Software Design Document (SDD) Template

Software design is a process by which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. The SDD shows how the software system will be structured to satisfy the requirements. It is the primary reference for code development and, therefore, it must contain all the information required by a programmer to write code. The SDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage, i.e. the detailed design stage, more detailed data structures are defined and algorithms are developed for the defined architecture.

This template is an annotated outline for a software design document adapted from the IEEE Recommended Practice for Software Design Descriptions. 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. For your own information, please refer to [IEEE Std 1016­1998](http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee-SDD.pdf)1 for the full IEEE Recommended Practice for Software Design Descriptions.

1 <http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee>­SDD.pdf

CarRental

# Software Design Document

Names:

Piorescu-Oprea Ovidiu Ilie

Munteanu Stefan

Neagoe Cristi-Florin

Lab Section:

Workstation:

Date: 03/26/2021

**TABLE OF CONTENTS**

1. [**INTRODUCTION 2**](#_gjdgxs)
   1. [Purpose 2](#_30j0zll)
   2. [Scope 2](#_1fob9te)
   3. [Overview 2](#_3znysh7)
   4. [Reference Material 2](#_2et92p0)
   5. [Definitions and Acronyms 3](#_tyjcwt)
2. [**SYSTEM OVERVIEW 4**](#_3dy6vkm)
3. [**SYSTEM ARCHITECTURE 5**](#_1t3h5sf)
   1. [Architectural Design 5](#_4d34og8)
   2. [Decomposition Description 7](#_2s8eyo1)
   3. [Design Rationale 14](#_17dp8vu)
4. [**DATA DESIGN 15**](#_3rdcrjn)
   1. [Data Description 15](#_26in1rg)
   2. [Data Dictionary 15](#_lnxbz9)
5. [**COMPONENT DESIGN 15**](#_35nkun2)
6. [**HUMAN INTERFACE DESIGN 15**](#_1ksv4uv)
   1. [Overview of User Interface 15](#_44sinio)
   2. [Screen Images 16](#_2jxsxqh)
   3. [Screen Objects and Actions 19](#_z337ya)
7. [**REQUIREMENTS MATRIX 19**](#_3j2qqm3)
8. [**APPENDICES 19**](#_1y810tw)

### INTRODUCTION

## Purpose

The purpose of this document is to record the requirements for the design and

development of a car rental service website. It describes requirements for a future, complete version of the project, to be created in the spring of 2021. There may be a need for future updates to the document.

Moreover, this document reflects the current requirements of the project as understood by the project team, presenting an initial description of the various functionalities and services provided by the software. Additionally, it may also serve as a basis for testing by the user.

## Scope

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.

## Overview

The first part (Introduction) of this document is a short description of our application, the purpose of SDD document, acronyms, definitions and some references.

The second part (System Overview) of this document is a short description of our system and the use case diagrams are presented.

The third part (System Architecture) of this document describes each level of high-level architecture diagram.

The fourth part (Data design) of this document describes the relational model.

The fifth part (Human Interface Design) of this document are some mockups, a short description for each page and functionalities of our application.

The sixth part (Requirement’s matrix) of this document Provide a cross reference that traces components and data structures to the requirements in your SRS document.

## Reference Material

|  |  |  |
| --- | --- | --- |
| No. | Name | Link |
| 1 | Java Tutorials | [*https://www.javatpoint.com/java-tutorial*](https://www.javatpoint.com/java-tutorial) |
| 2 | Hibernate Tutorials | [*https://www.tutorialspoint.com/hibernate/index.htm*](https://www.tutorialspoint.com/hibernate/index.htm) |
| 3 | Spring Tutorials | [*https://www.baeldung.com/spring-tutorial*](https://www.baeldung.com/spring-tutorial) |
| 4 | List of computing and IT abbreviations | [*https://en.wikipedia.org/wiki/List\_of\_computing\_and\_IT\_abbreviations*](https://en.wikipedia.org/wiki/List_of_computing_and_IT_abbreviations) |
| 5 | Web Tutorials | *https://www.w3schools.com/* |
| 6 | Html + CSS + PHP-MySQL Tutorials | *https://marplo.net/* |

## Definitions and Acronyms

|  |  |  |
| --- | --- | --- |
| **No.** | **Abbreviation** | **Definition of abbreviation** |
| 1 | **XAMPP** | Cross-platform web server solution stack package. |
| 2 | **MySQL** | Structured Query Language |
| 3 | **UI** | User Interface |
| 4 | **SRS** | Software Requirement Specification |
| 5 | **OS** | Operating System |
| 6 | **PHP** | Personal Home Page, general-purpose scripting language |
| 7 | **HTML** | Hypertext Markup Language |
| 8 | **PC** | Personal Computer |
| 9 | **IP** | Internet Protocol |
| 10 | **HTTP** | Hypertext Transfer Protocol |
| 11 | **HTTPS** | Hypertext Transfer Protocol Secure |
| 12 | **CSS** | Cascading Style Sheets |

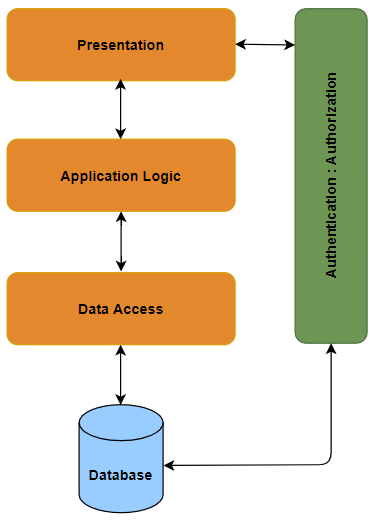
### SYSTEM OVERVIEW

This application is made for users that want to rent a car, just with the help of the internet. After creating the online car renting application, users will be able to connect with a username and a password. The users can create an account and rent a car, can search for a specific car, see all the cars he can rent,etc.



### SYSTEM ARCHITECTURE

## Architectural Design



The most common architecture is the layered architecture pattern, otherwise known as the n-tier architecture pattern.

**Presentation layer** would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request. Also, the presentation layer doesn’t need to know or worry about how to get customer data, as it only needs to display that information on a screen in particular format.

The **application layer** doesn’t need to be concerned about how to format customer data for display on a screen or even where the customer data is coming from, as it only needs to get the data from the data access layer, perform application logic against the data (e.g.: calculate values or aggregate data) and pass that information up to the presentation layer.

**Data access layer** in computer software is a layer of a computer program which provides simplified access to data stored in persistent storage of some kind, such as an entity-relational database.

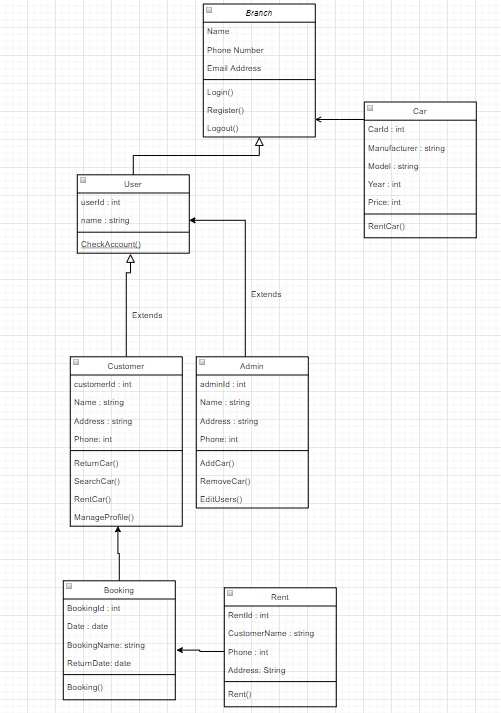
The **database layer** takes care of data access. An object from the database layer can write itself to one or more tables. In the database layer, you’ll find things like database, connection, table, SQL and result set.

**Authentication** is the process of verifying the identity of an individual. A unique identifier is associated with a user which is the username. A combination of username and password are used in order to authenticate a user. A user can interact with a web application using multiple actions. Access to certain actions or pages can be restricted. **Authorization** is the process of controlling user access through assigned roles and privileges.

The layered architecture pattern is a solid general-purpose pattern, making it a good starting point for most applications, particularly when you are not sure what architecture pattern is best suited for your application.

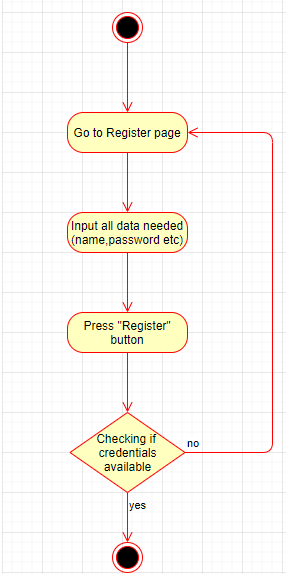
## Decomposition Description

**Class Diagram**

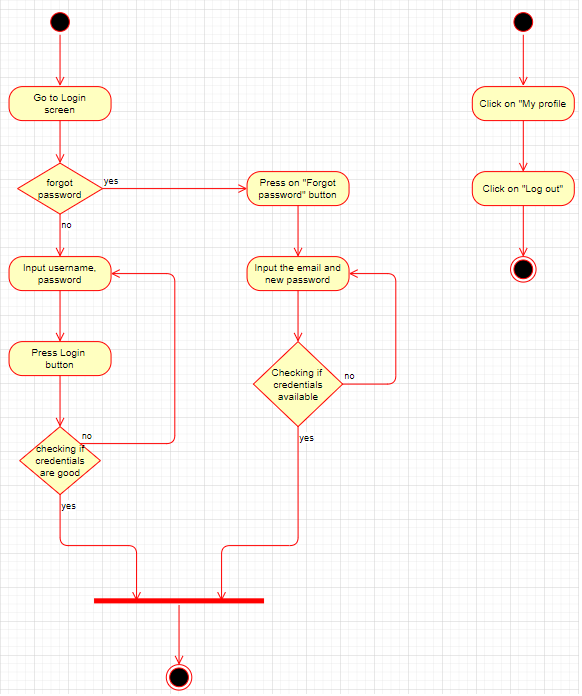
****

**Activity Diagram**

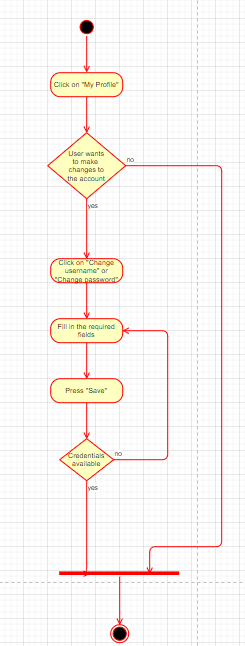
**User – Create Account**



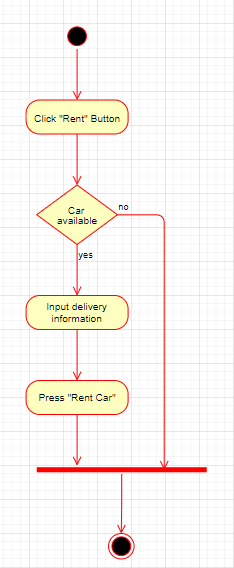
**User – Log in and Log Out**



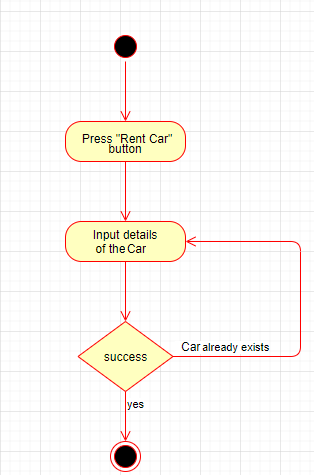
**Profile Manage**



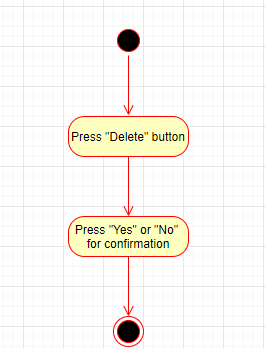
**Rent Car**



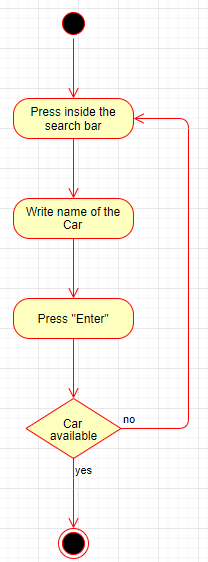
**Admin – Add Car**



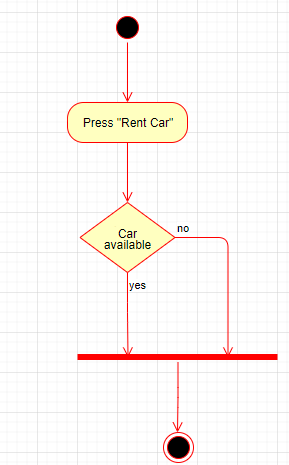
**Admin – Remove Car**



**Car Search**



**Car Availability Check**

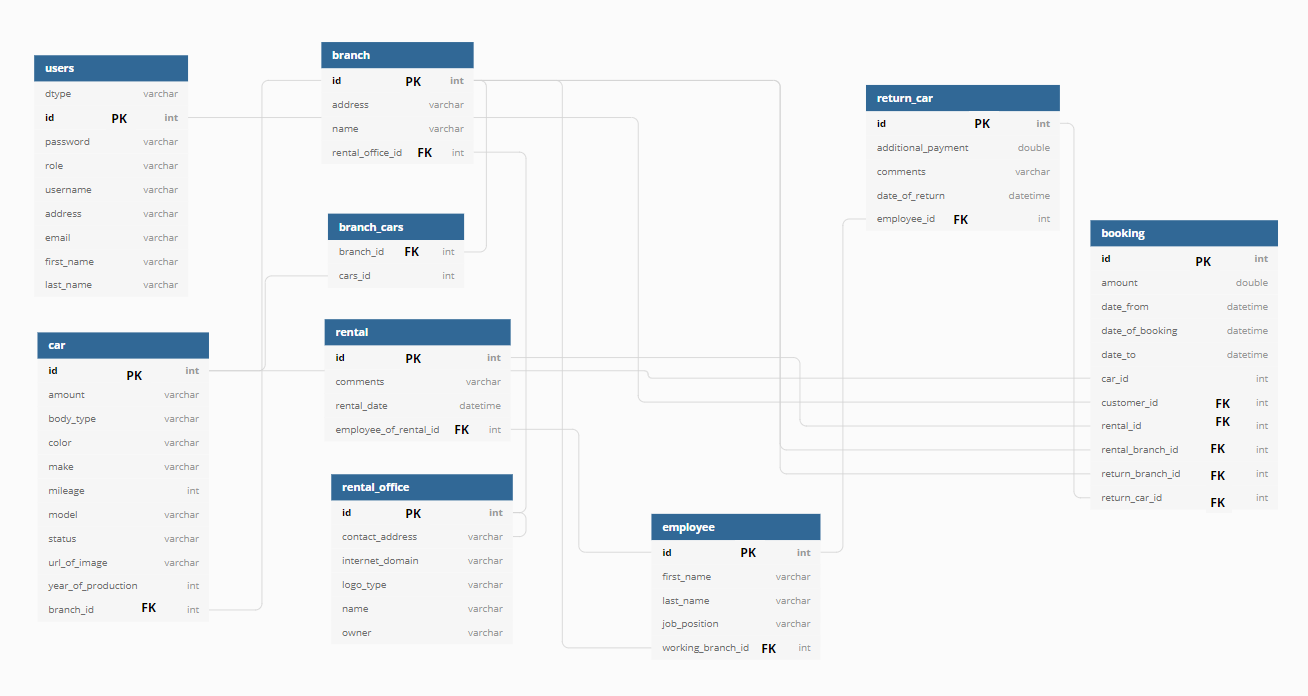


## Design Rationale

Discuss the rationale for selecting the architecture described in 3.1 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn’t choose them.

### DATA DESIGN

## Data Description



## Data Dictionary

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 3.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

### COMPONENT DESIGN

Not applicable

### HUMAN INTERFACE DESIGN

## Overview of User Interface

Users can be either logged in or not. The logged in users can access the whole website, see the description of the cars and will be able to rent them, while the users that are not logged in

cannot see anything besides just some information. Also, there are admins that are able to see all the cars that are rented, to delete or add certain cars, see the information about the users and

update descriptions about a car.

The functionalities:

* User can create an account
* User/Admin can log in
* User/Admin can log out
* User/Admin account can be deleted
* User/Admin can rent a car
* User/Admin can see all the cars he rented
* User/Admin can return the car
* User/Admin can search for a car
* User/Admin can see his profile information
* User/Admin can change his profile information
* The admin can see all the cars that are rented
* The admin can add a new car
* The admin can remove a car
* The admin can update information about a car
* The admin can see the information about the users

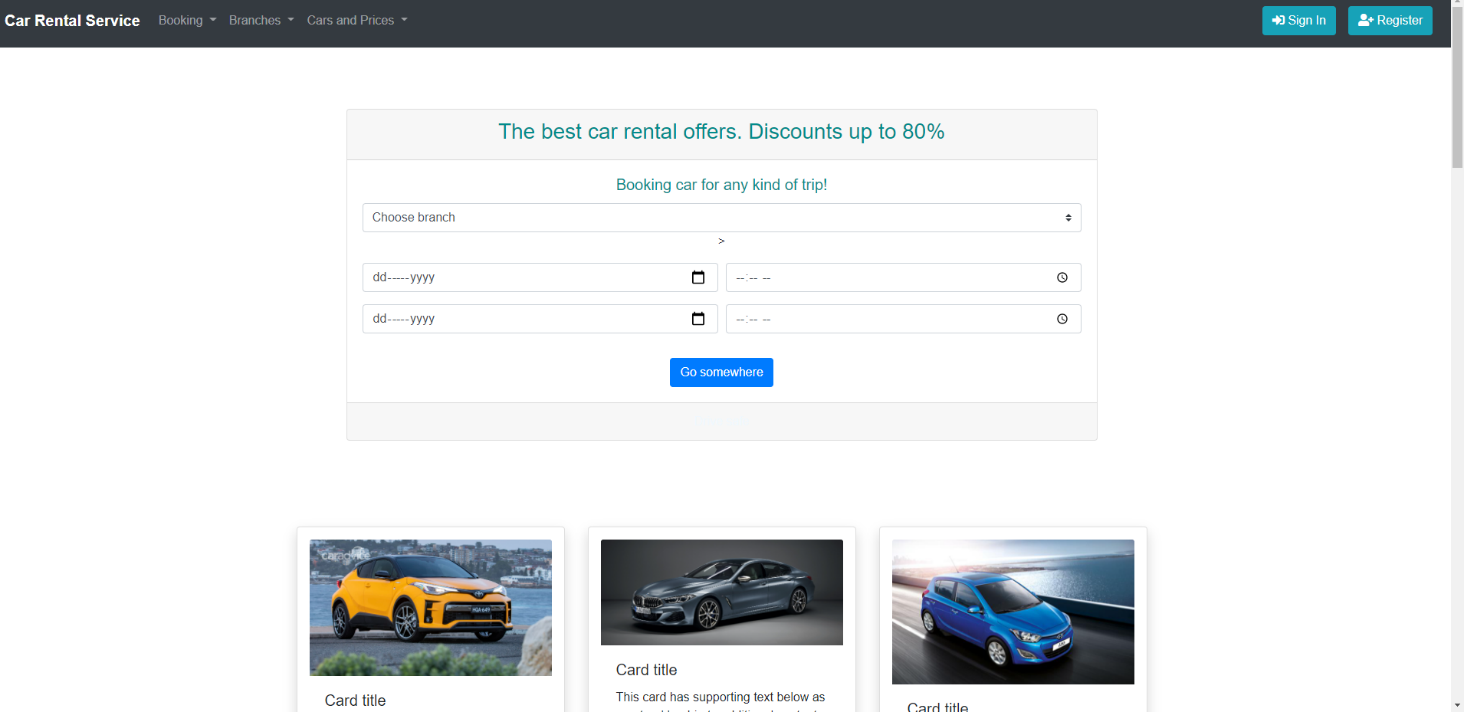
When opening the CarRental website, there will be the home page. The user can log in if he’s already registered, otherwise he can register by introducing the first name, last name, username, password, email, address and phone number. If logged in as a user, instead of the login/register button should be the “My profile”. In the first page is also presented a lot of cars and search bar.

Also if the user forgot his password, he can press ”Forgot password” and a new tab will appear and should write email, password and new password.

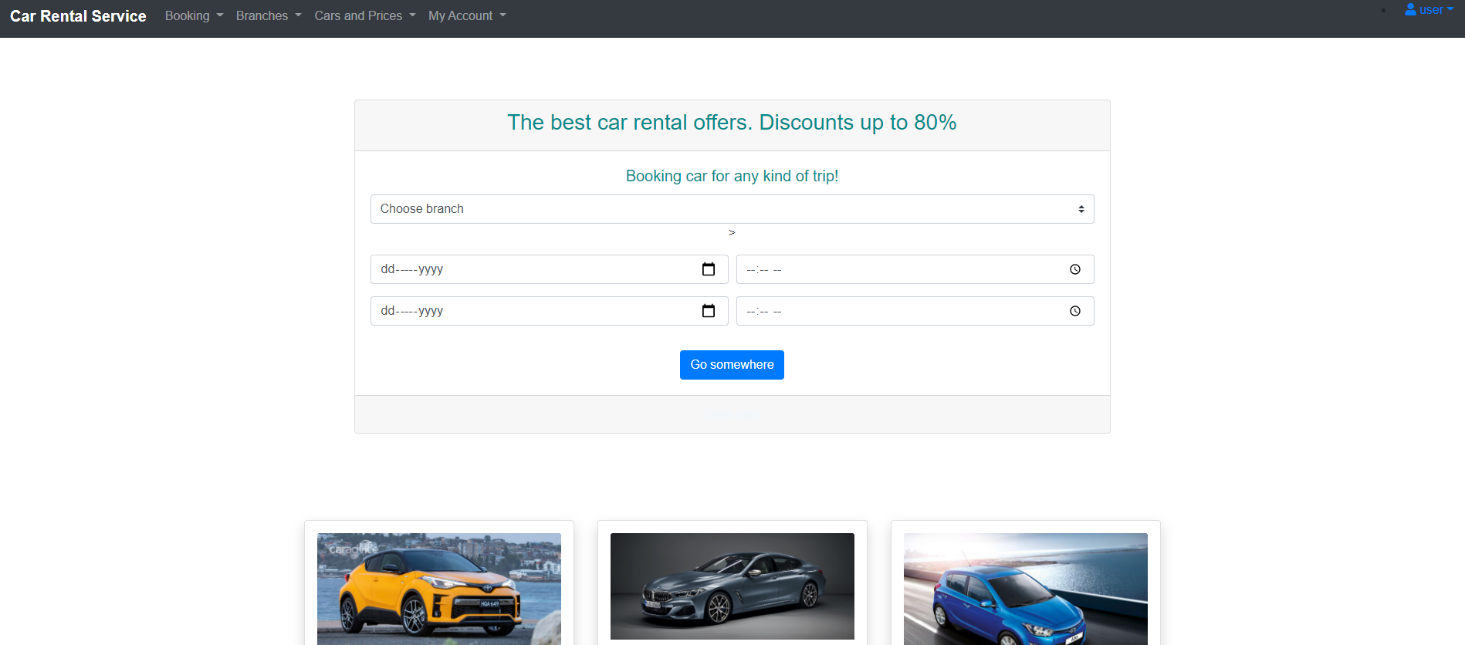
By clicking on a car from home page the user will be redirected to another page and here will be displayed some information about the car. If the user wants to rent a car, he can click on “Rent” button and will be asked to fill in some information.

## Screen Images

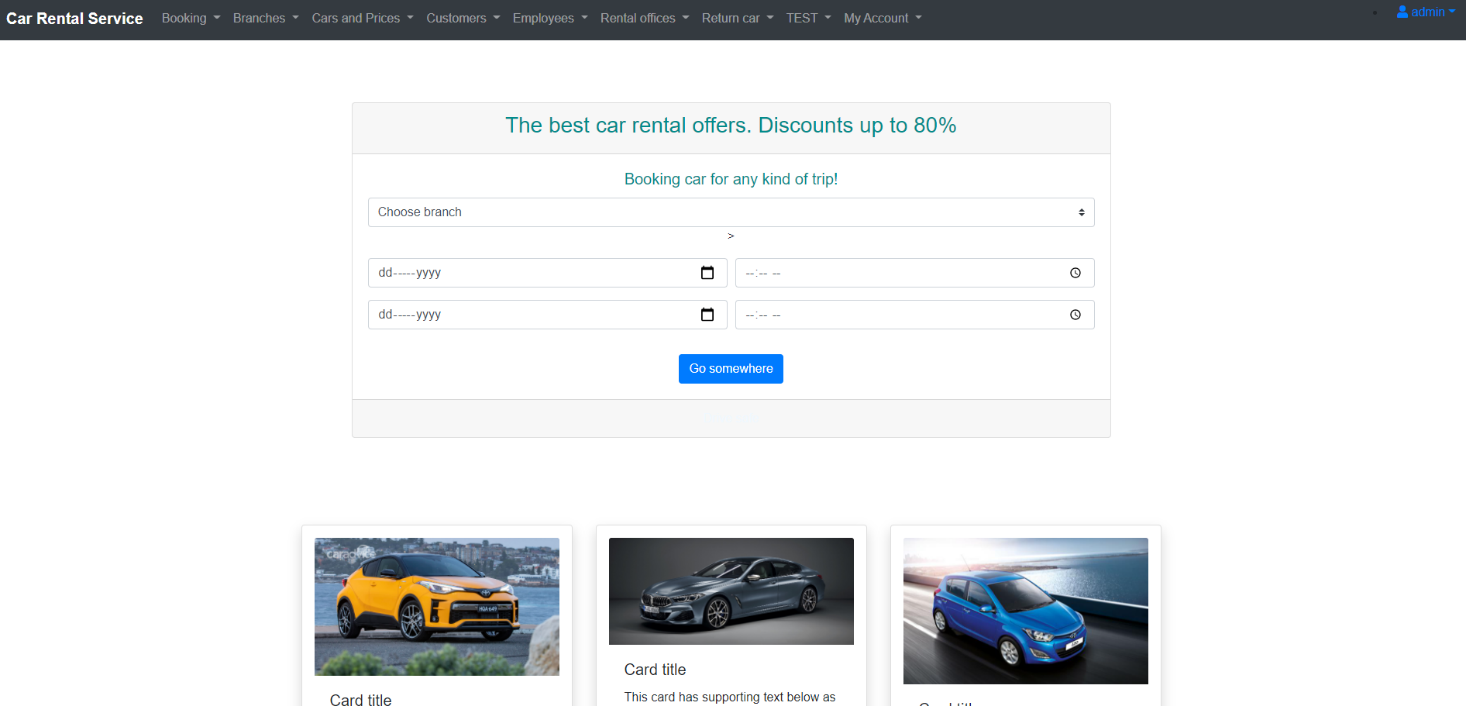
When opening the website, the homepage will be the first thing that appears. This page consists of some of the cars that our website will provide, a search bar and some functionalities like “Homepage”, “Vehicles”, “Rent” etc.



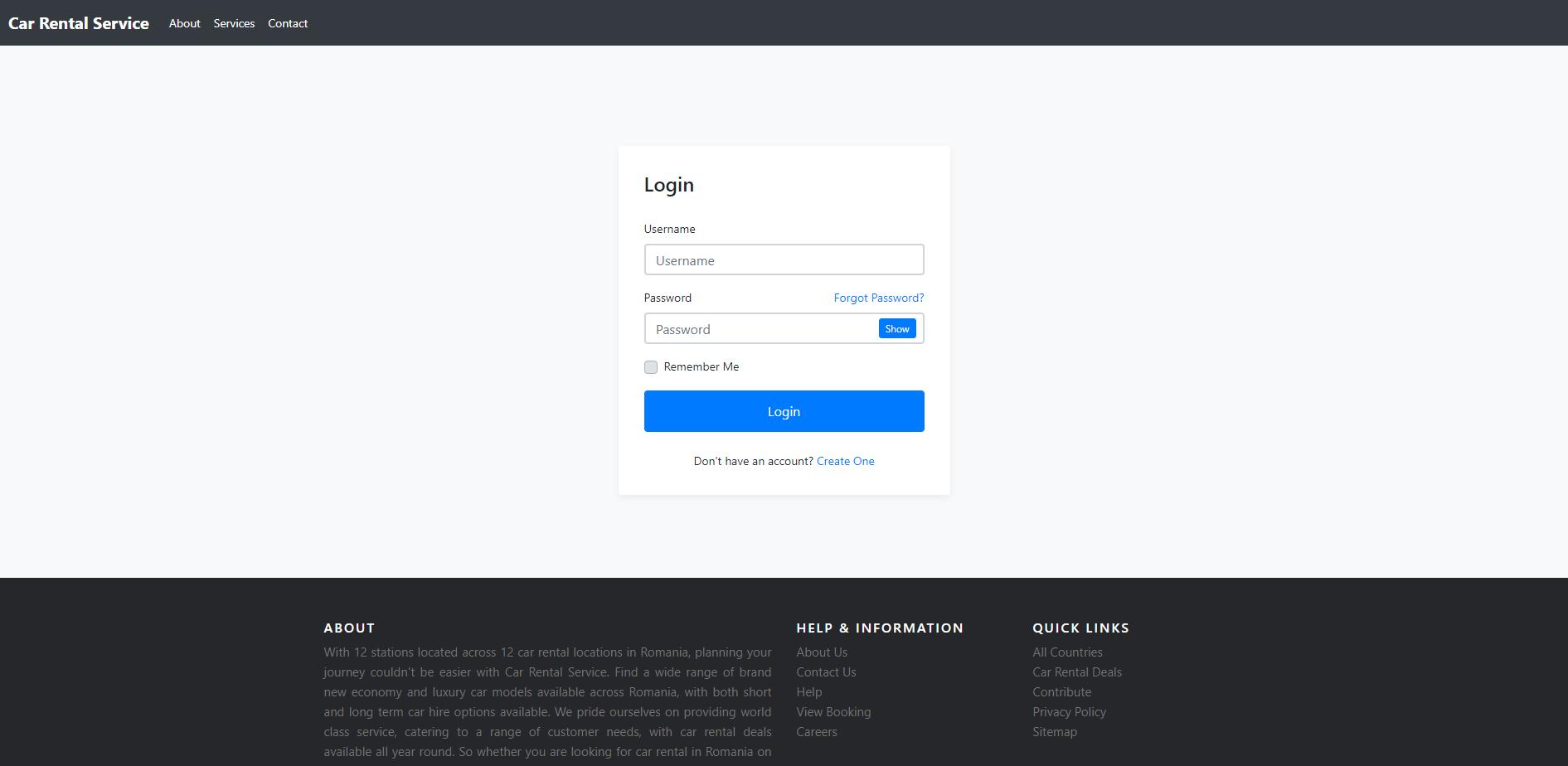
Once the user is logged in, the Log In interface won’t show up anymore. Instead, a My Account button will appear and his username.



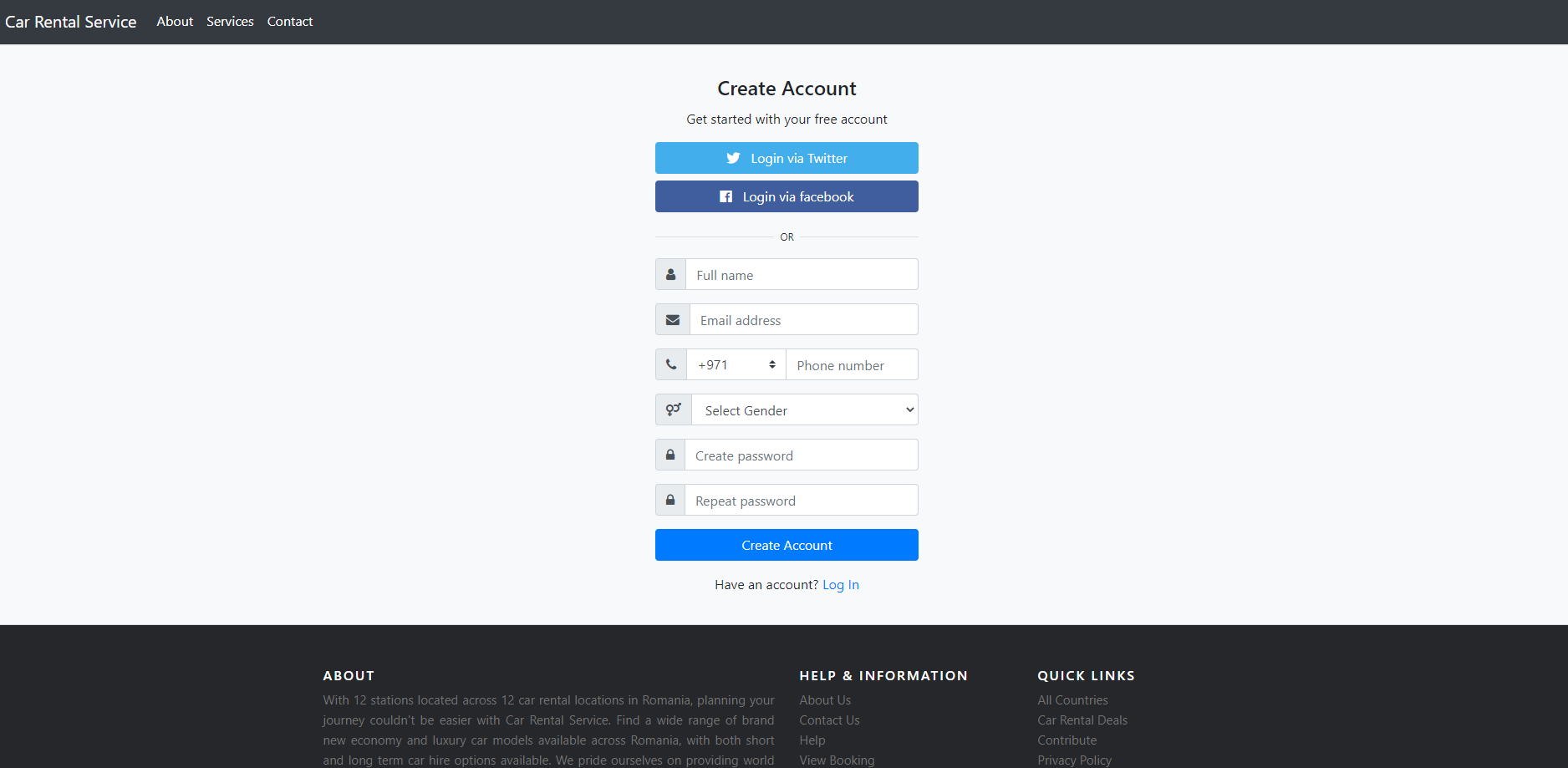
The Homepage from an admin point of view (an admin can delete cars and much more).



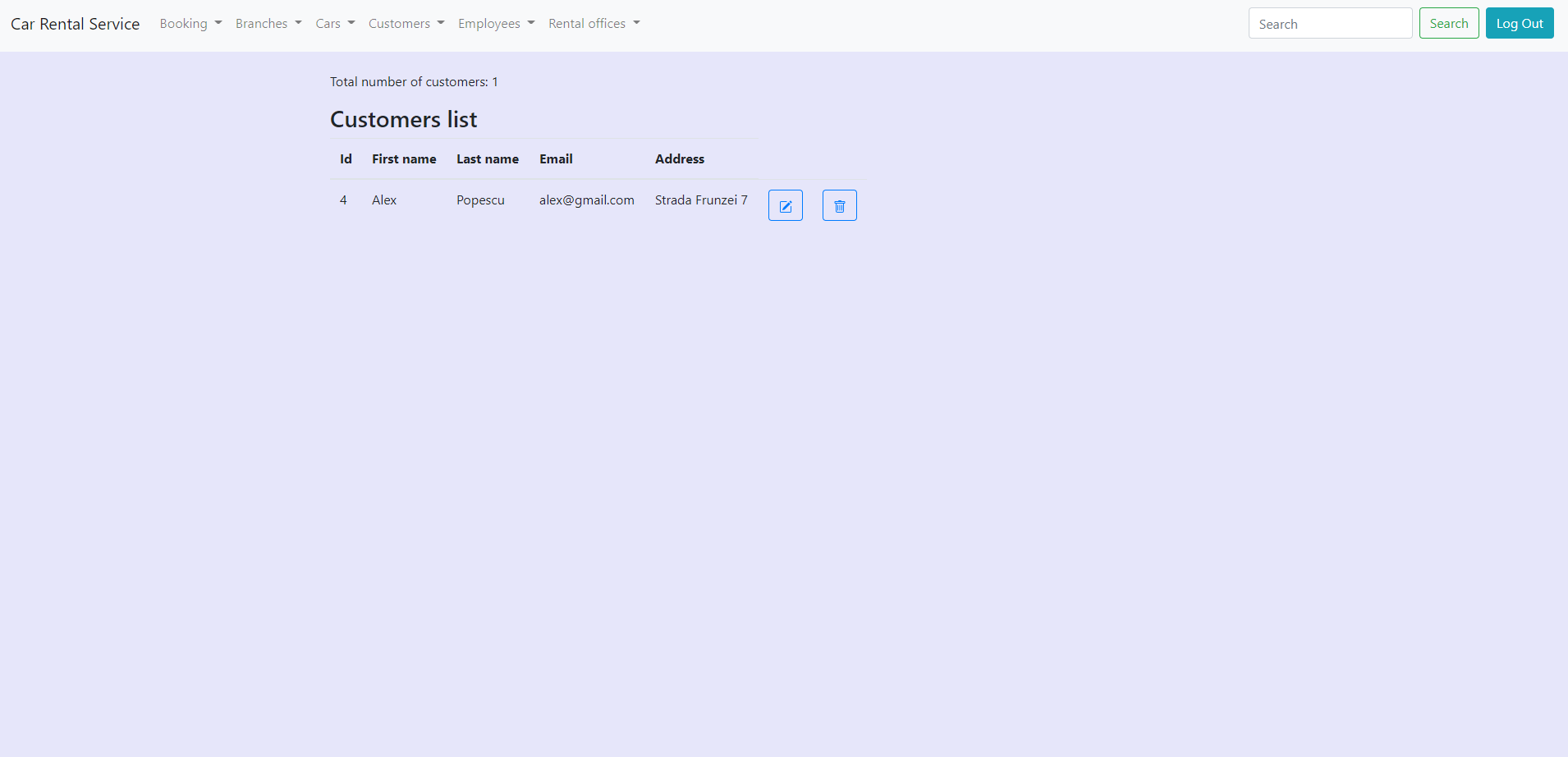
By pressing the “Login” button, the user will be redirected to the login page.



By pressing the “Register” button, the user will be redirected to the registration page.



Admin Customer List Page.



### REQUIREMENTS MATRIX

Provide a cross reference that traces components and data structures to the requirements in your SRS document.

Use a tabular format to show which system components satisfy each of the functional

requirements from the SRS. Refer to the functional requirements by the numbers/codes that you

gave them in the SRS.

### APPENDICES

*This section is optional.*

Appendices may be included, either directly or by reference, to provide supporting details that could aid in the understanding of the Software Design Document.

Appendices may be included, either directly or by reference, to provide supporting details that could aid in the understanding of the Software Design Document.