Team: System Interface Management Providers (S.I.M.P)

**Project 10 – Car Rental Website**

Software Design Document

Name (s): Belu Florin Andrei, Barbu Constantin-Mihai,

Marinescu Nanina-Andreea, Pîrvulescu Andreea-Beatrice

Lab Section: Software Engineering

Workstation: Desktop Computer (Intel Core i7-8700, NVIDIA GeForce GTX 970, 32 GB RAM, 1TB SSD)

Date: (03/22/2020)

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **1.** | **I**[**NTRODUCTION**](#page4) | **2** |
| 1.[1](#page4) | [Purpose](#page4) | 2 |
| 1.[2](#page4) | [Scope](#page4) | 2 |
| 1.3 | Overview | 2 |
| 1.4 | Reference Material | 2 |
| 1.[5](#page4) | [Definitions and Acronyms](#page4) | 2 |
| **2.** | **S**[**YSTEM**](#page4) **O**[**VERVIEW**](#page4) | **2** |
| **3.** | **S**[**YSTEM**](#page4) **A**[**RCHITECTURE**](#page4) | **2** |
| [3.1](#page4) | [Architectural Design](#page4) | 2 |
| [3.2](#page5) | [Decomposition Description](#page5) | 3 |
| [3.3](#page5) | [Design Rationale](#page5) | 3 |
| **4.** | **D**[**ATA**](#page5) **D**[**ESIGN**](#page5) | **3** |
| [4.1](#page5) | [Data Description](#page5) | 3 |
| [4.2](#page5) | [Data Dictionary](#page5) | 3 |
| **5.** | **C**[**OMPONENT**](#page5) **D**[**ESIGN**](#page5) | **3** |
| **6.** | **H**[**UMAN**](#page6) **I**[**NTERFACE**](#page6) **D**[**ESIGN**](#page6) | **4** |
| 6.1 | Overview of User Interface | 4 |
| [6.2](#page6) | [Screen Images](#page6) | [4](#page6) |
| [6.3](#page6) | [Screen Objects and Actions](#page6) | [4](#page6) |
| **7.** | [**REQUIREMENTSMATRIX**](#page6) | **4** |
| **8.** | **A**[**PPENDICES**](#page6) | **4** |

**1. INTRODUCTION**

**1.1 Purpose**

The purpose of this software design document is to describe the implementation, architecture and system design of the car rental website presented in the IEE Software Requirements Specifications document. This website is designed to perform car rental services, such that customers can log in and book their favorite available vehicle, while the administrator can manage the database, add vehicles and confirm rental requests.

Audiences generally vary, however, the targeted audience consists of people in need of transportation, tourists, out-of-towners, car enthusiasts or people that want to test a car before purchasing it.

**1.2 Scope**

This document describes the implementation details of the car rental website. The scope of the website is not only limited to attracting and encouraging people in need of transportation and car enthusiasts to consider using the rental services, but also aims to provide relevant information and various options, keeping in mind that the person visiting this website wants a full knowledge of the availability and facilities of the vehicle to be rented. This website aims to provide a user-friendly set of web pages that are easy to navigate and at the same time provides sufficient depth and information about car rentals.

Objectives relevant to this website include helping the customer with a friendly interface so that they can easily rent a vehicle, assuring the security and discretion of every user, and also making sure the administrator has full control of the database in order to make sure that the website functions without problems and that the customers are satisfied.

Additionally, the benefits of the website are related to how easy it is for a customer to rent a vehicle, customer support, and other quality-of-life features that might be more complicated through other services. The interface makes sure everyone has a clear view of all of their options. Any car can be rented in a small amount of time. Confirmation can be checked at any time on the user’s profile. This certainly makes a car rental website beneficial to the customer, compared to other methods of renting a car.

Finally, the goal of website is to encourage more people to choose rental services, eventually increasing the profit of the company, improvement of the services, and most importantly helping people with their business. Maintaining a simple and friendly site is deemed of highest importance in order to hold customer attention and guide the viewer to information that will lead to a decision to rent the best vehicle available.

**1.3 Overview**.

This document contains several representations of software components, interfaces and data necessary for implementation phase. It will show how the software system will be structured to satisfy the requirements. The IEEE Recommended Practice for Software Design Descriptions have been reduced in order to simplify this assignment while still retaining the main components and providing a general idea of a project definition report.

The Software Design Document is divided into 8 sections with various subsections. The organization for this document is:

**1. Introduction**

1.1 Purpose

1.2 Scope

1.3 Overview

1.4 Reference Material

1.5 Definitions and Acronyms

**2. System Overview**

**3. System Architecture**

3.1 Architectural Design

3.2 Decomposition Description

3.3 Design Rationale

**4. Data Design**

4.1 Data Description

4.2 Data Dictionary

**5. Component Design**

**6. Human Interface Design**

6.1 Overview of User Interface

6.2 Screen Images

6.3 Screen Objects and Actions

**7. Requirements Matrix**

**8. Appendices**

**1.4 Reference Material**

https://www.draw.io/

https://www.toptal.com/freelance/why-design-documents-matter

https://en.wikipedia.org/wiki/Software\_design\_description

**1.5 Definitions and Acronyms**

**XAMPP** - a free and open-source cross-platform web server solution stack package developed by Apache Friends consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

**PHP** - a popular general-purpose scripting language that is especially suited to web development.

**MySQL** - an open-source relational database management system.

**HTML**- Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser.

**CSS**- Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML.

**MariaDB** - a community-developed, commercially supported fork of the MySQL relational database management system

**2. SYSTEM OVERVIEW**

Car rental services have faced many challenges and shortcomings in managing the capabilities, quality and status of their vehicles. The car rental website will utilize a database and with the help of customer feedback it will efficiently keep track of all vehicles which will standardize and improve the efficiency of the car rental company’s management capabilities.

The car rental website is designed to be compatible with services and infrastructure existing in the car rental company. Additionally, car rental website is compliant with all network security protocols as well as regulatory policies.

The car rental website is also compatible with XAMPP, phpMyAdmin and MariaDB to include SQL database files, as well as functional PHP components. The admin panel contained within the car rental website will provide various user interfaces which will allow data entry, updates, tracking, and report generation.

The car rental website will provide the following capabilities:

• User account management, which shows a detailed view of the user’s bookings, testimonies, booking status and other options.

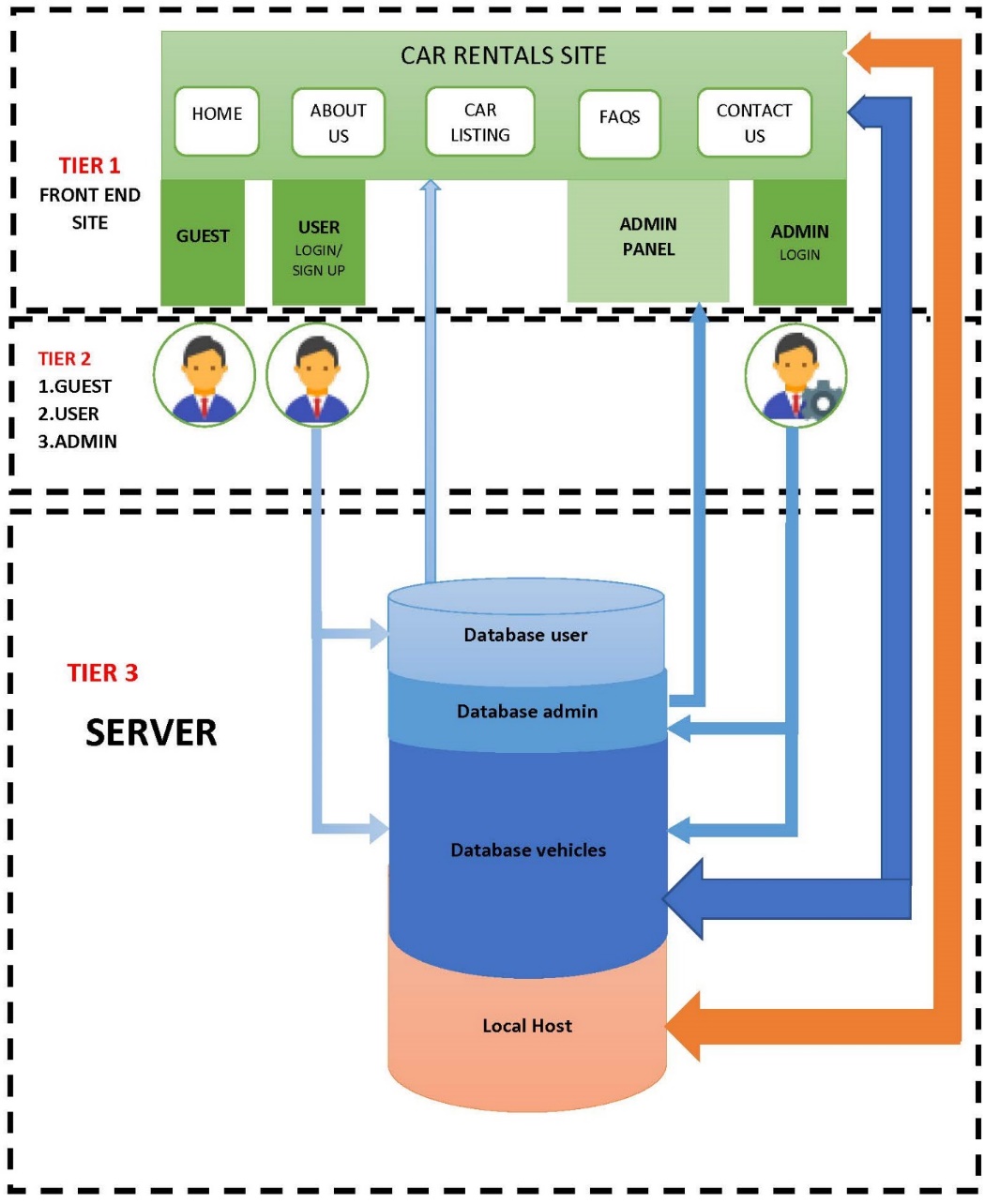
• Integration of all maintenance data which allows for real-time car booking availability and simplifies management of all maintenance activities.

• Enhanced and additional user interfaces which provide users with much simpler data entry, updates, queries, and other capabilities.

• Feedback capabilities which allow users to report to the admin various suggestions, which will be listen in the admin panel. Users can also subscribe to newsletter to receive updates from the administrator.

**3. SYSTEM ARCHITECTURE**

**3.1 Architectural Design**



**Relations between**

Tier 1 – Tier 2

a. Guests can only visualize the car rental website without having access to other User-only options such as booking a car, customizing profile and sending testimonials.

b. User

- can login through the interface;

- can visualize the car rental website;

- is able to book a vehicle, customize their profile and send testimonials;

c. Administrator

- can login through the interface using special admin login button;

- can visualize the car rental website;

- can manage many things through the admin panel, such as adding a vehicle, manage bookings, etc.;

Tier 2 – Tier 3

a. Guest has no relation here. It would be possible to implement a special section in the database for newsletter.

b. User

-the Login interface sends a query to the database that’s on the Apache server;

- as the user performs command such as bookings, it gets added to the database;

c. Administrator

- admin login interface sends a query to the database;

- whenever the administrator adds new vehicles or changes information about users, these changes get added to the database;

Tier 1 – Tier 3

* the data introduced by the user and administrator from the server change the website’s appearance;
* the back-end of this website is located in the local host (XAMPP + Apache Server);

**3.2 Decomposition Description**

**3.3 Design Rationale**

The approach for this car rental website architecture has been selected because it is probably the most common; it is usually built around the database, and many applications in business naturally lend themselves to storing information in tables.

Many of the biggest and best software frameworks—like Java EE, Drupal, and Express—were built with this structure in mind, so many of the applications built with them naturally come out in a layered architecture.

The architecture is arranged so the data enters the top layer and works its way down each layer until it reaches the bottom, which is usually a database. Along the way, each layer has a specific task, like checking the data for consistency or reformatting the values to keep them consistent. It is the standard software development approach offered by most of the popular web frameworks and it is a layered architecture. Just above the database is the model layer, which often contains business logic and information about the types of data in the database. At the top is the view layer, which is often CSS, JavaScript, and HTML with dynamic embedded code. In the middle, you have the controller, which has various rules and methods for transforming the data moving between the view and the model.

The advantage of a layered architecture is the separation of concerns, which means that each layer can focus solely on its role. This makes it:

* Maintainable
* Testable
* Easy to assign separate "roles"
* Easy to update and enhance layers separately

Proper layered architectures will have isolated layers that aren’t affected by certain changes in other layers, allowing for easier refactoring. This architecture can also contain additional open layers, like a service layer, that can be used to access shared services only in the business layer but also get bypassed for speed.

**Best for:**

* New applications that need to be built quickly
* Enterprise or business applications that need to mirror traditional IT departments and processes
* Teams with inexperienced developers who don’t understand other architectures yet
* Applications requiring strict maintainability and testability standards

**4. DATA DESIGN**

**4.1 Data Description**

Problems that can be solved by the transformation of information domains in the system into data structures:

* Code that looped through the result set of a SQL response and filtered items. Inserted the items into an array to return by re-dimensioning the array to hold 1 more item each time.
* Poorly written, custom code to compare two arrays of data.
* Using lists to store unique keyed data instead of a dictionary or hash map, and then not handling the duplicated data resulting in exceptions being thrown.
* Array List abuse and not thought through loops causing poor run times.

**4.2 Data Dictionary**

## **Table structure for admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Tip** | **Nul** | **Implicit** |
| ***id*** | int(11) | Nu |  |
| UserName | varchar(100) | Nu |  |
| Password | varchar(100) | Nu |  |
| updationDate | timestamp | Nu | 0000-00-00 00:00:00 |

## **Table structure for tblusers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Tip** | **Nul** | **Implicit** |
| ***id*** | int(11) | Nu |  |
| FullName | varchar(120) | Da | NULL |
| EmailId | varchar(100) | Da | NULL |
| Password | varchar(100) | Da | NULL |
| ContactNo | char(11) | Da | NULL |
| dob | varchar(100) | Da | NULL |
| Address | varchar(255) | Da | NULL |
| City | varchar(100) | Da | NULL |
| Country | varchar(100) | Da | NULL |
| RegDate | timestamp | Da | current\_timestamp() |
| UpdationDate | timestamp | Da | NULL |

## **Table structure for tblbrands**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Tip** | **Nul** | **Implicit** |
| ***id*** | int(11) | Nu |  |
| BrandName | varchar(120) | Nu |  |
| CreationDate | timestamp | Da | current\_timestamp() |
| UpdationDate | timestamp | Da | NULL |

## **Table structure for tblbooking**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Tip** | **Nul** | **Implicit** |
| ***id*** | int(11) | Nu |  |
| userEmail | varchar(100) | Da | NULL |
| VehicleId | int(11) | Da | NULL |
| FromDate | varchar(20) | Da | NULL |
| ToDate | varchar(20) | Da | NULL |
| message | varchar(255) | Da | NULL |
| Status | int(11) | Da | NULL |
| PostingDate | timestamp | Nu | current\_timestamp() |

## **Table structure for tblvehicles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Tip** | **Nul** | **Implicit** |
| ***id*** | int(11) | Nu |  |
| VehiclesTitle | varchar(150) | Da | NULL |
| VehiclesBrand | int(11) | Da | NULL |
| VehiclesOverview | longtext | Da | NULL |
| PricePerDay | int(11) | Da | NULL |
| FuelType | varchar(100) | Da | NULL |
| ModelYear | int(6) | Da | NULL |
| SeatingCapacity | int(11) | Da | NULL |
| Vimage1 | varchar(120) | Da | NULL |
| Vimage2 | varchar(120) | Da | NULL |
| Vimage3 | varchar(120) | Da | NULL |
| Vimage4 | varchar(120) | Da | NULL |
| Vimage5 | varchar(120) | Da | NULL |
| AirConditioner | int(11) | Da | NULL |
| PowerDoorLocks | int(11) | Da | NULL |
| AntiLockBrakingSystem | int(11) | Da | NULL |
| BrakeAssist | int(11) | Da | NULL |
| PowerSteering | int(11) | Da | NULL |
| DriverAirbag | int(11) | Da | NULL |
| PassengerAirbag | int(11) | Da | NULL |
| PowerWindows | int(11) | Da | NULL |
| CDPlayer | int(11) | Da | NULL |
| CentralLocking | int(11) | Da | NULL |
| CrashSensor | int(11) | Da | NULL |
| LeatherSeats | int(11) | Da | NULL |
| RegDate | timestamp | Nu | current\_timestamp() |
| UpdationDate | timestamp | Da | NULL |

## **Table structure for tblcontactusinfo**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Tip** | **Nul** | **Implicit** |
| ***id*** | int(11) | Nu |  |
| Address | tinytext | Da | NULL |
| EmailId | varchar(255) | Da | NULL |
| ContactNo | char(11) | Da | NULL |

## **Table structure for tblpages**

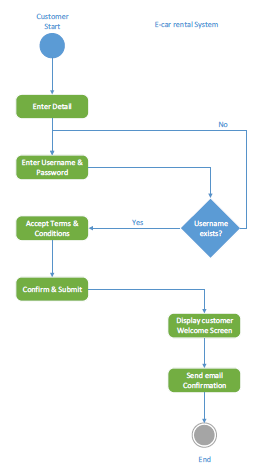
|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Tip** | **Nul** | **Implicit** |
| ***id*** | int(11) | Nu |  |
| PageName | varchar(255) | Da | NULL |
| type | varchar(255) | Nu |  |
| detail | longtext | Nu |  |

## **Table structure for tblcontactusquery**

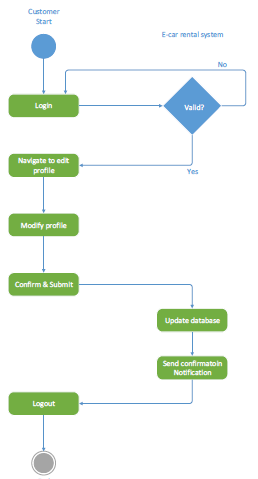
|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Tip** | **Nul** | **Implicit** |
| ***id*** | int(11) | Nu |  |
| name | varchar(100) | Da | NULL |
| EmailId | varchar(120) | Da | NULL |
| ContactNumber | char(11) | Da | NULL |
| Message | longtext | Da | NULL |
| PostingDate | timestamp | Nu | current\_timestamp() |
| status | int(11) | Da | NULL |

**5. COMPONENT DESIGN**

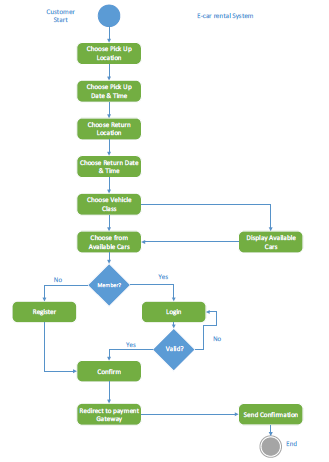
**Member Registration**



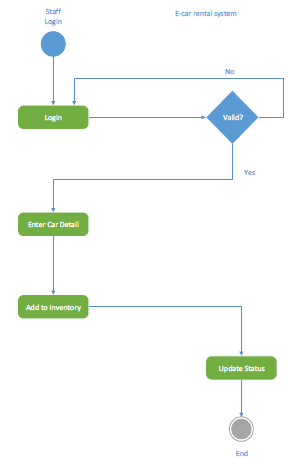
**Profile Modification**



**Reservation of Car**



**Adding a New Car**



**6. HUMAN INTERFACE DESIGN**

**6.1 Overview of User Interface**

Based on the user’s perspective, upon accessing the website they will see the home page, where they can see an announcement, recommended vehicles, a newsletter and a miniature “about us” menu with several selections (newsletter and ‘about us’ miniature menu shall be the footer, visible on all other pages). They will be able to browse all pages, such as ‘Car Listings’, ’About Us’, ‘FAQ’, ‘Contact Us’ (this will be the header, visible on all pages, along with the website logo, login button and other information). Everyone starts as a guest-type user, and because this is a car rental service website, they must click on the “login/register” button to be able to use the full functionalities of the website. Of course, guest-type users are still able to view most of the website and even subscribe to the newsletter and utilize the ‘Contact us’ page, where they can contact the administrator. Newsletter subscription requires that they enter an email address.

Unsuccessful registrations/logins will lead to a warning. Failing to complete mandatory fields will also lead to a warning. Once logged in, the user will be able to book the vehicles listed in the ‘Car Listings’ page. This page is equipped with a search function which contains categories such as vehicle brand and fuel type. It also contains a list of recently listed vehicles. The logged user can select any vehicle and view the details about it. An overview, registered year, fuel type, seats, rental price and a list of accessories are available for all to see. Here, the booking function is also available. The user must enter the booking period (From <Date>, To <Date>) and a short message regarding their request. If successful, the user will be alerted by a message. Then, they can access their user profile located in the drop-down menu where their username is displayed in the header and check their bookings and their status. They must then wait for the administrator’s approval. On a side note, in this drop-down menu, the user can also customize their personal information, update their password, send a testimonial to the administrator, view them and finally, sign out.

User interface share following qualities or characteristics:

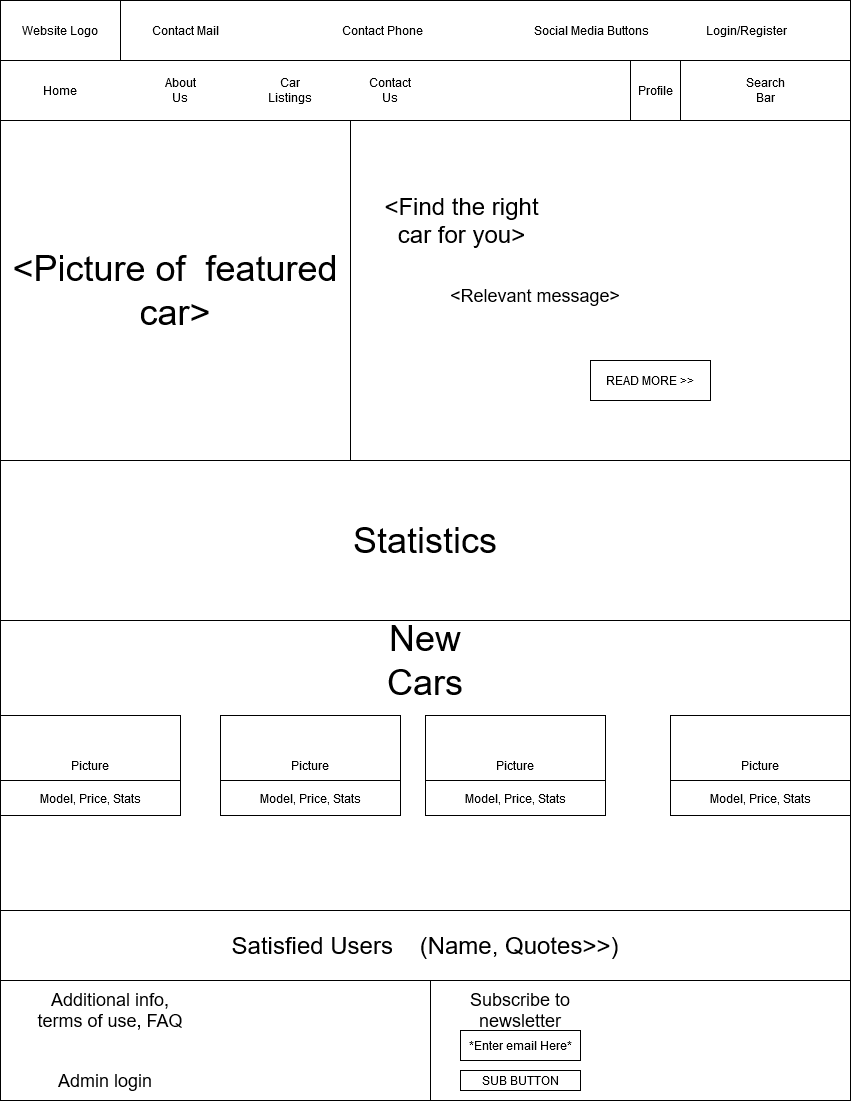
**Clarity**: The interface avoids ambiguity by making everything clear through language, flow, hierarchy and metaphors for visual elements.

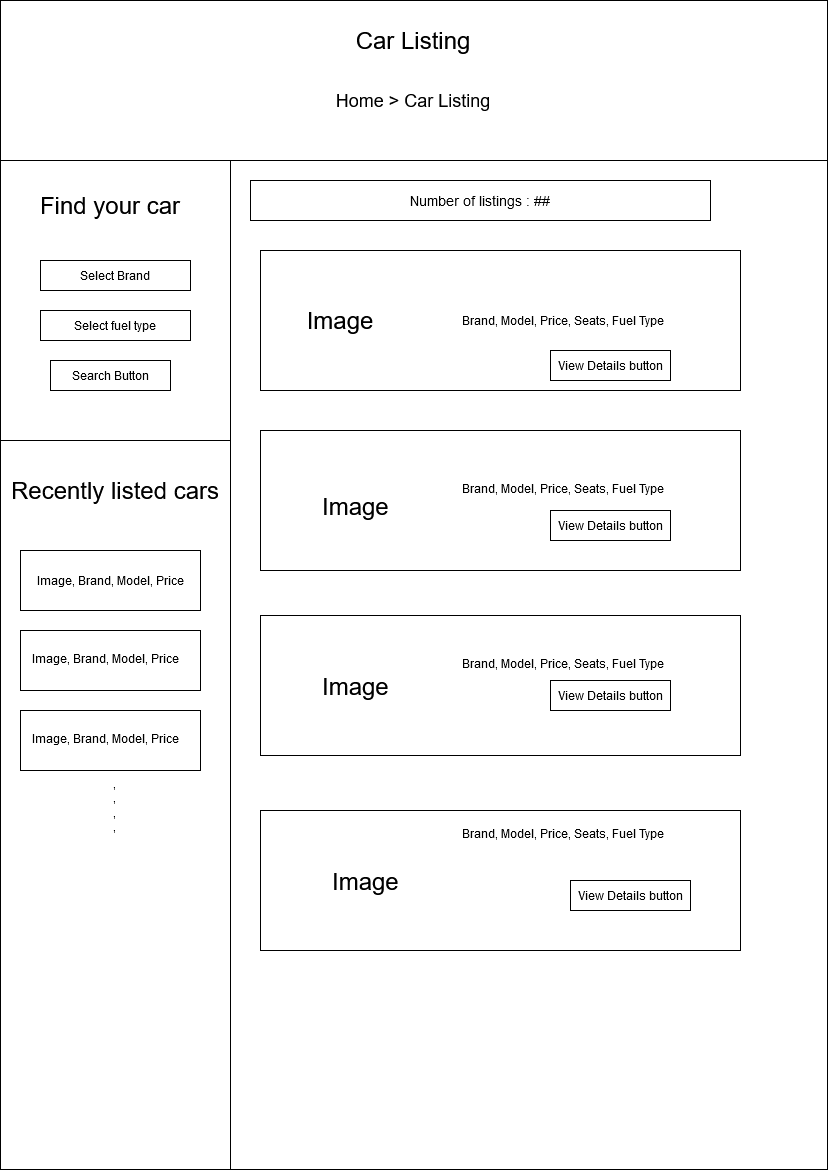
**Responsiveness**: This means a couple of things. First, responsiveness means speed: a good interface should not feel sluggish. Secondly, the interface should provide good feedback to the user about what’s happening and whether the user’s input is being successfully processed.

**Aesthetics**: While you don’t need to make an interface attractive for it to do its job, making something look good will make the time your users spend using your application more enjoyable; and happier users can only be a good thing.

**Efficiency**: Time is money, and a great interface should make the user more productive through shortcuts and good design.

**6.2 Screen Images**





**6.3 Screen Objects and Actions**

Interface elements include:

* **Input Controls**: dropdown lists, list boxes, buttons, toggles(carousel), text fields
* **Navigational Components**: breadcrumb, slider, search field, pagination, icons
* **Informational Components**: tooltips, icons, notifications, message boxes, modal windows
* **Containers**: accordion

**7. REQUIREMENTS MATRIX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **REQUIREMENTS**  **TRACEABILITY MATRIX** | | | | | | |
| Project Name: Car Rental Website | | | | | | |
| User class ID# | User Class Name | Functional Requirement ID# | Functional Requirement Name/Use Case | System Component | Priority | Test Case ID# |
| UC\_1 | Customer User | 1.1 | Website access | Link | High | #TC001 |
|  |  | 1.2 | Update notification | Newsletter | Medium | #TC002 |
|  |  | 1.3 | User Registration | Login/Register | High | #TC003 |
|  |  | 1.4 | User Login | Login/Register | High | #TC004 |
|  |  | 1.5 | Password retrieval | Login/Register | High | #TC005 |
|  |  | 1.6 | Rental search | Search module | High | #TC006 |
|  |  | 1.7 | Rental selection & booking | Booking module | High | #TC007 |
|  |  | 1.8 | Profile update | Profile module | High | #TC008 |
|  |  | 1.9 | Check bookings | Profile module | High | #TC009 |
|  |  | 1.10 | Post testimonial | Profile module | High | #TC010 |
|  |  | 1.11 | Sign out and Log back in | Login/Register | High | #TC011 |
| UC\_2 | Administrator User | 2.1 | Admin Login | Super Login module | High | #TC012 |
|  |  | 2.2 | Create/Delete Vehicle listing | Admin Panel  module | High | #TC013 |
|  |  | 2.3 | Post Vehicle listing | Admin Panel  module | High | #TC014 |
|  |  | 2.4 | Confirm/Cancel customer booking | Admin Panel  module | High | #TC015 |
|  |  | 2.5 | Manage testimonial | Admin Panel  module | High | #TC016 |
|  |  | 2.6 | Contact us query management | Admin Panel  module | High | #TC017 |
|  |  | 2.7 | Check User Details | Admin Panel  module | High | #TC018 |
|  |  | 2.8 | Update public details | Admin Panel  module | High | #TC019 |
|  |  | 2.9 | Manage newsletter subscribers | Admin Panel  module | High | #TC020 |
|  |  | 2.10 | Dashboard menu statistics | Admin Panel  module | High | #TC021 |
|  |  | 2.11 | Change admin password | Admin Panel  module | High | #TC022 |
|  |  | 2.12 | Log out of Admin Panel | Admin Panel  module | High | #TC023 |
| UC\_3 | Guest User | 3.1 | Explore website | Link | High | #TC024 |
|  |  | 3.2 | Booking restriction | Login/Register  prompt | High | #TC025 |
|  |  | 3.3 | Contact admin | Contact us page | High | #TC026 |
|  |  | 3.4 | Transition into Customer User | Login/Register | High | #TC027 |
|  |  | 3.5 | No interaction with admin, booking-wise | Login/Register  prompt | High | #TC028 |
|  |  | 3.6 | Update notification | Newsletter | Medium | #TC029 |
|  |  | 3.7 | Encounter illegal action | Login/Register prompt | High | #TC030 |

**8. APPENDICES**

This section is optional.