MINI PROJECT REPORT

**On**

**RENT & RUN (car rental service)**



**Submitted by: -**

**PRIYANSHU TIWARI Roll No: 181500516**

**SHIVENDRA PRATAP SINGH Roll No: 181500683**

**SHASHANK YADAV Roll No: 181500657**

**TANEESHA SHARMA Roll No: 181500748**

**KARTIKEYA PANDEY Roll No: 181500314**

**Supervised by: -**

**Ms Harvinder Kaur**

**Assistant Professor**

Department of Computer Engineering & Applications

**Institute of Engineering & Technology**



**GLA University**

**Mathura- 281406, INDIA**

**2020**

**OBJECTIVES OF THE PROJECT**:

This project is designed to aid the car rental company to enable renting of cars through an

online system. It helps the users to search for available cars view profile and book the cars

for the time period. It has a user-friendly interface which helps the user to check for cars

and rent them for the period specified. They could also make payment online. Th rental cars

shall be categorized into economy, premium etc. Based on the type of car required by the

customer, the user shall be able to make bookings. The use of internet technology has made

it easy for the customers to rent a car any time. This system makes the bookings easy. It

saves time and labour. The tool shall ask the user for information such as the date and time

of journey, type of car etc. Also, it will need an identification number. Using these details,

the tool shall help the customer to book a car for the journey.

**EXISTING SYSTEM**:

Car rental service will help users to book a car for some fee specified. Till now there was no

clear web-based UI to help the users to rent the vehicle. They had to manually rent the

vehicle through their offices. It was a difficult task to manage rental vehicles. Keeping track

of all the rental cars was a problem.

**PROPOSED SYSTEM:**

This tool will enable the user to rent a vehicle. The user shall login to the system and check

for availability of cars. The user specifies a type of car and the journey date and time. The

system shall check for the availability of the car and rent the car to the customer. The

user can make payment online. The tool is designed using Php. All the data regarding the

rental cars are stored in MySQL database. The user has to enter his name, address, phone

details and check for the cars available for rent. The main advantage is that the user shall be able to choose a car depending on his budget.

**FUNCTIONAL SPECIFICATIONS:**

**User Specification:**

* *Admin:* Admin can add a car, manage booking car and rent and also view feedback and  
  enquiry.
* *User:* User can view information of available car, booking car, easily get the car on rent  
  and also give feedback and can enquiry.

**MODULE SPECIFICATIONS:**

**View Available Cars:**It is a system design specially for large, premium and small car rental business. The user can view Available cars and user can book for that car.

**Booking Car:**The user can view Available cars and user can book for that car.

**Easily Get the Car on rent:**The Customer can easily get the car whenever they need to on the rent with use of this system.

**Give Feedback:**The customer will give the feedback to the admin.

**Enquiry:**The inquiry can easily done by user.

**Add Car**: The Admin can add the car so that The user can see the available cars and book the car.

**Manage Rent**: The Admin can manage the rent so that the user can see the rent and book the car.

**View Feedback**: The admin easily view the feedbacks and solve the query.

**HARDWARE REQUIREMENT**

**Client side:**

|  |  |
| --- | --- |
| **RAM** | 512 MB or more |
| **Hard disk** | 10 GB(minimum) |
| **Processor** | 1.0 GHz(minimum) |

**Server Side:**

|  |  |
| --- | --- |
| **RAM** | 1 GB(minimum) |
| **Hard disk** | 20 GB(minimum) |
| **Processor** | 2.0 GHz(minimum) |

**SOFTWARE REQUIREMENT**

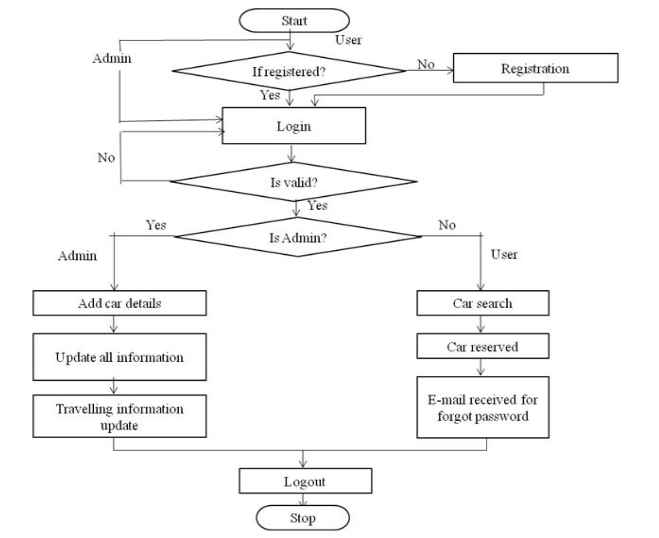
**Client side:**

|  |  |
| --- | --- |
| **Web Browser** | Internet Explorer or any compatible browser |
| **Operating System** | Windows or any equivalent OS |

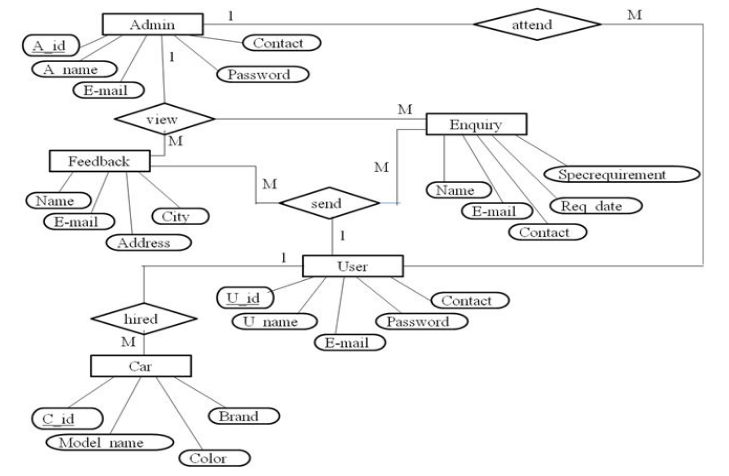
**Server side:**

|  |  |
| --- | --- |
| **Framework** | php |
| **Database Server** | MS SQL Server 2008 |
| **Web Browser** | Internet Explorer 6 or any compatible browser |
| **Operating System** | Windows or Linux |
|  |  |
|  |  |

**SYSTEM FLOWCHART**



**E-R DIAGRAM:**



**DATA FLOW DIAGRAM OF CAR RENTAL MANAGEMENT SYSTEM**

A data flow diagram is graphical tool used to describe and analyse movement of data through a system . These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be

described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labelled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD’S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is

the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

**DFD SYMBOLS:**

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

Process that transforms data flow.

Source or Destination of data

Data flow

Data Store

**CONSTRUCTING A DFD:**

Several rules of thumb are used in drawing DFD’S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

**SAILENT FEATURES OF DFD’S**

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

**TYPES OF DATA FLOW DIAGRAMS:**

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

**CURRENT PHYSICAL:**

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

**CURRENT LOGICAL:**

The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

**NEW LOGICAL**:

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

**NEW PHYSICAL:**

The new physical represents only the physical implementation of the new system.

**RULES GOVERNING THE DFD’S PROCESS**

1. No process can have only outputs.
2. No process can have only inputs. If an object has only inputs than it must be a sink.
3. A process has a verb phrase label.

**DATA STORE**

1. Data cannot move directly from one data store to another data store, a process must move data.
2. Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3. A data store has a noun phrase label.

**SOURCE OR SINK**

The origin and /or destination of data.

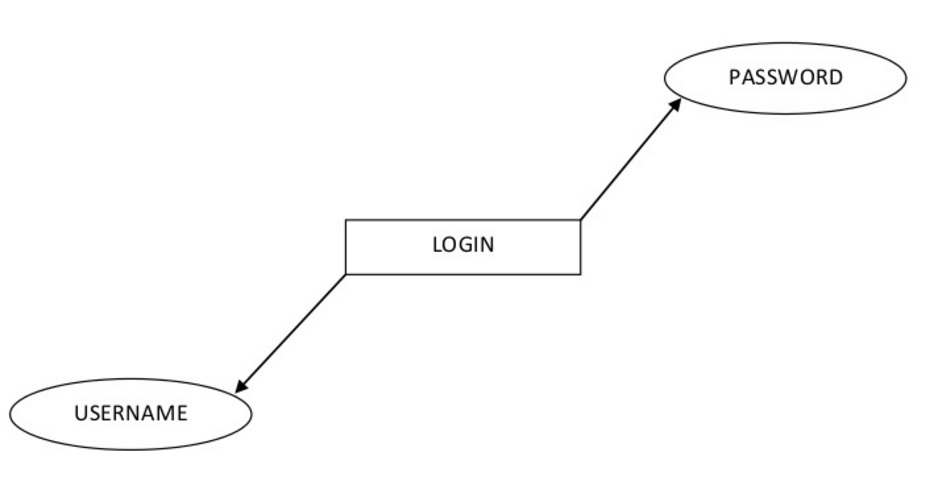
1. Data cannot move direly from a source to sink it must be moved by a process
2. A source and /or sink has a noun phrase land

**DATA FLOW**

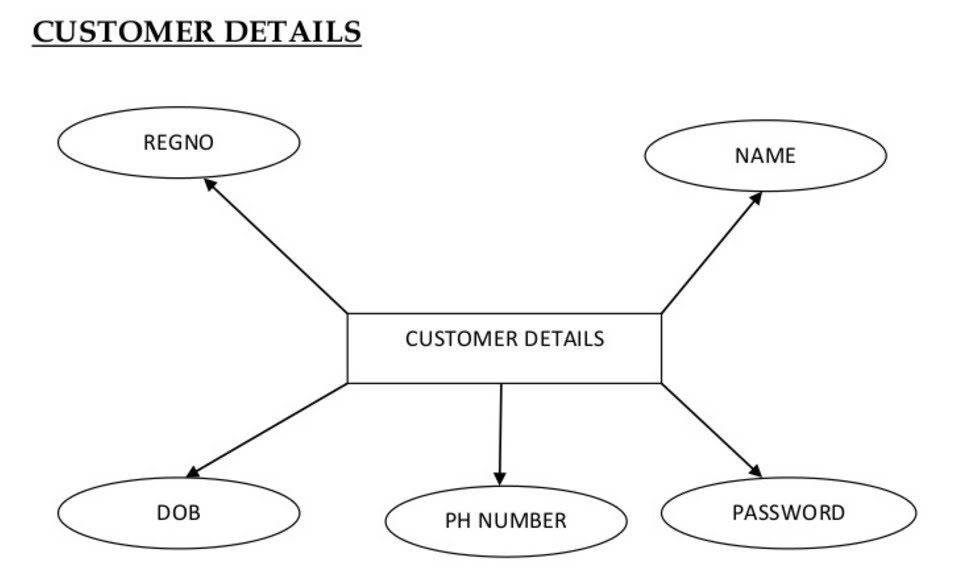
1. A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated however by two separate arrows since these happen at different type.
2. A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3. A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4. A Data flow to a data store means update (delete or change).
5. A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

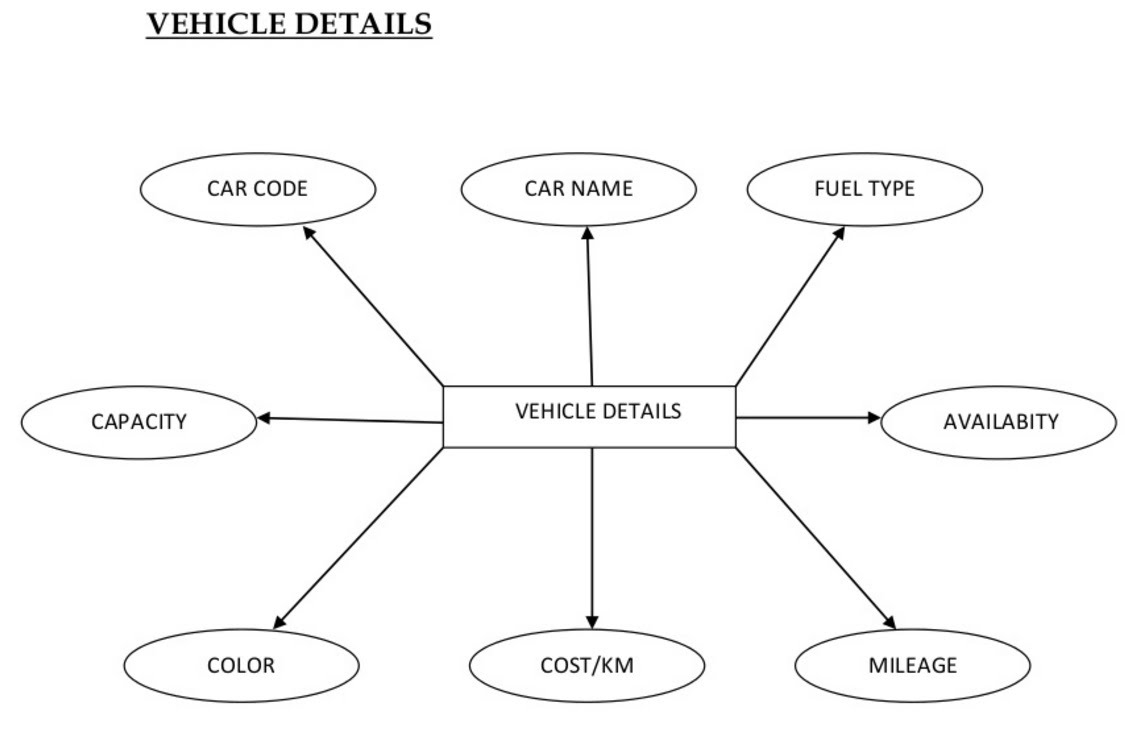
**LOGIN DFD -**



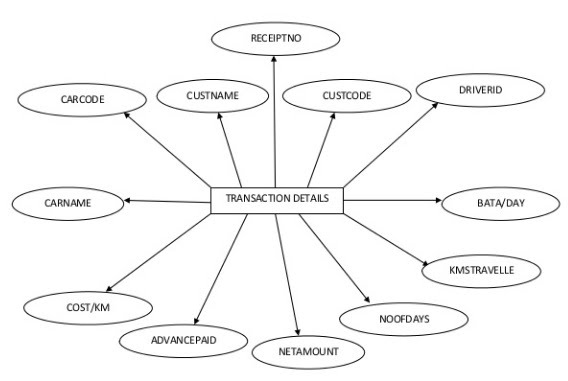
**CUSTOMER DETAILS DFD-**



**VEHICLE DETAILS:**



**TRANSACTION DETAILS:**



**DATA DICTIONARY**

**Tbl\_AdminLogin**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Columnname** | **Datatype** | **Constraint** | **reference** |
| **1** | **AdminID** | **varchar(50)** | **Primary key** |  |
| **2** | **UserName** | **varchar(50)** | **Not null** |  |
| **3** | **Password** | **varchar(50)** | **Notnull** |  |
| **4** | **EmailID** | **varchar(50)** | **Allow null** |  |
| **5** | **Department** | **varchar(50)** | **Not null** |  |

**Tbl\_userDetails**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Columnname** | **Datatype** | **Constraint** | **reference** |
| **1** | **usID** | **int** | **Primary key** |  |
| **2** | **userID** | **varchar(50)** | **Allow null** |  |
| **3** | **Name** | **varchar(50)** | **Allow null** |  |
| **4** | **Address** | **varchar(50)** | **Allow null** |  |
| **5** | **PhoneNo** | **varchar(50)** | **Allow null** |  |
| **6** | **DOB** | **Datetime** | **Allow null** |  |
| **7** | **Experience** | **varchar(50)** | **Allow null** |  |
| **8** | **LicenceNo** | **varchar(50)** | **Allow null** |  |
| **9** | **ImagePath** | **varchar(50)** | **Allow null** |  |
| **10** | **NoOfAccident** | **varchar(50)** | **Allow null** |  |

**Tbl\_FeedBackFrom**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Columnname** | **Datatype** | **Constraint** | **reference** |
| **1** | **FBID** | **int** | **PK** |  |
| **2** | **FeedBackID** | **varchar(50)** | **Not NUll** |  |
| **4** | **VehicleID** | **varchar(50)** | **FK** | **TBL Vehicle** |
| **5** | **usID** | **varchar(50)** | **FK** | **Tbl\_UserDetails** |
| **6** | **Remarks** | **varchar(50)** | **Not NUll** |  |

**Tbl\_VehicleDetails**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Columnname** | **Datatype** | **Constraint** | **Reference** |
| **1** | **VHID** | **int** | **PK** |  |
| **2** | **VehicleID** | **varchar(50)** | **Not Null** |  |
| **3** | **Name** | **varchar(50)** | **Not Null** |  |
| **4** | **VenderID** | **varchar(50)** | **Not Null** |  |
| **5** | **UserID** | **varchar(50)** | **Not Null** |  |
| **6** | **VehicleType** | **varchar(50)** | **Not Null** |  |
| **7** | **RegistorNo** | **varchar(50)** | **Not Null** |  |
| **8** | **RateDay** | **varchar(50)** | **Not Null** |  |
| **9** | **Capacity** | **varchar(50)** | **Not Null** |  |
| **10** | **Routed** | **int** | **Not Null** |  |

**Tbl\_VenderDetails**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Columnname** | **Datatype** | **Constraint** | **reference** |
| **1** | **VID** | **Int** | **PK** |  |
| **2** | **VenderID** | **varchar(50)** | **NotNull** |  |
| **3** | **VenderName** | **varchar(50)** | **NotNull** |  |
| **4** | **Address** | **varchar(50)** | **NotNull** |  |
| **5** | **PhoneNo** | **varchar(50)** | **NotNull** |  |

**Description of Technology Used in Project:**

1. **Php**

PHP started out 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 Pre-processor".
* PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
* It is integrated with a number of 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 C-Like.

**Common uses of PHP:**

* PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
* PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
* You add, delete, modify elements within your database through PHP.
* Access cookies variables and set cookies.
* Using PHP, you can restrict users to access some pages of your website.
* It can encrypt data.

**Characteristics of PHP:**

Five important characteristics make PHP's practical nature possible −

* Simplicity
* Efficiency
* Security
* Flexibility
* Familiarity

**What can PHP do?**

Anything. PHP is mainly focused on server-side scripting, so you can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies. But PHP can do much more.

There are three main areas where PHP scripts are used.

Server-side scripting. This is the most traditional and main target field for PHP. You need three things to make this work: the PHP parser (CGI or server module), a web server and a web browser. You need to run the web server, with a connected PHP installation. You can access the PHP program output with a web browser, viewing the PHP page through the server. All these can run on your home machine if you are just experimenting with PHP programming.

Command line scripting. You can make a PHP script to run it without any server or browser. You only need the PHP parser to use it this way. This type of usage is ideal for scripts regularly executed using cron (on \*nix or Linux) or Task Scheduler (on Windows). These scripts can also be used for simple text processing tasks.

Writing desktop applications. PHP is probably not the very best language to create a desktop application with a graphical user interface, but if you know PHP very well, and would like to use some advanced PHP features in your client-side applications you can also use PHP-GTK to write such programs. You also have the ability to write cross-platform applications this way. PHP-GTK is an extension to PHP, not available in the main distribution.

PHP can be used on all major operating systems, including Linux, many Unix variants (including HP-UX, Solaris and OpenBSD), Microsoft Windows, macOS, RISC OS, and probably others. PHP also has support for most of the web servers today. This includes Apache, IIS, and many others. And this includes any web server that can utilize the Fast CGI PHP binary, like lighttpd and nginx. PHP works as either a module, or as a CGI processor.

So, with PHP, you have the freedom of choosing an operating system and a web server. Furthermore, you also have the choice of using procedural programming or object-oriented programming (OOP), or a mixture of them both.

With PHP you are not limited to output HTML. PHP's abilities include outputting images, PDF files and even Flash movies (using libswf and Ming) generated on the fly. You can also output easily any text, such as XHTML and any other XML file. PHP can autogenerate these files, and save them in the file system, instead of printing it out, forming a server-side cache for your dynamic content.

One of the strongest and most significant features in PHP is its support for a wide range of databases. Writing a database-enabled web page is incredibly simple using one of the database specific extensions (e.g., for mysql), or using an abstraction layer like PDO, or connect to any database supporting the Open Database Connection standard via the ODBC extension. Other databases may utilize cURL or sockets, like CouchDB.

PHP also has support for talking to other services using protocols such as LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (on Windows) and countless others. You can also open raw network sockets and interact using any other protocol. PHP has support for the WDDX complex data exchange between virtually all Web programming languages. Talking about interconnection, PHP has support for instantiation of Java objects and using them transparently as PHP objects.

PHP has useful text processing features, which includes the Perl compatible regular expressions (PCRE), and many extensions and tools to parse and access XML documents. PHP standardizes all of the XML extensions on the solid base of libxml2, and extends the feature set adding SimpleXML, XMLReader and XMLWriter support.

1. **SQL SERVER**

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields.

**SQL SERVER TABLES**

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

**PRIMARY KEY**

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

**RELATIONAL DATABASE**

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

**FOREIGN KEY**

When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

**REFERENTIAL INTEGRITY**

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

**DATA ABSTRACTION**

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

**Physical level:**

This is the lowest level of abstraction at which one describes how the data are actually stored.

**Conceptual Level**:

At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.

**View level**:

This is the highest level of abstraction at which one describes only part of the database.

**ADVANTAGES OF RDBMS**

1. Redundancy can be avoided
2. Inconsistency can be eliminated
3. Data can be Shared
4. Standards can be enforced
5. Security restrictions can be applied
6. Integrity can be maintained
7. Conflicting requirements can be balanced
8. Data independence can be achieved.

**DISADVANTAGES OF DBMS**

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage.

**FEATURES OF SQL SERVER (RDBMS**)

SQL SERVER is one of the leading database management systems (DBMS) because it is the only Database that meets the uncompromising requirements of today’s most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous DSS and OLTP access to the same critical data, SQL Server leads the industry in both performance and capability

**ENTERPRISE WIDE DATA SHARING**

The unrivaled portability and connectivity of the SQL SERVER DBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

**PORTABILITY**

SQL SERVER is fully portable to more than 80 distinct hardware and operating systems platforms, including UNIX, MSDOS, OS/2, Macintosh and dozens of proprietary platforms. This portability gives complete freedom to choose the database sever platform that meets the system requirements.

**OPEN SYSTEMS**

SQL SERVER offers a leading implementation of industry –standard SQL. SQL Server’s open architecture integrates SQL SERVER and non –SQL SERVER DBMS with industries most comprehensive collection of tools, application, and third party software products SQL Server’s Open architecture provides transparent access to data from other relational database and even non-relational database.

**DISTRIBUTED DATA SHARING**

SQL Server’s networking and distributed database capabilities to access data stored on remote server with the same ease as if the information was stored on a single local computer. A single SQL statement can access data at multiple sites. You can store data where system requirements such as performance, security or availability dictate.

**UNMATCHED PERFORMANCE**

The most advanced architecture in the industry allows the SQL SERVER DBMS to deliver unmatched performance.

**SOPHISTICATED CONCURRENCY CONTROL**

Real World applications demand access to critical data. With most database Systems application becomes “contention bound” – which performance is limited not by the CPU power or by disk I/O, but user waiting on one another for data access . SQL Server employs full, unrestricted row-level locking and contention free queries to minimize and in many cases entirely eliminates contention wait times.

**NO I/O BOTTLENECKS**

SQL Server’s fast commit groups commit and deferred write technologies dramatically reduce disk I/O bottlenecks. While some database write whole data block to disk at commit time, SQL Server commits transactions with at most sequential log file on disk at commit time, On high throughput systems, one sequential writes typically group commit multiple transactions. Data read by the transaction remains as shared memory so that other transactions may access that data without reading it again from disk. Since fast commits write all data necessary to the recovery to the log file, modified blocks are written back to the database independently of the transaction commit, when written from memory to disk.

**Software testing and implementation:**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

**SOFTWARE TESTING**

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

UNIT TESTING

MODULE TESTING

SUB-SYSTEM TESING

SYSTEM TESTING

ACCEPTANCE TESTING

Component Testing

Integration Testing

User Testing

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have is white box oriented and some modules the steps are conducted in parallel.

* 1. **WHITE BOX TESTING**

This type of testing ensures that

* All independent paths have been exercised at least once
* All logical decisions have been exercised on their true and false sides
* All loops are executed at their boundaries and within their operational bounds
* All internal data structures have been exercised to assure their validity.
  1. **BASIC PATH TESTING**

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

V(G)=E-N+2 or

V(G)=P+1 or

V(G)=Number Of Regions

Where V(G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

* 1. **CONDITIONAL TESTING**

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

* 1. **DATA FLOW TESTING**

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The definition-use chain method was used in this type of testing. These were particularly useful in nested statements.

* 1. **LOOP TESTING**

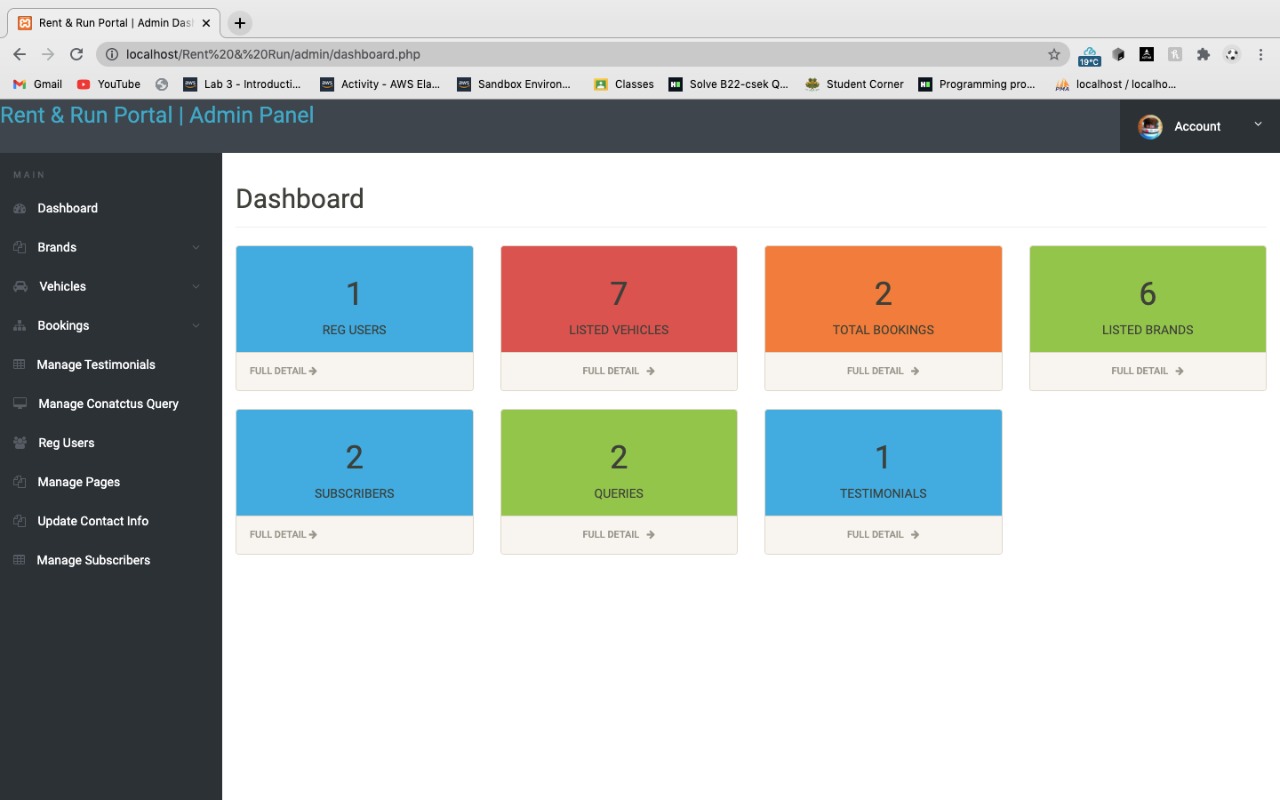
In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

* All the loops were tested at their limits, just above them and just below them.
* All the loops were skipped at least once.
* For nested loops test the inner most loop first and then work outwards.
* For concatenated loops the values of dependent loops were set with the help of connected loop.
* Unstructured loops were resolved into nested loops or concatenated loops and tested as above.

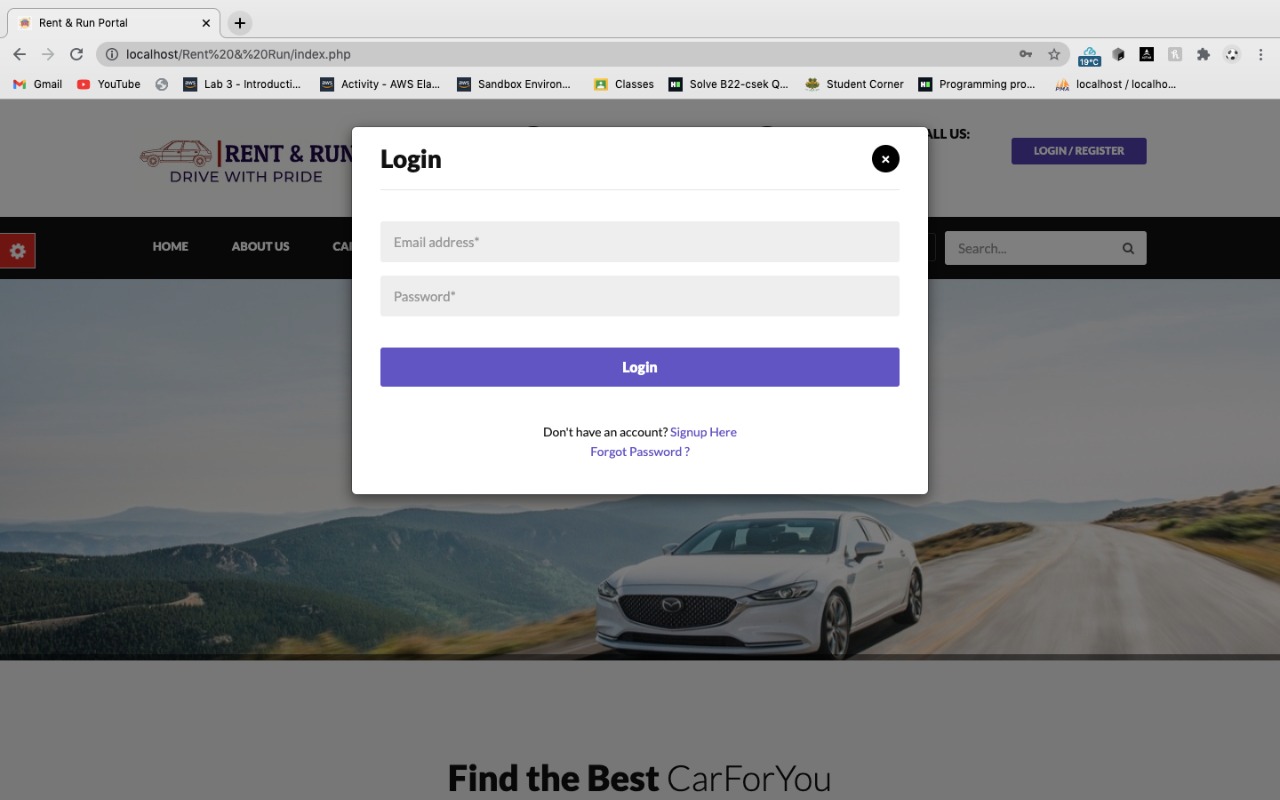
Each unit has been separately tested by the development team itself and all the input have been validated.

**Output Screens:**

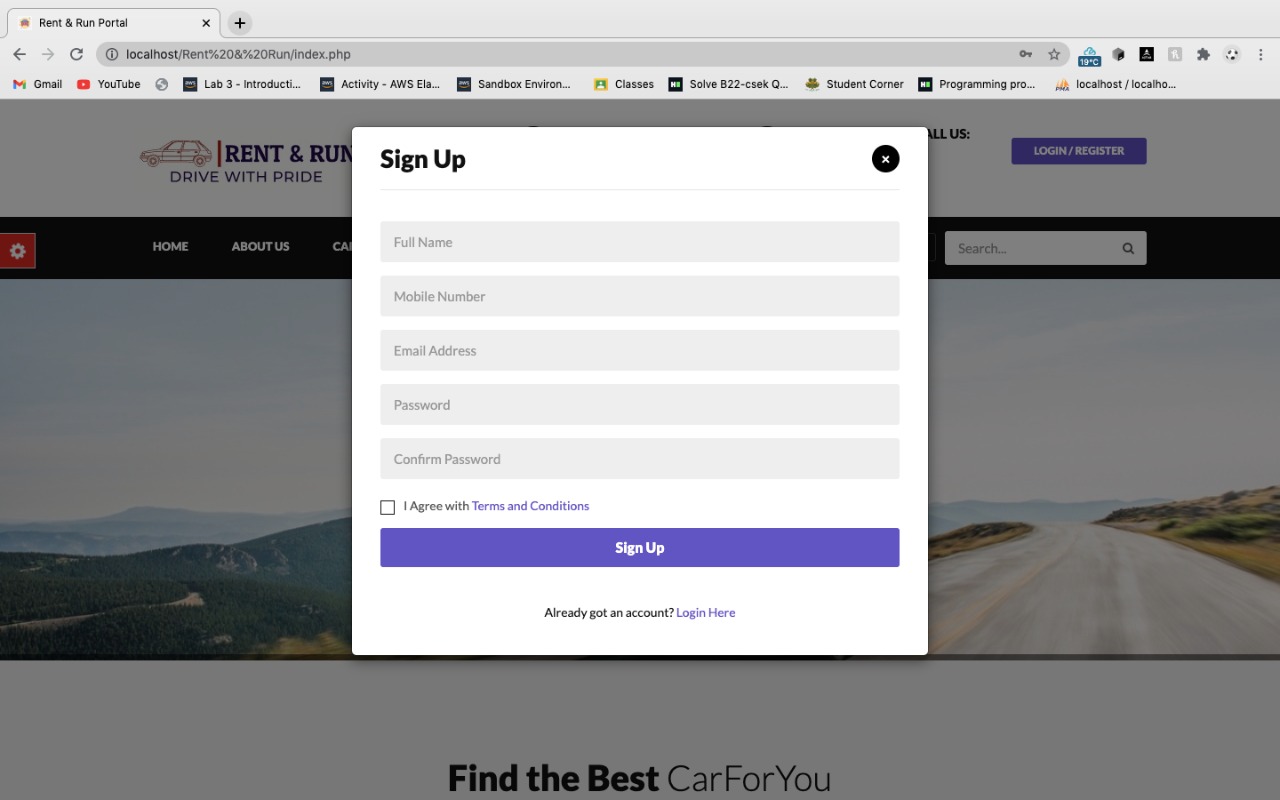
**Dashboard**

****

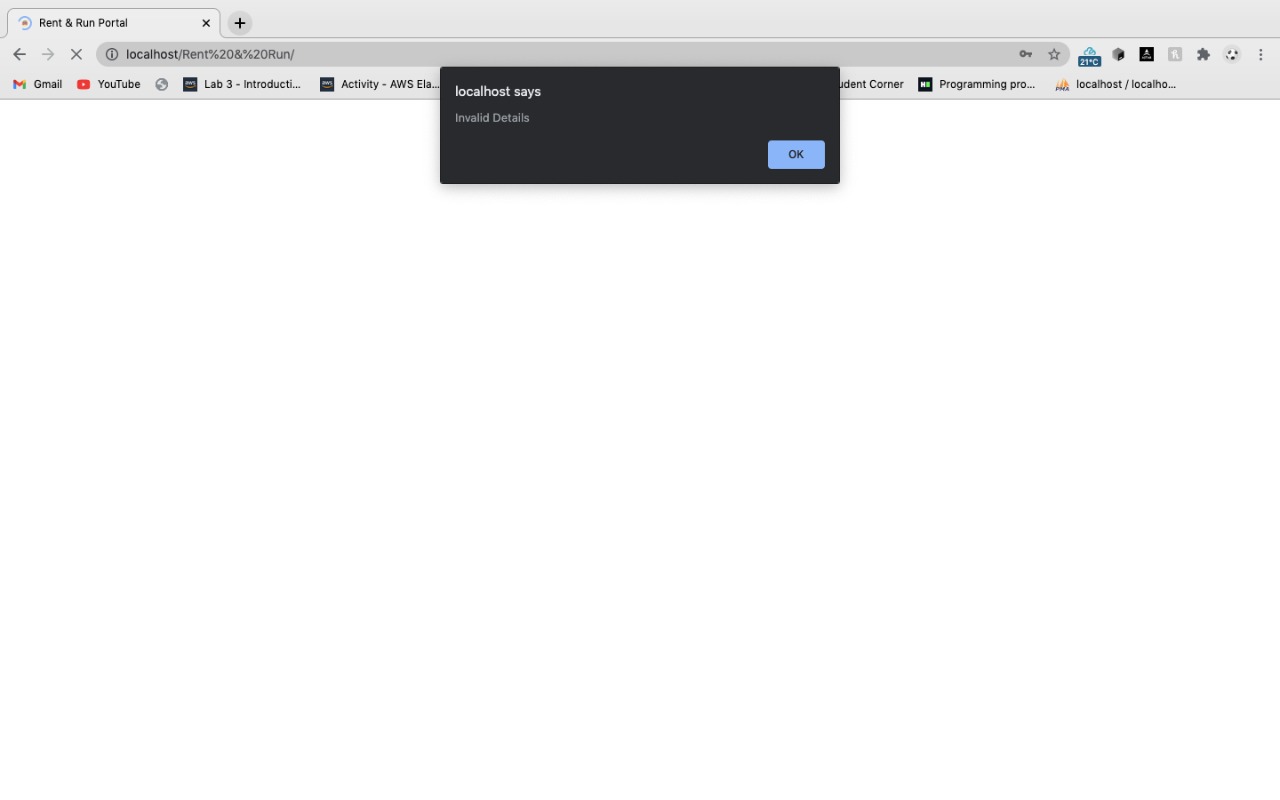
**Login page:**

****

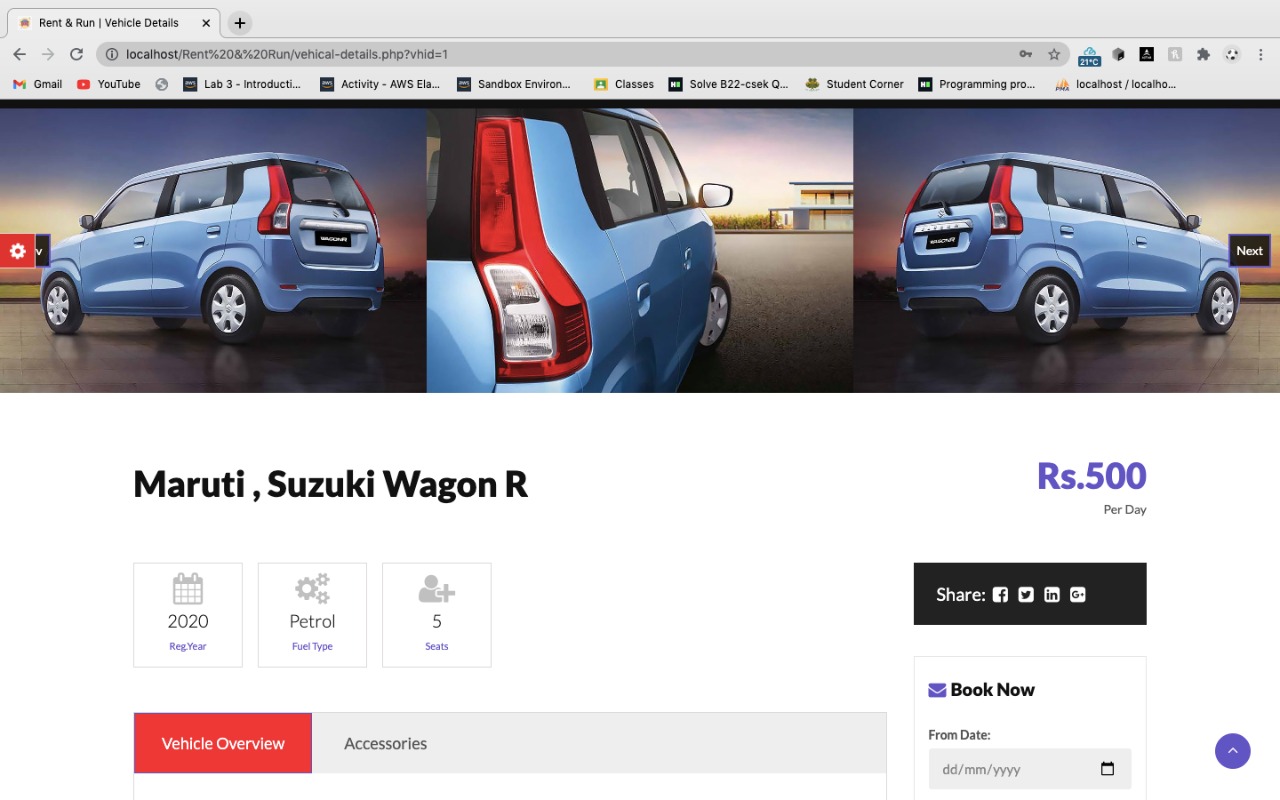
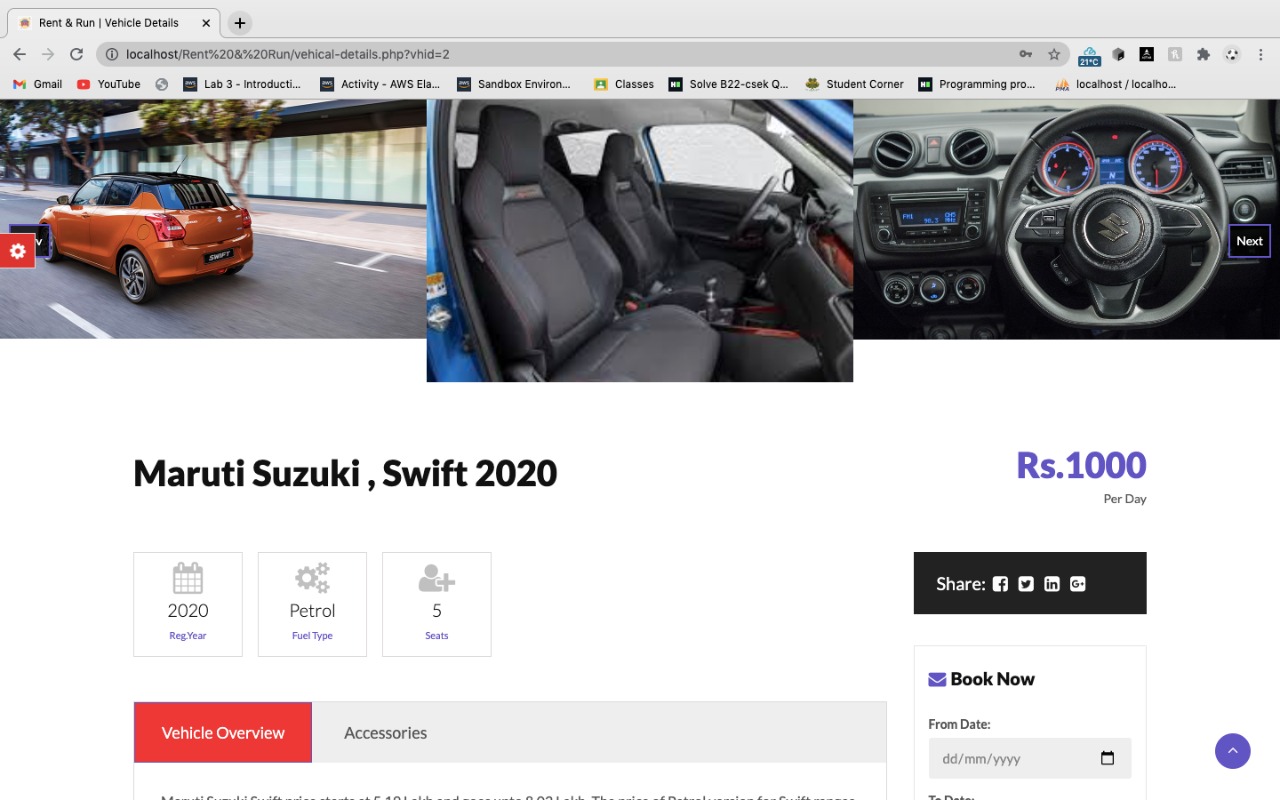
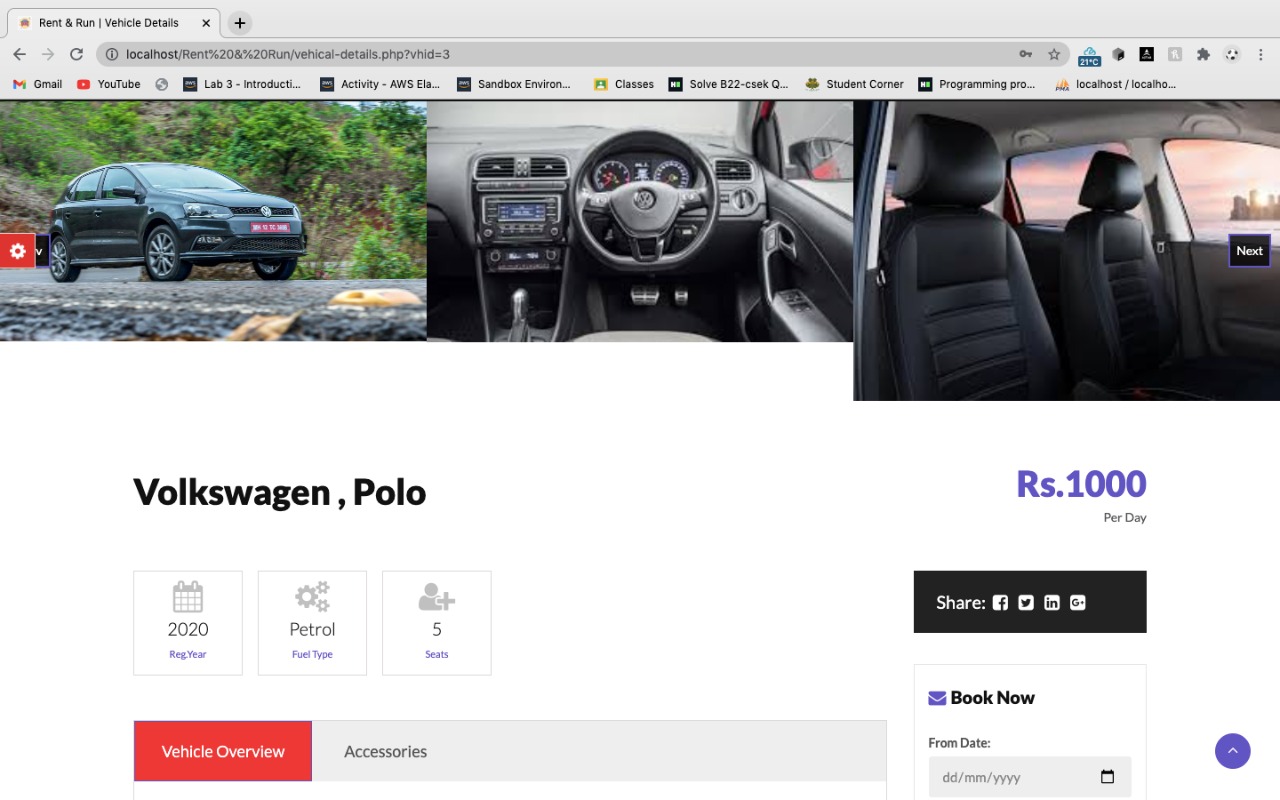
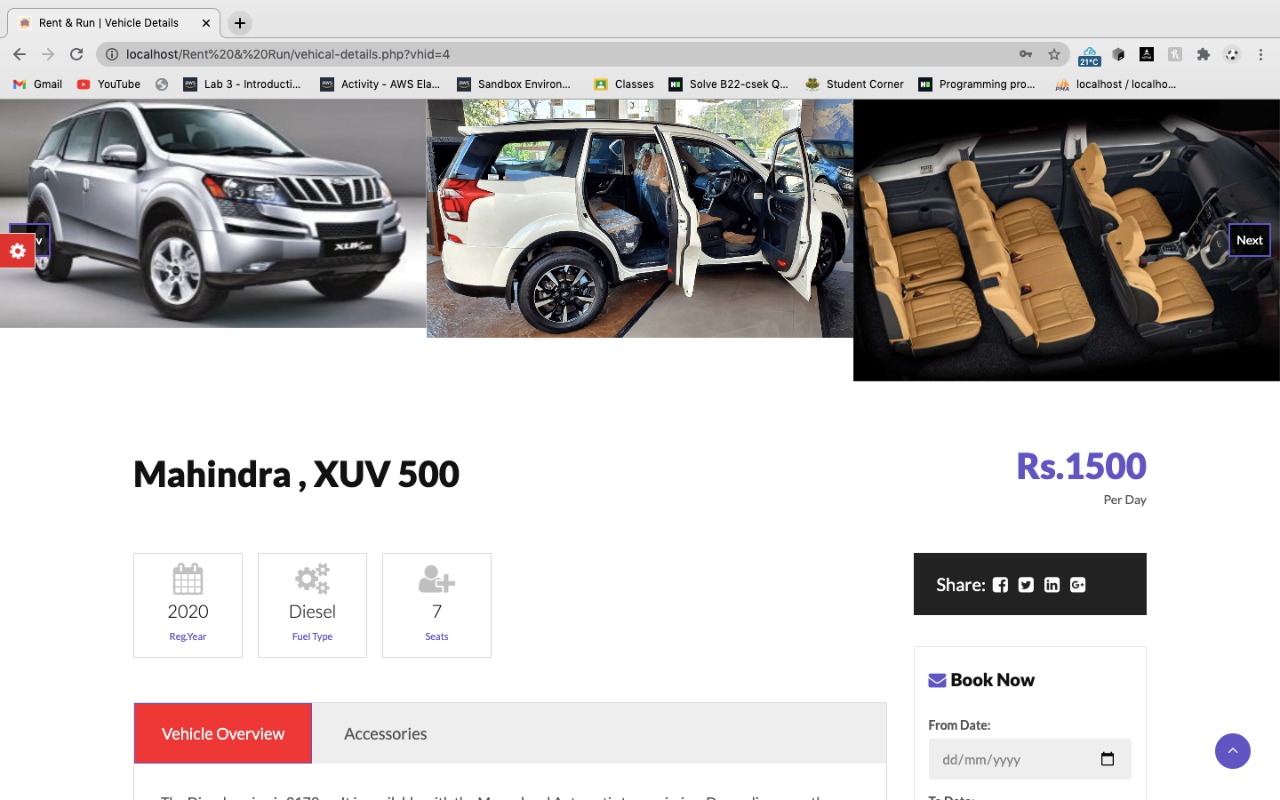
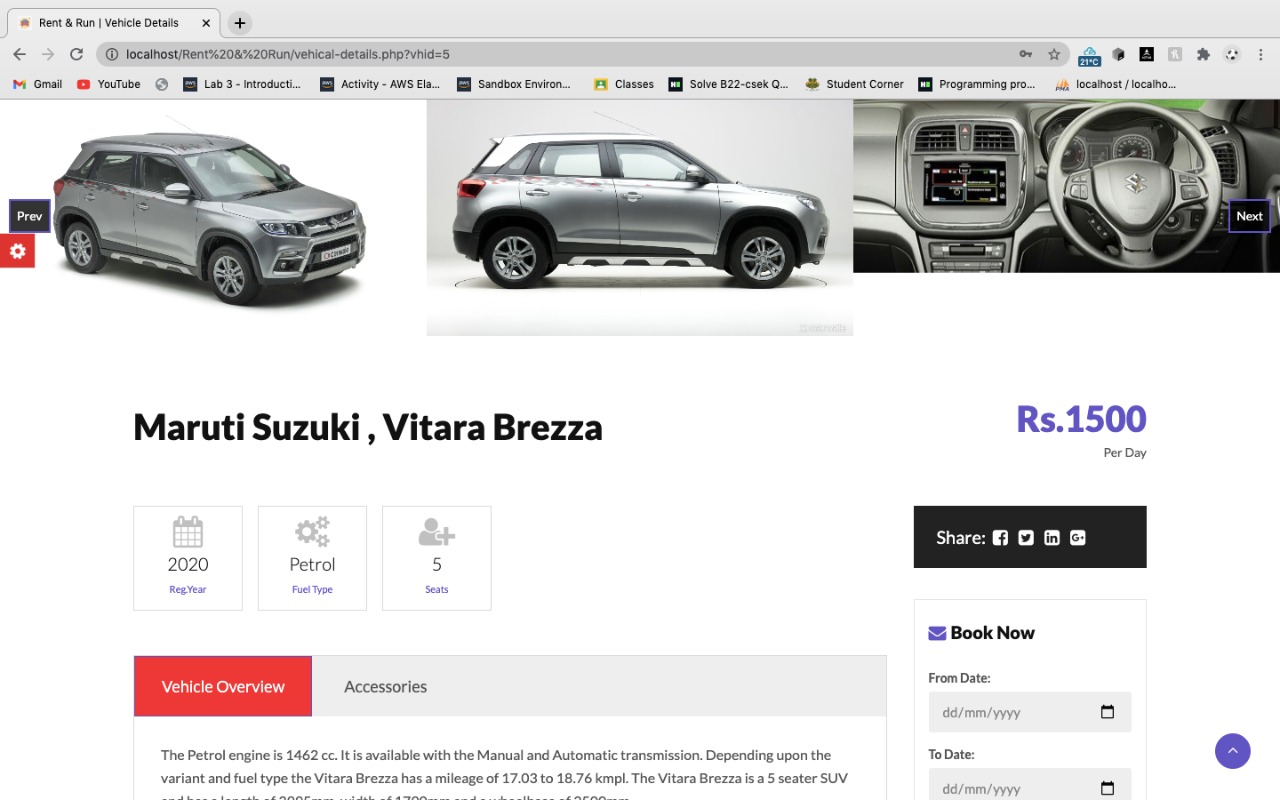
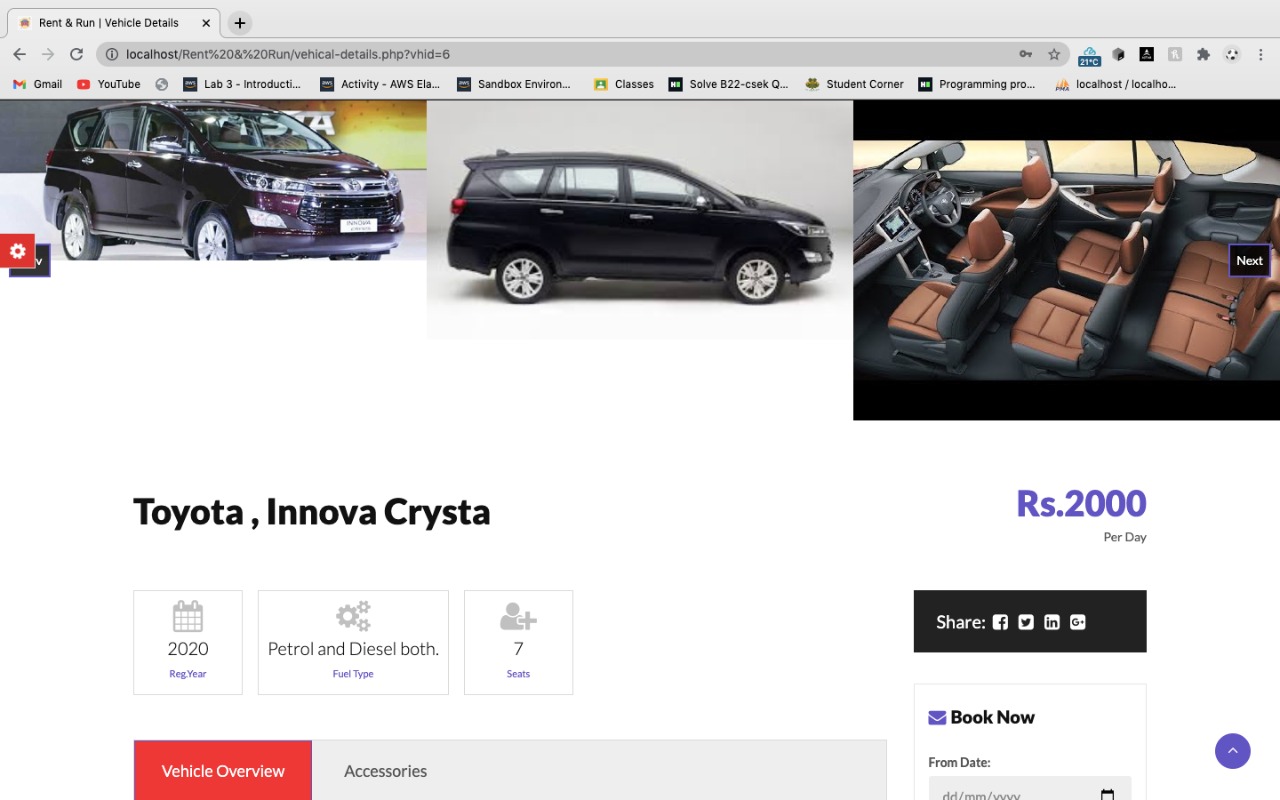
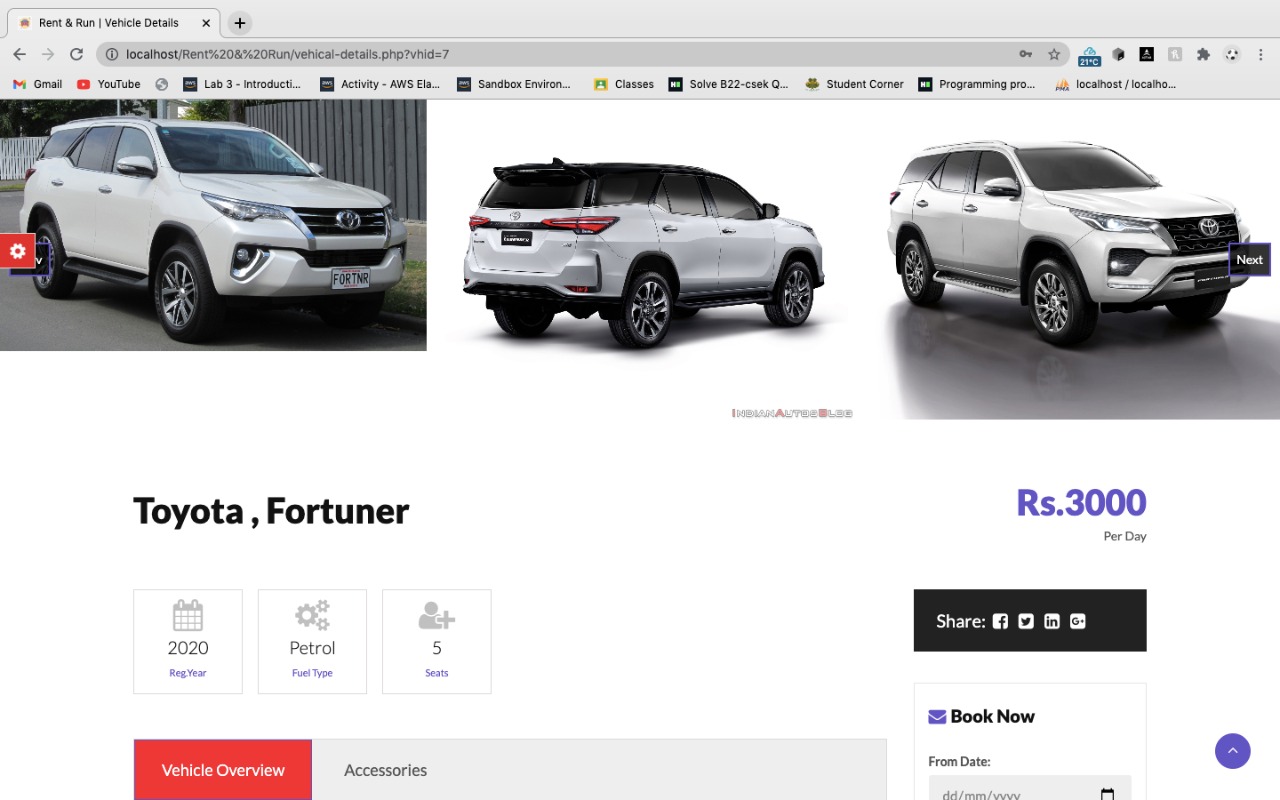
**Registration page:**

****

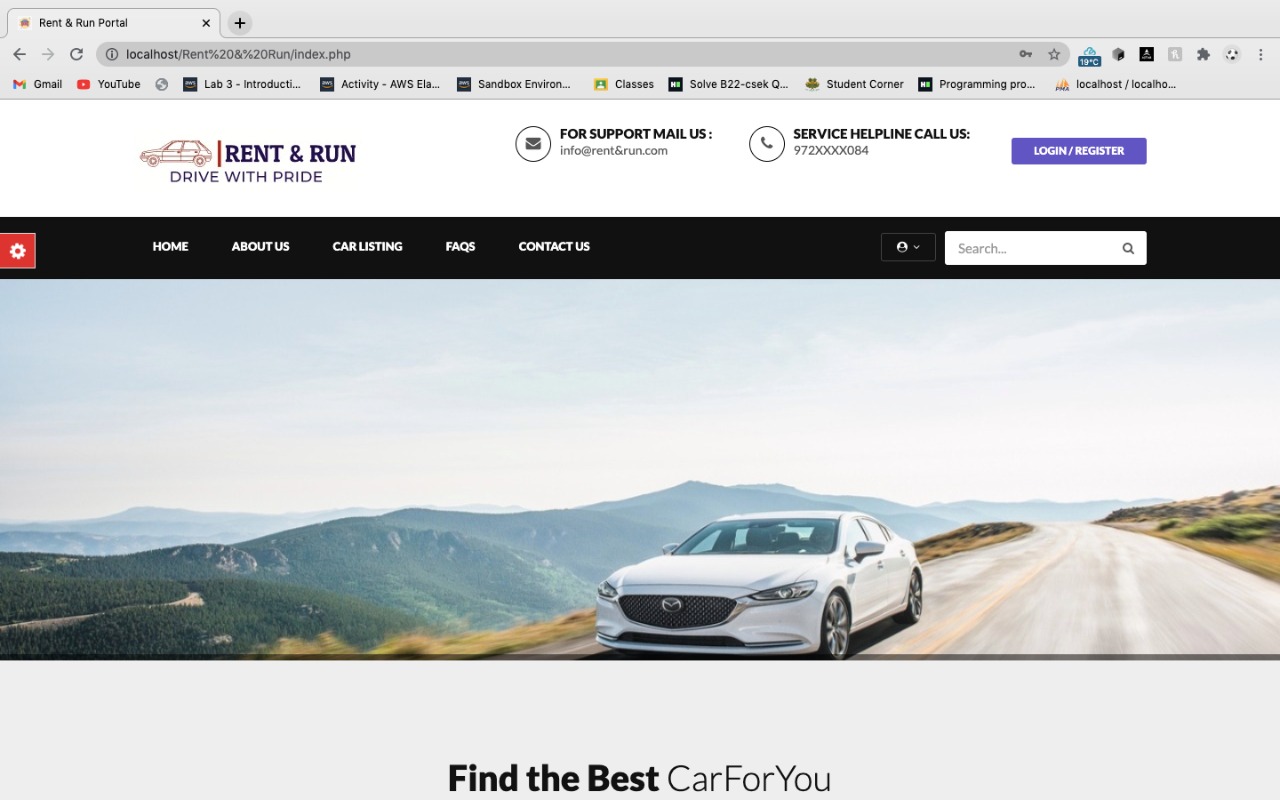
**Displaying error after giving invalid details;**

****

**Catogeries of cars:**

****

**Frontpage:**

****

**Conclustion:**

Rent and Run is a Web application, and it is restricted to only limited type of users. In this application, Different types of user and admins have been given access rights and they are restricted up to their functionalities, so that the data is maintained securely, and redundant data is prevented. As the Data is stored electronically, it is necessary to have a Computer and Network connection to access the Application. Here the Details of user, cars are maintained.

**Future Enhancements**

Every Edition of a book comes with new topics and modifications if any errors are present. In the similar way, in near future, our application will overcome the flaws if occurred, and attains new features offered to users for the Flexible and easy Transportation. Following are the Enhancements to the application.

* Providing Good User Interface.
* Try to Implement the GPS system in the Car.