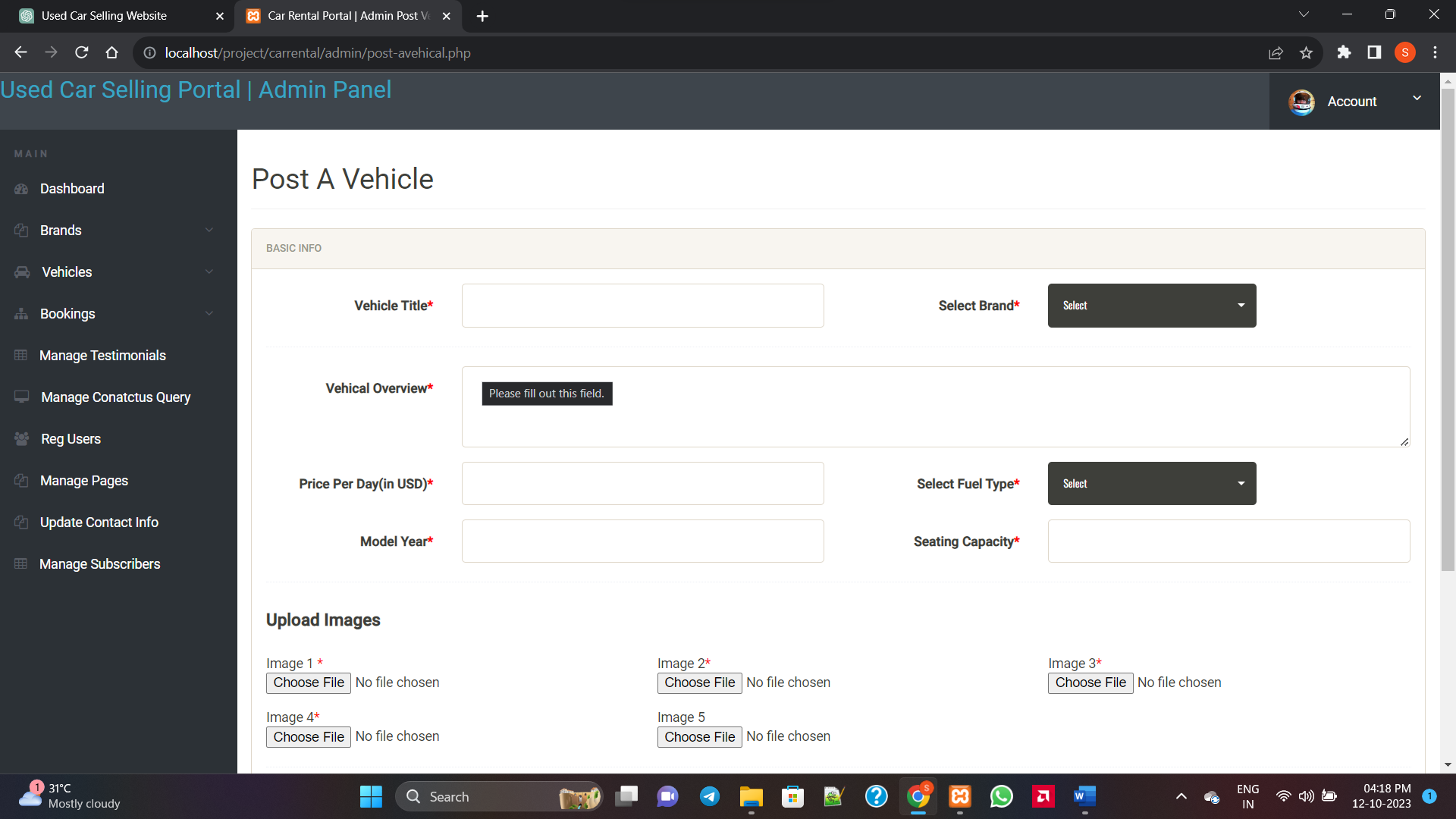
**Admin Login**

****

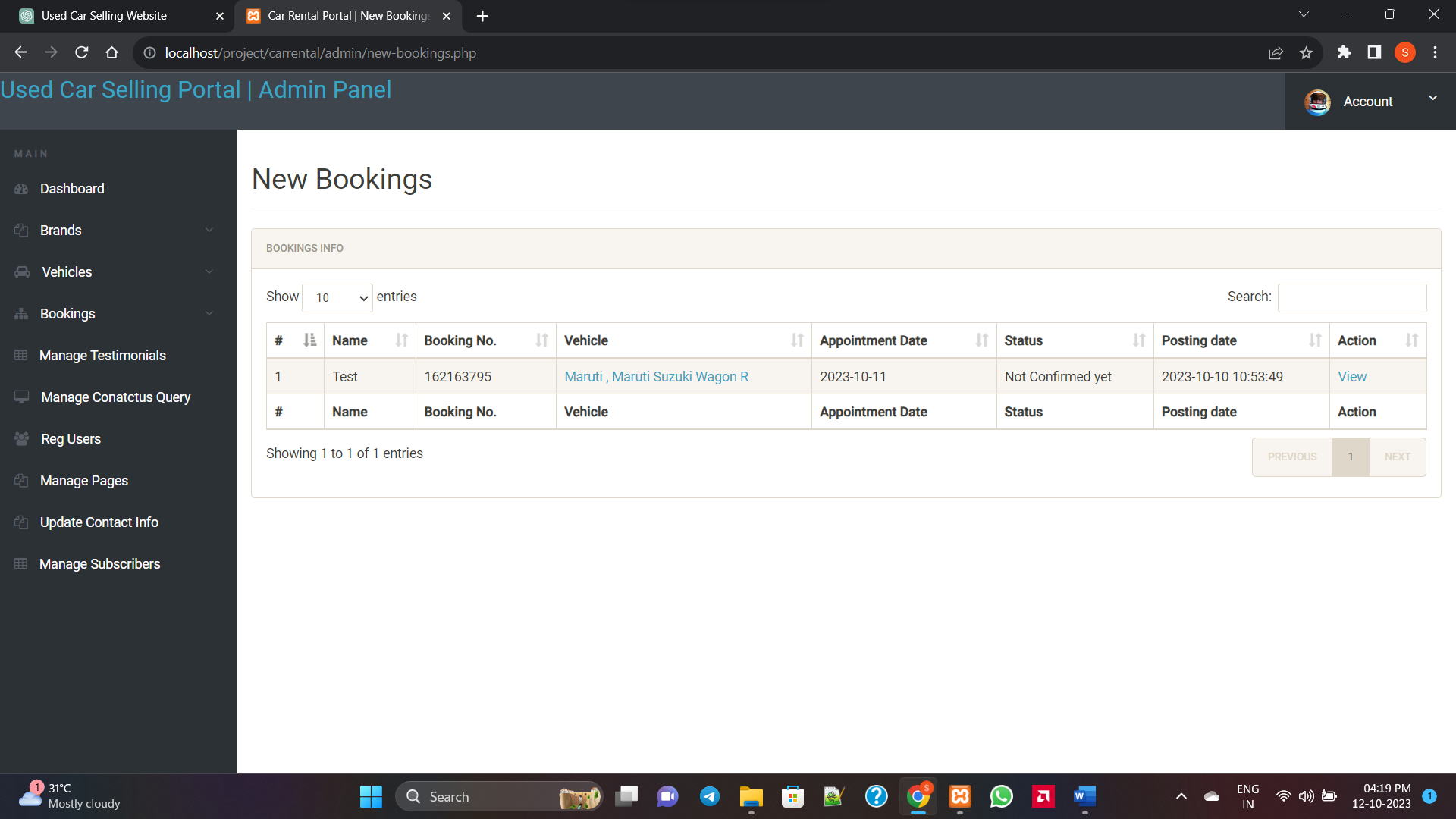
**Admin Dashboard**

****

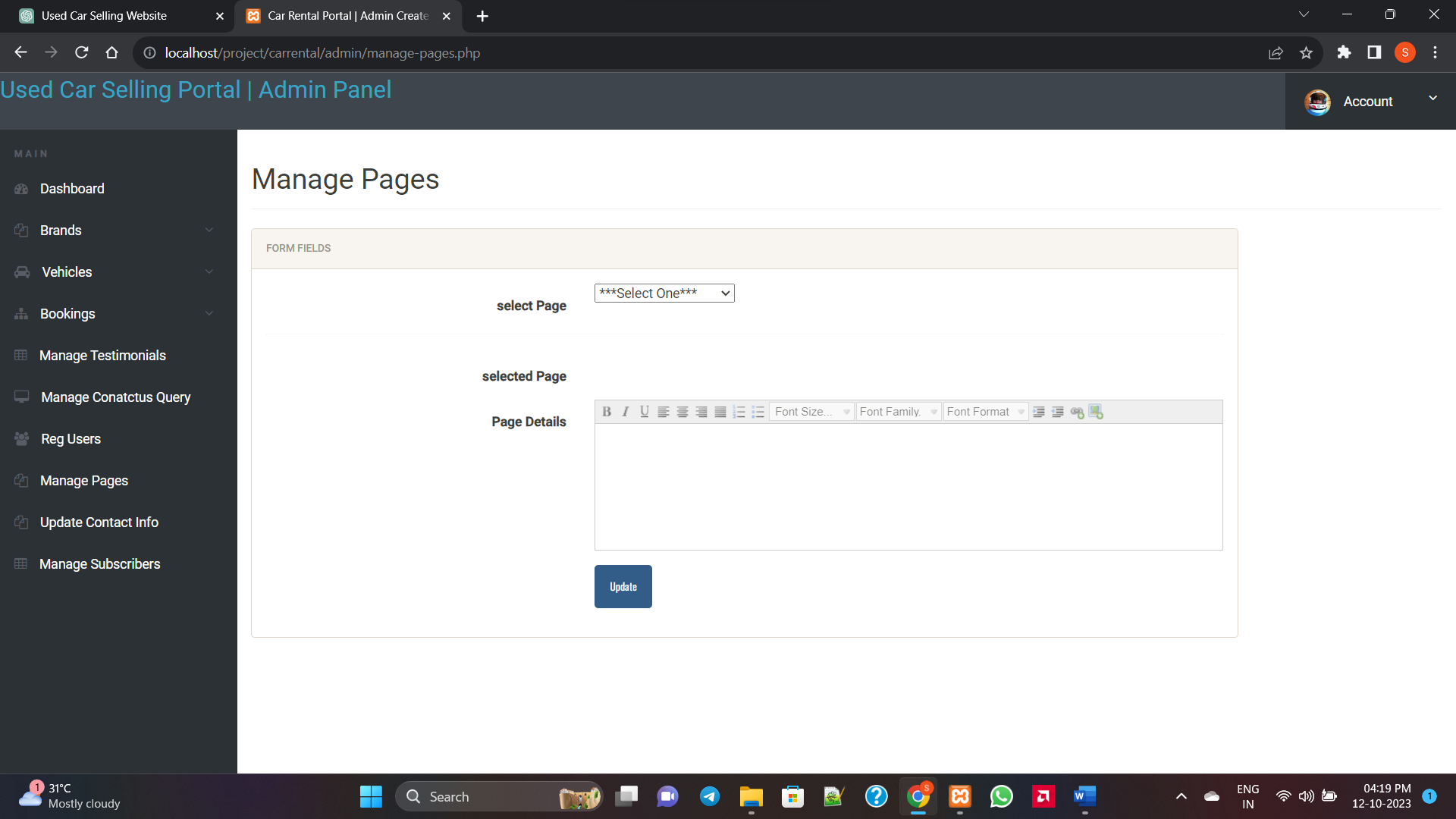
**Add vehicle page**

****

**Manage bookings page**

****

**Manage page**

****

**SAMPLE CODE**

**<?php**

**session\_start();**

**error\_reporting(0);**

**include('includes/config.php');**

**if(strlen($\_SESSION['alogin'])==0)**

**{**

**header('location:index.php');**

**}**

**else{**

**?>**

**<!doctype html>**

**<html lang="en" class="no-js">**

**<head>**

**<meta charset="UTF-8">**

**<meta http-equiv="X-UA-Compatible" content="IE=edge">**

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

**<meta name="description" content="">**

**<meta name="author" content="">**

**<meta name="theme-color" content="#3e454c">**

**<title>Bike and scooty Rental Portal | Canceled Bookings </title>**

**<!-- Font awesome -->**

**<link rel="stylesheet" href="css/font-awesome.min.css">**

**<!-- Sandstone Bootstrap CSS -->**

**<link rel="stylesheet" href="css/bootstrap.min.css">**

**<!-- Bootstrap Datatables -->**

**<link rel="stylesheet" href="css/dataTables.bootstrap.min.css">**

**<!-- Bootstrap social button library -->**

**<link rel="stylesheet" href="css/bootstrap-social.css">**

**<!-- Bootstrap select -->**

**<link rel="stylesheet" href="css/bootstrap-select.css">**

**<!-- Bootstrap file input -->**

**<link rel="stylesheet" href="css/fileinput.min.css">**

**<!-- Awesome Bootstrap checkbox -->**

**<link rel="stylesheet" href="css/awesome-bootstrap-checkbox.css">**

**<!-- Admin Stye -->**

**<link rel="stylesheet" href="css/style.css">**

**<style>**

**.errorWrap {**

**padding: 10px;**

**margin: 0 0 20px 0;**

**background: #fff;**

**border-left: 4px solid #dd3d36;**

**-webkit-box-shadow: 0 1px 1px 0 rgba(0,0,0,.1);**

**box-shadow: 0 1px 1px 0 rgba(0,0,0,.1);**

**}**

**.succWrap{**

**padding: 10px;**

**margin: 0 0 20px 0;**

**background: #fff;**

**border-left: 4px solid #5cb85c;**

**-webkit-box-shadow: 0 1px 1px 0 rgba(0,0,0,.1);**

**box-shadow: 0 1px 1px 0 rgba(0,0,0,.1);**

**}**

**</style>**

**</head>**

**<body>**

**<?php include('includes/header.php');?>**

**<div class="ts-main-content">**

**<?php include('includes/leftbar.php');?>**

**<div class="content-wrapper">**

**<div class="container-fluid">**

**<div class="row">**

**<div class="col-md-12">**

**<h2 class="page-title">Canceled Bookings</h2>**

**<!-- Zero Configuration Table -->**

**<div class="panel panel-default">**

**<div class="panel-heading">Bookings Info</div>**

**<div class="panel-body">**

**<table id="zctb" class="display table table-striped table-bordered table-hover" cellspacing="0" width="100%">**

**<thead>**

**<tr>**

**<th>#</th>**

**<th>Name</th>**

**<th>Booking No.</th>**

**<th>Vehicle</th>**

**<th>From Date</th>**

**<th>To Date</th>**

**<th>Status</th>**

**<th>Posting date</th>**

**<th>Action</th>**

**</tr>**

**</thead>**

**<tfoot>**

**<tr>**

**<th>#</th>**

**<th>Name</th>**

**<th>Booking No.</th>**

**<th>Vehicle</th>**

**<th>From Date</th>**

**<th>To Date</th>**

**<th>Status</th>**

**<th>Posting date</th>**

**<th>Action</th>**

**</tr>**

**</tfoot>**

**<tbody>**

**<?php**

**$status=2;**

**$sql = "SELECT tblusers.FullName,tblbrands.BrandName,tblvehicles.VehiclesTitle,tblbooking.FromDate,tblbooking.ToDate,tblbooking.message,tblbooking.VehicleId as vid,tblbooking.Status,tblbooking.PostingDate,tblbooking.id,tblbooking.BookingNumber from tblbooking join tblvehicles on tblvehicles.id=tblbooking.VehicleId join tblusers on tblusers.EmailId=tblbooking.userEmail join tblbrands on tblvehicles.VehiclesBrand=tblbrands.id where tblbooking.Status=:status";**

**$query = $dbh -> prepare($sql);**

**$query -> bindParam(':status',$status, PDO::PARAM\_STR);**

**$query->execute();**

**$results=$query->fetchAll(PDO::FETCH\_OBJ);**

**$cnt=1;**

**if($query->rowCount() > 0)**

**{**

**foreach($results as $result)**

**{ ?>**

**<tr>**

**<td><?php echo htmlentities($cnt);?></td>**

**<td><?php echo htmlentities($result->FullName);?></td>**

**<td><?php echo htmlentities($result->BookingNumber);?></td>**

**<td><a href="edit-vehicle.php?id=<?php echo htmlentities($result->vid);?>"><?php echo htmlentities($result->BrandName);?> , <?php echo htmlentities($result->VehiclesTitle);?></td>**

**<td><?php echo htmlentities($result->FromDate);?></td>**

**<td><?php echo htmlentities($result->ToDate);?></td>**

**<td><?php**

**if($result->Status==0)**

**{**

**echo htmlentities('Not Confirmed yet');**

**} else if ($result->Status==1) {**

**echo htmlentities('Confirmed');**

**}**

**else{**

**echo htmlentities('Cancelled');**

**}**

**?></td>**

**<td><?php echo htmlentities($result->PostingDate);?></td>**

**<td>**

**<a href="bookig-details.php?bid=<?php echo htmlentities($result->id);?>"> View</a>**

**</td>**

**</tr>**

**<?php $cnt=$cnt+1; }} ?>**

**</tbody>**

**</table>**

**</div>**

**</div>**

**</div>**

**</div>**

**</div>**

**</div>**

**</div>**

**<!-- Loading Scripts -->**

**<script src="js/jquery.min.js"></script>**

**<script src="js/bootstrap-select.min.js"></script>**

**<script src="js/bootstrap.min.js"></script>**

**<script src="js/jquery.dataTables.min.js"></script>**

**<script src="js/dataTables.bootstrap.min.js"></script>**

**<script src="js/Chart.min.js"></script>**

**<script src="js/fileinput.js"></script>**

**<script src="js/chartData.js"></script>**

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

**</body>**

**</html>**

**<?php } ?>**

FUTURE SCOPE

1. **Electric Vehicles (EVs) Integration**: With the growing popularity of electric vehicles, including electric bikes and scooters, there is a significant opportunity to integrate EVs into bike and scooty rental fleets. Investing in electric vehicles can help reduce environmental impact, lower operating costs, and appeal to environmentally conscious consumers.
2. **Dockless Rental Systems**: Future advancements in GPS and IoT technologies may enable the development of dockless bike and scooty rental systems. Users can locate, unlock, and rent vehicles using a mobile app, without the need for designated docking stations. This flexibility enhances convenience and expands the reach of rental services.
3. **Subscription-based Models**: Introducing subscription-based rental models can cater to the needs of frequent users, offering them a cost-effective and hassle-free way to access bikes and scooters for extended periods. Subscription plans could include unlimited rides, discounted rates, and additional perks to incentivize long-term commitments.
4. **Integration with Mobility-as-a-Service (MaaS) Platforms**: Collaborating with MaaS platforms that offer integrated transportation solutions can enhance the accessibility and visibility of bike and scooty rental services. Integration with ride-sharing, public transit, and other mobility services provides users with seamless multimodal transportation options.
5. **Smart Infrastructure and Parking Solutions**: Future developments in smart infrastructure and urban planning may include dedicated bike lanes, parking zones, and charging stations for electric vehicles. By leveraging smart city initiatives, bike and scooty rental services can optimize operations, improve safety, and enhance user experience.
6. **Personalization and Customization**: Implementing personalized rental experiences based on user preferences, such as preferred vehicle models, routes, and amenities, can enhance customer satisfaction and loyalty. Customization options could include adjustable seats, storage compartments, and integrated technology features.
7. **Augmented Reality (AR) and Virtual Reality (VR) Experiences**: Integrating AR and VR technologies into the rental process can provide users with immersive experiences, such as virtual vehicle tours, interactive maps, and safety simulations. These technologies enhance engagement, education, and user confidence in using rental vehicles.
8. **Blockchain-based Solutions for Security and Transparency**: Leveraging blockchain technology can enhance security, transparency, and trust in bike and scooty rental transactions. Blockchain-based solutions can provide immutable records of rental agreements, secure payment processing, and decentralized authentication systems.
9. **Innovative Revenue Streams**: Exploring alternative revenue streams beyond traditional rental fees, such as advertising partnerships, data monetization, and value-added services, can diversify revenue sources and improve profitability for rental operators.
10. **Global Expansion and Market Penetration**: As bike and scooty rental services continue to gain popularity worldwide, there is potential for global expansion into new markets and regions. Strategic partnerships, franchising opportunities, and localization efforts can facilitate market penetration and brand growth on a global scale.

REFFERENCES

BIBLIOGRAPHY

Comprehensive Web Development Textbook References

General Web Development:

• MDN Web Docs: (https://developer.mozilla.org/) - The authoritative source from Mozilla, offering in-depth documentation, tutorials, and references for various web technologies.

• W3Schools: (https://www.w3schools.com/) - A well-established website with interactive tutorials, references, and examples for a wide range of web development topics.

• The Odin Project: (https://theodinproject.com/) - A free, full-stack web development curriculum with a strong focus on practical projects.

• FreeCodeCamp: (https://www.freecodecamp.org/) - Another free coding platform with interactive lessons, projects, and a supportive community for web development learning.

Specific Technologies:

• HTML:

o HTML Dog: (https://www.htmldog.com/) - Offers interactive tutorials, challenges, and references for learning HTML.

• CSS:

o CSS-Tricks: (https://css-tricks.com/) - A popular website with articles, tips, tricks, and best practices for mastering CSS.

• JavaScript:

o JavaScript30: (https://javascript30.com/) - Provides 30 JavaScript coding challenges for practicing and improving your JavaScript skills.

o Eloquent JavaScript Website: (https://eloquentjavascript.net/) - Interactive tutorials and resources aligned with the book "Eloquent JavaScript" by Marijn Haverbeke.

• Bootstrap:

o Get Bootstrap: (https://getbootstrap.com/) - The official Bootstrap website with comprehensive documentation, examples, and tutorials for learning and using the framework.

o Start Bootstrap: (https://startbootstrap.com/) - Offers free Bootstrap templates to use as a starting point for your projects.

• PHP:

o PHP.net: (https://www.php.net/manual/en/index.php) - The official PHP website with comprehensive documentation, tutorials, and a reference manual.

o Laracasts: (https://laracasts.com/) - Features video tutorials and screencasts for learning PHP, Laravel (a popular PHP framework), and other web development topics.

• MySQL:

o MySQL Documentation: (https://dev.mysql.com/doc/) - Official documentation from MySQL, providing detailed information on using the database management system.

o SQLBolt: (https://sqlbolt.com/) - An interactive platform where you can practice writing and running SQL queries, the language used with MySQL.

Community Resources:

• Stack Overflow: (https://stackoverflow.com/) - A question-and-answer website for programmers, where you can search for solutions to your coding problems or ask questions related to web development.

• GitHub: (https://github.com/) - A version control system for code hosting and collaboration. GitHub also offers a wealth of open-source web development projects that you can explore and learn from.

Comprehensive Web Development Online References

General Web Development:

• MDN Web Docs: (https://developer.mozilla.org/) - The authoritative source from Mozilla, offering in-depth documentation, tutorials, and references for various web technologies.

• W3Schools: (https://www.w3schools.com/) - A well-established website with interactive tutorials, references, and examples for a wide range of web development topics.

• The Odin Project: (https://theodinproject.com/) - A free, full-stack web development curriculum with a strong focus on practical projects.

• FreeCodeCamp: (https://www.freecodecamp.org/) - Another free coding platform with interactive lessons, projects, and a supportive community for web development learning.

Specific Technologies:

• HTML:

o HTML Dog: (https://www.htmldog.com/) - Offers interactive tutorials, challenges, and references for learning HTML.

• CSS:

o CSS-Tricks: (https://css-tricks.com/) - A popular website with articles, tips, tricks, and best practices for mastering CSS.

• JavaScript:

o JavaScript30: (https://javascript30.com/) - Provides 30 JavaScript coding challenges for practicing and improving your JavaScript skills.

o Eloquent JavaScript Website: (https://eloquentjavascript.net/) - Interactive tutorials and resources aligned with the book "Eloquent JavaScript" by Marijn Haverbeke.

• Bootstrap:

o Get Bootstrap: (https://getbootstrap.com/) - The official Bootstrap website with comprehensive documentation, examples, and tutorials for learning and using the framework.

o Start Bootstrap: (https://startbootstrap.com/) - Offers free Bootstrap templates to use as a starting point for your projects.

• PHP:

o PHP.net: (https://www.php.net/manual/en/index.php) - The official PHP website with comprehensive documentation, tutorials, and a reference manual.

o Laracasts: (https://laracasts.com/) - Features video tutorials and screencasts for learning PHP, Laravel (a popular PHP framework), and other web development topics.

• MySQL:

o MySQL Documentation: (https://dev.mysql.com/doc/) - Official documentation from MySQL, providing detailed information on using the database management system.

o SQLBolt: (https://sqlbolt.com/) - An interactive platform where you can practice writing and running SQL queries, the language used with MySQL.

Community Resources:

• Stack Overflow: (https://stackoverflow.com/) - A question-and-answer website for programmers, where you can search for solutions to your coding problems or ask questions related to web development.

• GitHub: (https://github.com/) - A version control system for code hosting and collaboration. GitHub also offers a wealth of open-source web development projects that you can explore and learn from.