(Team Name)

**VrrrRent**

Software Design Document

Name (s):

Sîrbu Ilie-Dănuț

Poenaru Ștefi Sebastian

Oprea Mihai Andrei

Lab Section:-

Workstation:-

Date: (03/22/2021)

**TABLE OF CONTENTS**

[1. INTRODUCTION 2](#_Toc4987)

[1.1 Purpose 2](#_Toc4988)

[1.2 Scope 2](#_Toc4989)

[1.3 Overview 2](#_Toc4990)

[1.4 Reference Material 2](#_Toc4991)

[1.5 Definitions and Acronyms 2](#_Toc4992)

[2. SYSTEM OVERVIEW 2](#_Toc4993)

[3. SYSTEM ARCHITECTURE 2](#_Toc4994)

[3.1 Architectural Design 2](#_Toc4995)

[3.2 Decomposition Description 3](#_Toc4996)

[3.3 Design Rationale 3](#_Toc4997)

[4. DATA DESIGN 3](#_Toc4998)

[4.1 Data Description 3](#_Toc4999)

[4.2 Data Dictionary 3](#_Toc5000)

[5. COMPONENT DESIGN 3](#_Toc5001)

[6. HUMAN INTERFACE DESIGN 4](#_Toc5002)

[6.1 Overview of User Interface 4](#_Toc5003)

[6.2 Screen Images 4](#_Toc5004)

[6.3 Screen Objects and Actions 4](#_Toc5005)

[7. REQUIREMENTS MATRIX 4](#_Toc5006)

[8. APPENDICES 4](#_Toc5007)

# INTRODUCTION

## Purpose

This Software Design Document describes the architecture and system design of the VrrrRent application. It is a translation of requirements into a description of the software structure, software components, interfaces and data necessary for the implementation phase. In essence, the SDD becomes a detailed blueprint for the implementation activity done by the designers.

It is anticipated that the SDD will be used by the application designers. Designers will use the information recorded here as the basis for creating the structure and design of each component.

## Scope

The scope of this product is to help people in need of private transportation for a prolonged period of time for themselves and their private needs when a personal vehicle is not available. In the application, a goal is to provide a user friendly environment where people can easily access what they need in order to hit the road.

This application intends to provide the following goals:

* To create a system that can help people find a suitable vehicle
* To simplify the browsing of vehicles
* To give the ability of manipulating the database which includes adding and removing vehicles
* The application will provide the user details on the various products present in the application
* Providing the ability of creating and accessing their accounts

## Overview

This document completely describes the system at the architecture level, including subsystems and their services, data management, component design which will describe what each component does in a more systematic way and human interface design which will provide the functionality of the system from the user’s perspective.

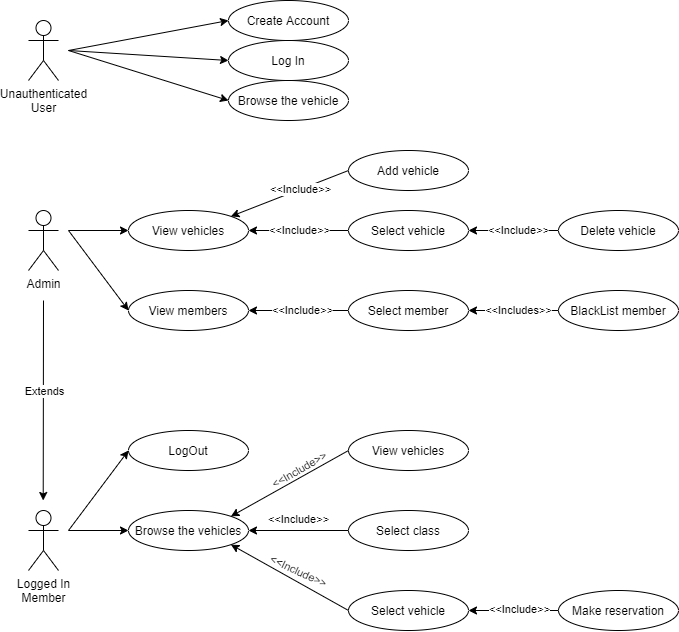
## Reference Material

List any documents, if any, which were used as sources of information for the test plan.

## Definitions and Acronyms

Provide definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the SDD. These definitions should be items used in the SDD that are most likely not known to the audience.

# SYSTEM OVERVIEW



The system diagram provides a brief overview of the car rental web application. In the diagram are presented the main cathegories of users and what each one of them can do together wiith their limitations.

The authenticated user, on top of the common rights shared by the unauthenticated user, can also make actual usage of the products present by making a reservation for a rental.

# SYSTEM ARCHITECTURE

## Architectural Design

--- To be completed with proper diagram ---

## Decomposition Description

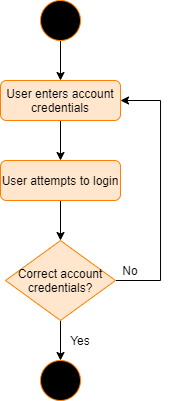
The Model View ViewModel (MVVM) is an architectural pattern used in software engineering.

It is a way of creating client applications that leverages core features of the WPF platform and allows for simple unit testing of application functionality.

* VIEW: A view is defined in XAML and should not have any logic in the code-behind. It binds to the view-model by only using data binding.
* MODEL: A Model is responsible for exposing data in a way that is easily consumable by WPF.
* VIEWMODEL: A ViewModel is a model for a view in the application or as an abstraction of the view. It exposes data relevant to the view and exposes the behavior for the views, usually with Commands.

1. **User login:**

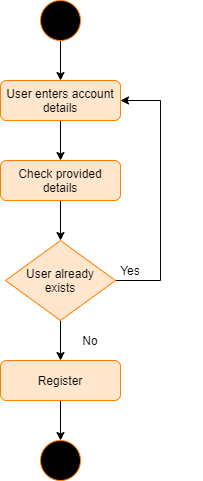
Whatever type of user coming to the site will be required to provide account credentials and press the login button. If the procedure is a success then they will be redirected to the main page of the website. If not, they will be requested to repeat the procedure.



1. **User registration:**

In the case of a user not having a account they will need to create one via registration.

The new user will provide various details which will be checked to see if some of the details are already used by another user. If so then the person will be asked to introduce different details. If not, then the user will be redirected to the home page.

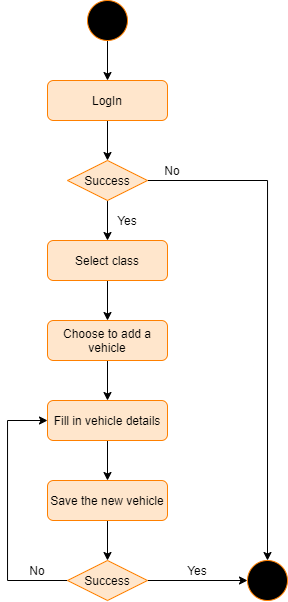


1. **Add vehicle:**

A admin member is capable of adding a new vehicle to the roster.

The admin member is required to log in first. In the case of a successful log in, the admin member will then choose a class of vehicle and choose to add a new vehicle.

Here the admin will be required to complete a form with various details about the new vehicle and save them. If any mandatory detail is missing, the admin will need to complete that part. If anything is in order then they will be redirected to the updated catalog of vehicles.

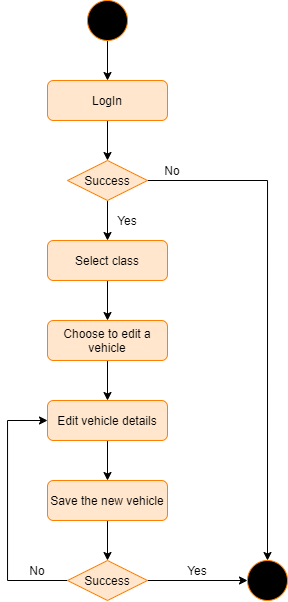
****

1. **Edit vehicle:**

A admin member is capable of editing vehicles from the roster.

The admin member is required to log in first. In the case of a successful log in, the admin member will then choose a class of vehicle and choose to edit a vehicle.

Here the admin will be required to edit a form with various details about the new vehicle and save them. If any mandatory detail is missing, the admin will need to complete that part. If anything is in order then they will be redirected to the updated catalog of vehicles.

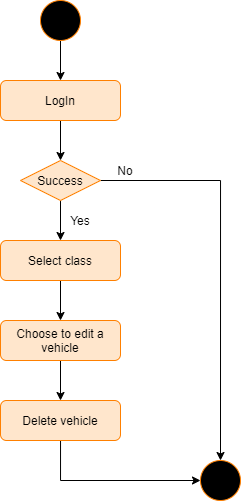


1. **Delete vehicle:**

A admin member is capable of deleting vehicles from the roster.

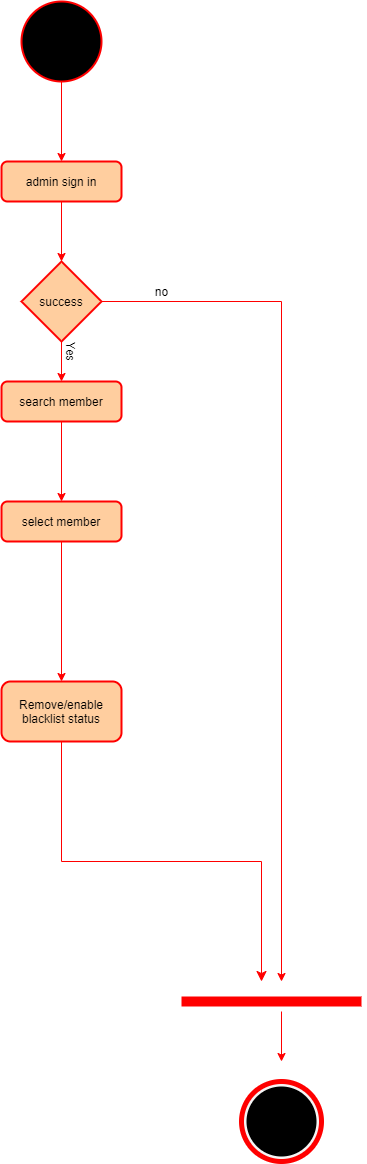
The admin member is required to log in first. In the case of a successful log in, the admin member will then choose a class of vehicle and choose to edit a vehicle.

Here, next to the editable details about the vehicle, will be a “Delete” button which will remove the vehicle from the roster. The admin will be redirected to the updated catalog.



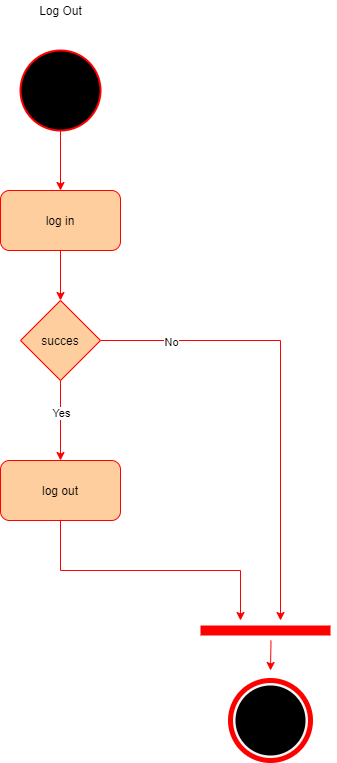
1. **Edit member status:**

Admins will have the capability of editing a member’s status to blacklisted or remove this status based on information received by workers on the field regarding bad habits members may present. The admin will need to be logged in first. Then the admin will be able to look for the specific member and select it. Upon selection, the admin will have a check option “BlackList”. The admin can check or uncheck this option.



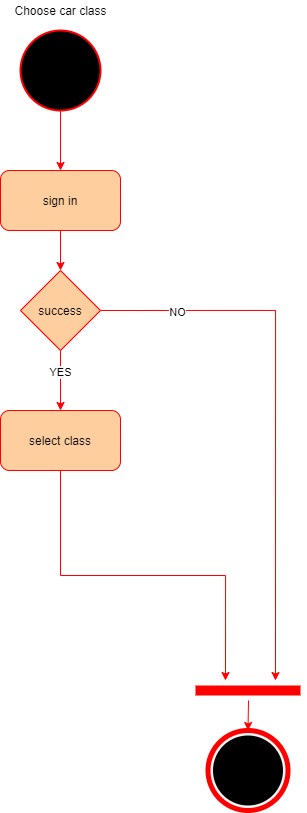
1. **LogOut:**

Both Admin members and usual members, once logged in will be presented with the option of logging out from their accounts via the usage of a “LogOut” button.



1. **Select vehicle class:**

Admin members and usual members of the application have the option, while browsing the vehicle catalog, of choosing a specific vehicle class in order to “filter” their search for a specific vehicle present to offer. To be classified as member they will firstly need to be logged into their accounts.



## Design Rationale

The MVVM architecture was chosen because this architecture is targeted at modern UI Development platforms (Windows Presentation Foundation) in which exist requirements that take into consideration the user experience (UX). The View-Model of MVVM is basically a “powerful value converter” meaning that the View-Model is responsible for exposing the data objects from the Model in such a way that those objects are easily managed and consumed. In this respect, the View-Model is more Model than View, and handles most if not all of the View’s display logic.

More than that, MVVM was designed to make use of specific functions in WPF to better facilitate the separation of View layer development from the rest of the pattern by removing virtually all “code-behind” from the View layer. Instead of requiring Interactive Designers to write View code, they can use the native WPF markup language XAML and create bindings to the ViewModel, which is written and maintained by application developers.

Furthermore, the MVVM architecture guides us how to distribute responsibilities between classes in a GUI application (or between layers), with the goal of having a small number of classes, while keeping the number of responsibilities per class small and well defined.

# DATA DESIGN

## Data Description

## Data Dictionary

# COMPONENT DESIGN

# HUMAN INTERFACE DESIGN

## Overview of User Interface

The web application will include the following features:

1. Create account
2. User Login
3. User Logout
4. Admin edit member functionality
5. Admin add vehicle functionality
6. Admin edit vehicle functionality
7. Admin delete vehicle functionality
8. User/Admin select class functionality

Used frameworks:

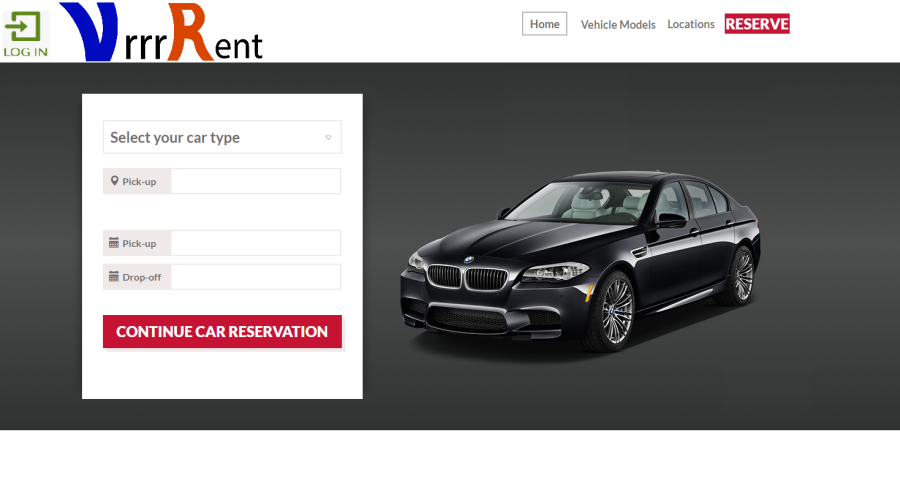
Windows Presentation Foundation (WPF) is a UI framework that creates desktop client applications.

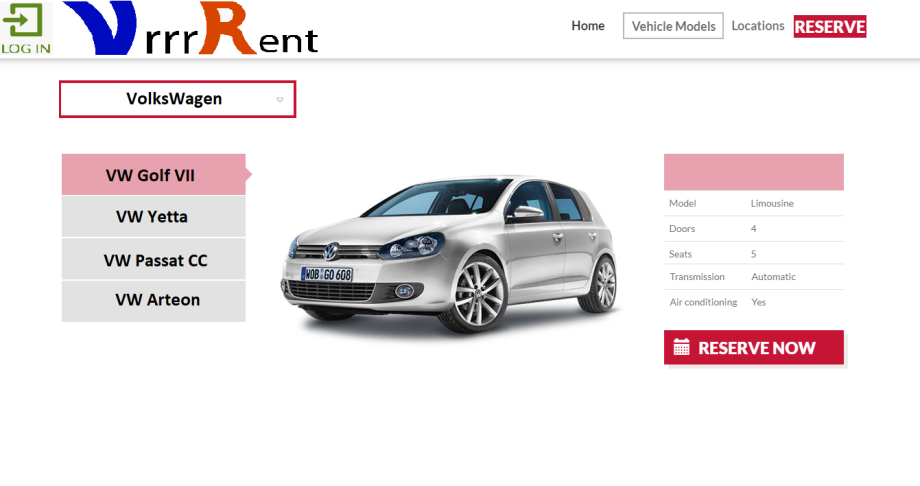
Windows Communication Foundation (WCF) is a framework for building service-oriented applications.

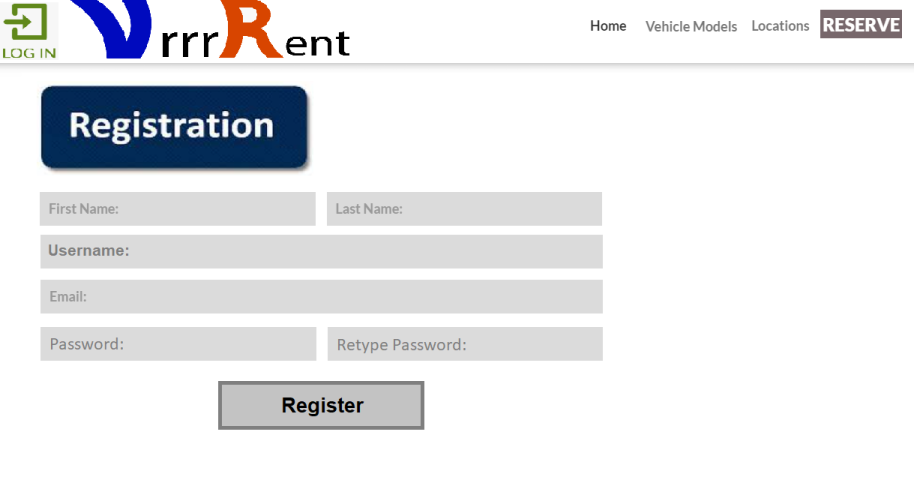
The application guidelines will cover 5 elements:

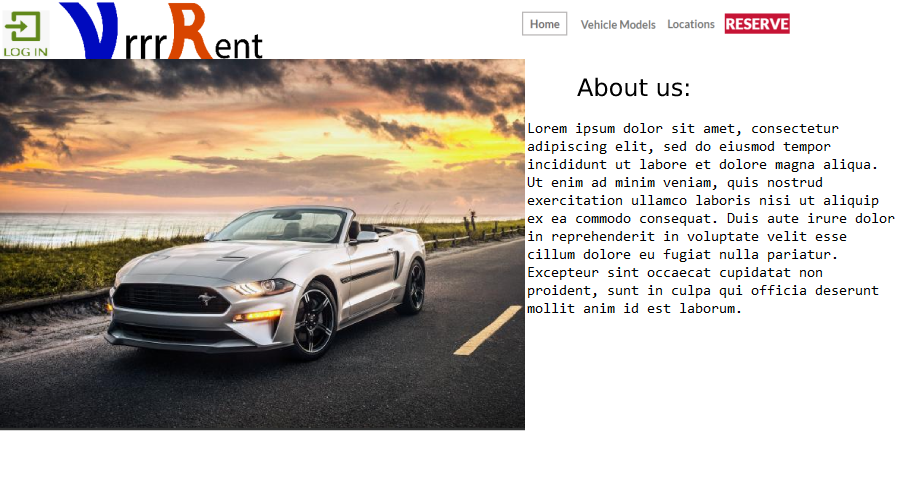
1. Logo
2. Color
3. Typography
4. Illustration
5. Photography

## Screen Images









## Screen Objects and Actions

# REQUIREMENTS MATRIX

# APPENDICES