

991242195| 991496851

AutosExchange by OVVR

PROG37721 | Web Services Using .NET & C# | Final Project

oleh vytvitskyy | Vincent Romani

2019

Contents

[Project Synopsis 2](#_Toc16421323)

[System Design 3](#_Toc16421324)

[Functionality & Workflow 5](#_Toc16421325)

[Home Page 5](#_Toc16421326)

[Account Registration 6](#_Toc16421327)

[Managing Your Account 7](#_Toc16421328)

[Creating an Ad 7](#_Toc16421329)

[Editing an Ad 9](#_Toc16421330)

[Deleting an Ad 11](#_Toc16421331)

[Database Design & Data Used 13](#_Toc16421332)

[Solution Explorer & Files 15](#_Toc16421333)

[Solution Explorer 15](#_Toc16421334)

[HomeController.cs 16](#_Toc16421335)

[About.cshtml 17](#_Toc16421336)

[Contact.cshml 17](#_Toc16421337)

[Index.cshtml: 19](#_Toc16421338)

[Future Direction 23](#_Toc16421339)

# Project Synopsis

AutosExchange is an all-in-one classifieds advertising platform for automobiles. Much like any other classified’s platform, users can identify themselves as a dealership or an independent party and place their automobile for sale. Competitors include AutoTrader and Kijiji Autos, however the competitive advantage is that AutosExchange will not prompt the poster to upgrade the status of ads, spam them with unnecessary notifications, or provide false market analysis information.

The development environment used for the project was Microsoft’s Visual Studio 2019 Community Edition, Version 16.1. ASP.NET was used as the server-side web application framework to produce dynamic web pages. The server-side application will be written mostly in C# using the .NET and Collections frameworks. ASP.NET Web Forms was used to create the graphical user interface for users to interact with.

As for the database portion of the application, Microsoft SQL Express was used. User information as well as listing information will be stored indefinitely, contingent on current legislature and GDPR standards.

# System Design

The functional requirements for the project are as follows:

Account Creation

* The initials step in the web application workflow is to create an account. AutosExchange supports full account registration. A ‘register’ option is available when selecting the login link or create an ad link.

Initial Search and Filtering

* A search and filtering implementation are Present on the home page of the website. Searching and filtering will be possible on parameters that will be determined after a focus group meeting has been conducted determining the most sought-after parameters of cars i.e. Make, Model, Mileage and Drivetrain.

Listing a Car for Sale

* Listing a car for sale is the most fundamental requirement. All users can list a car for sale, enter predefined parameters (some being mandatory [i.e. Price]) of different types, some being a text field, some a combo box of predefined values to choose from. A user will then identify themselves as a dealer or a private party. Should they identify themselves as a dealer, additional information will appear for the user to populate requesting dealership information. A user will be generated a random token that will be used to enter for authentication for editing the ad in order to circumvent having to use unique login accounts.

Editing a Listing

* Using the uniquely generated token that is given once a car is listed, users will be able to use this token to verify they have access to edit the listing. Once the token is entered, a user is presented with editable fields for the listing that corresponds to the uniquely generated token.

Deleting a Listing

* Much like editing, a user can delete a listing. In order to delete a listing a user must enter the unique token provided to them on listing creation. Once a user marks a listing as deleted, it is still stored in then database, however marked deleted [historic] for data collection

Creating a Lister Profile

* After logging in, users can create a lister profile that has their name, address, phone number and location information for information to contact a lister when interested in a listing. This is also integrated into the user account.

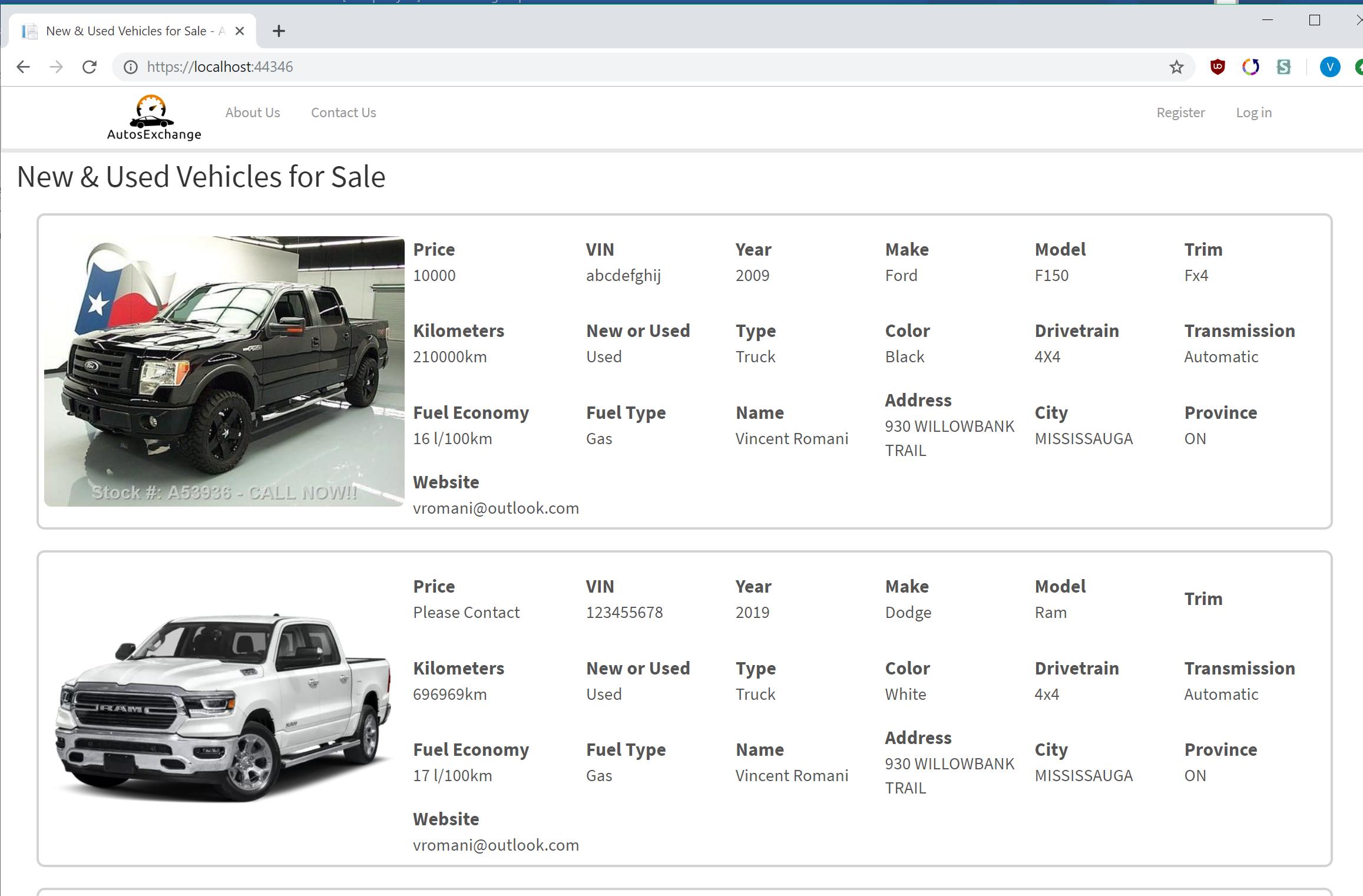
About & Contact Us

* Much like standard businesses, AutosExchange includes a contact us page with integrated Google Maps functionality to locate our office and an About page identifying the purpose of the web application.

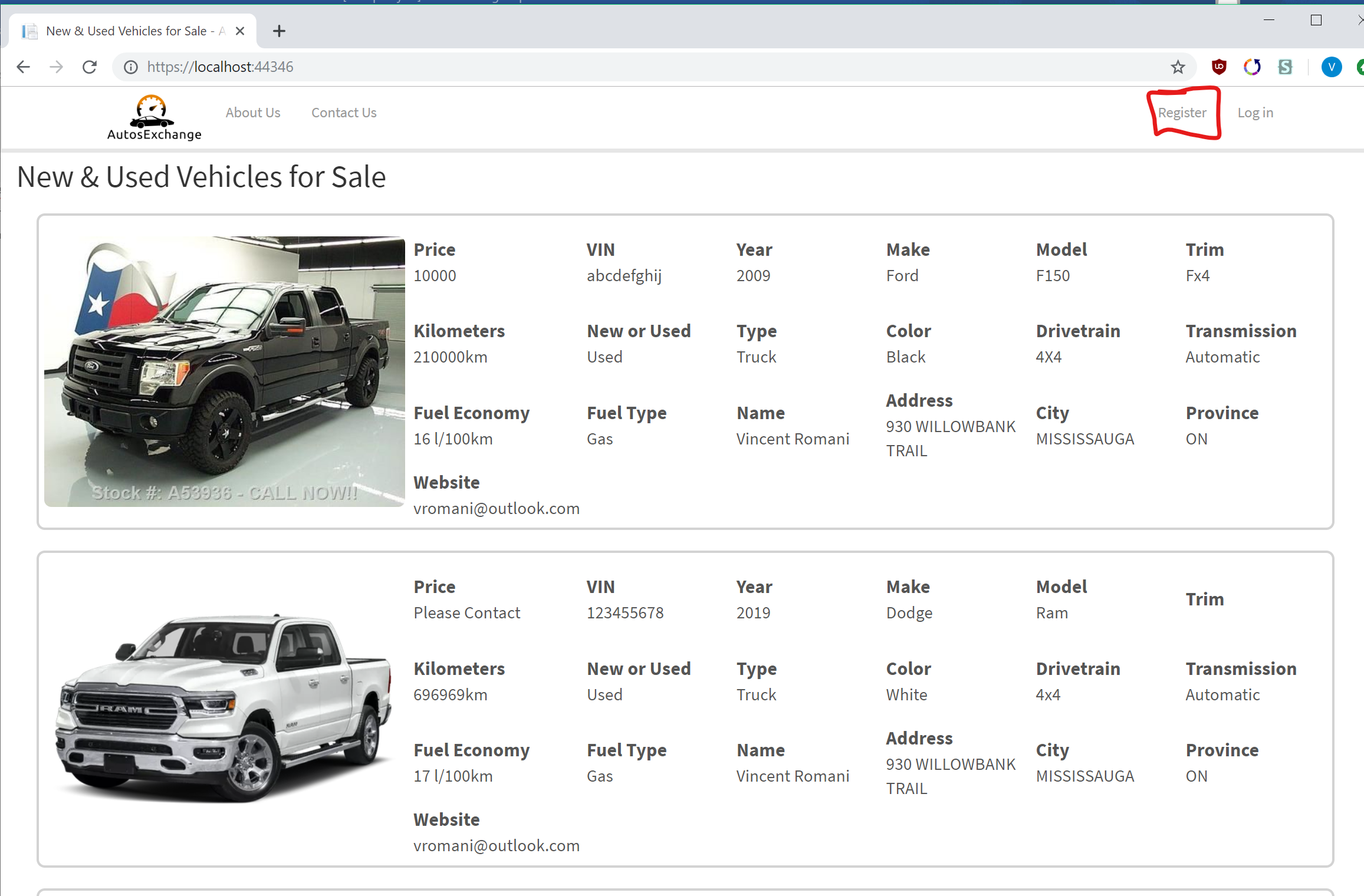
# Functionality & Workflow

### Home Page

The home page of the application will show a header with the company logo, a link to the about and contact us page.



The intended workflow after the homepage is for the user to select the register button at the top right and proceed with their account registration, as so:

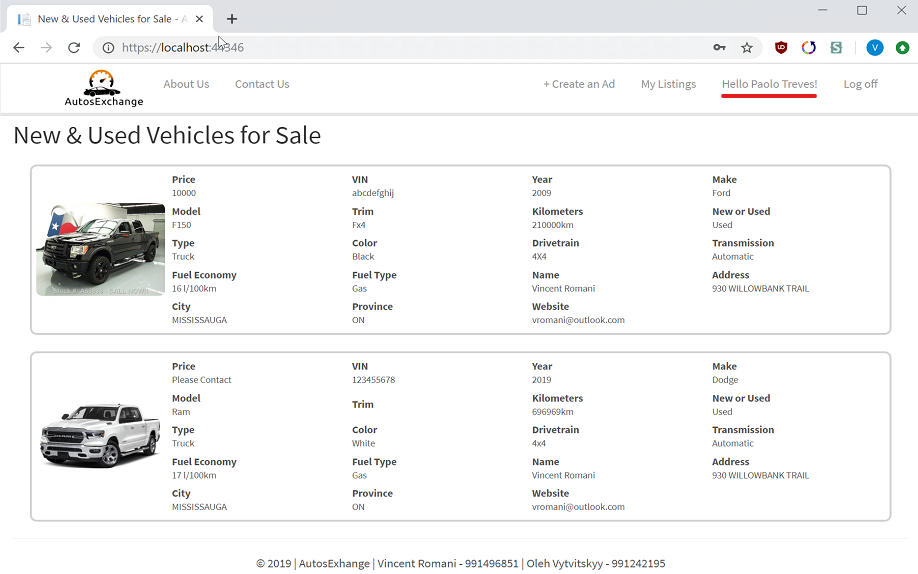


### Account Registration

The following information is requested from the user in order to create an account. For a workflow example we will use the email **paolo.treves@sheridancollege.ca** with the password **IamPaolo1!** and click on **Register** to complete the process.

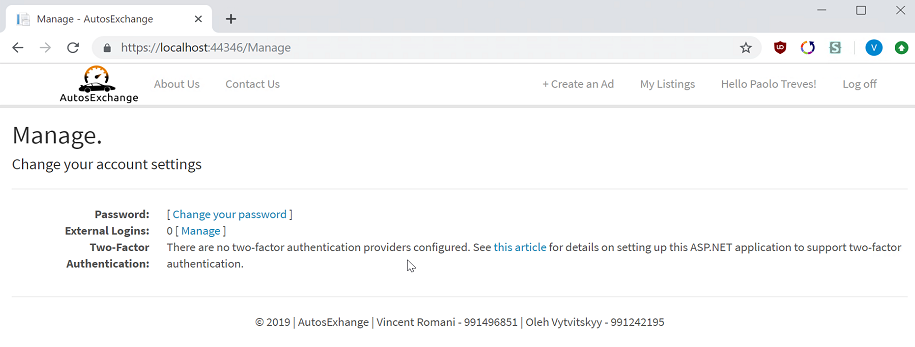


Upon a successful registration, the user will see a greeting with their name in the banner, indicating that they are in fact registered and can proceed with either creating a listing or viewing listings.



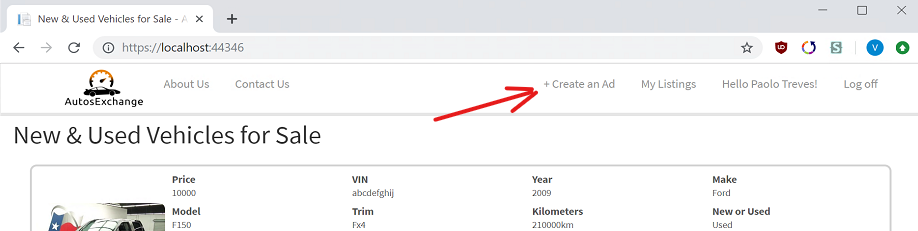
### Managing Your Account

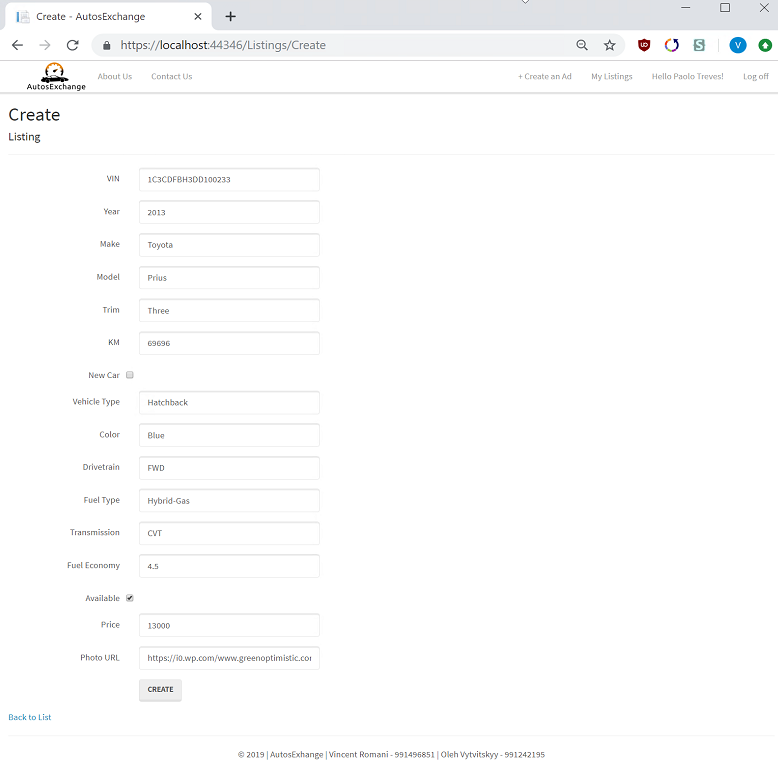
If a user is logged in after registering their account, the user is now able to select their username at the top right in the banner to manage their account and be redirected to the following screen:



### Creating an Ad

After a user registers an account and logs in, they can select the ‘***+Create an Ad’*** link in order to create a posting for their vehicle for sale, like so:

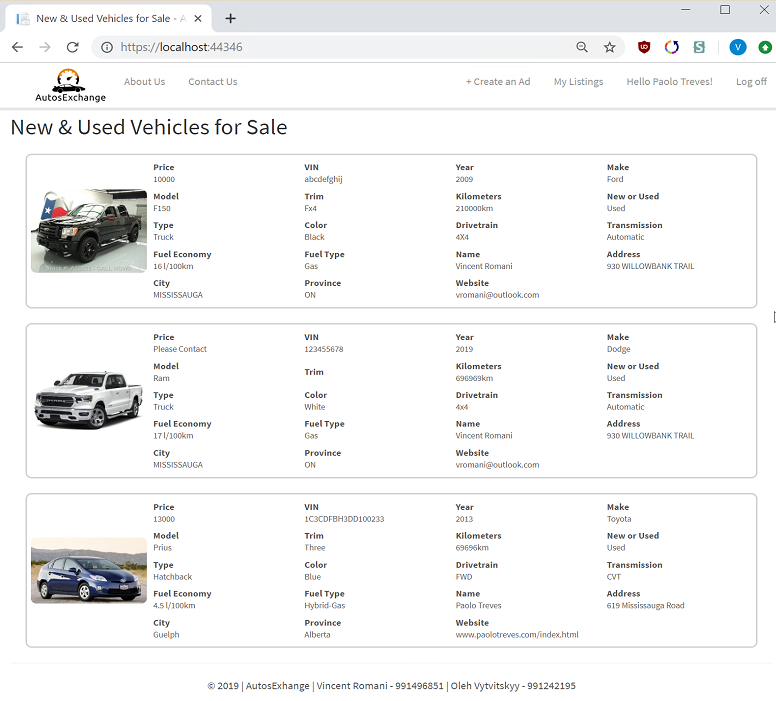


On the following page, the user enters the info for the car they will be posting for sale:

Upon successful creation of the ad, a user will be taken to their current listings:

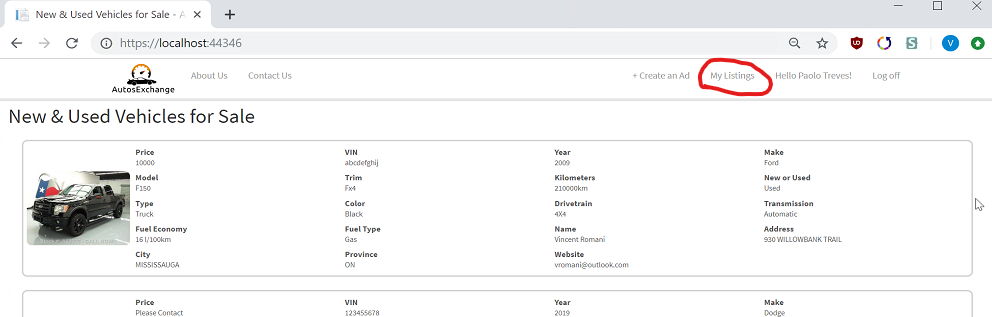


Returning to the home page by clicking the company logo, we can view the listing we added on the home page:

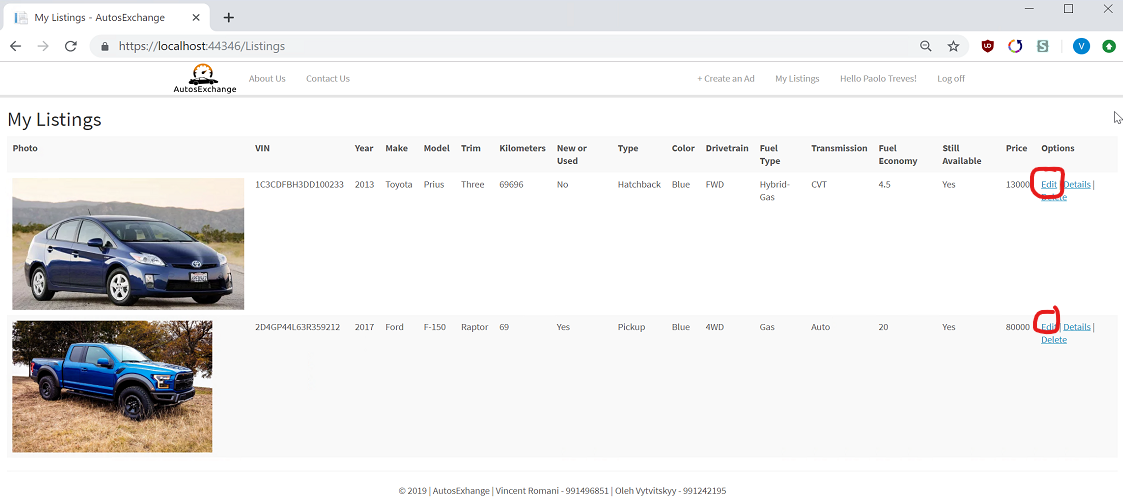


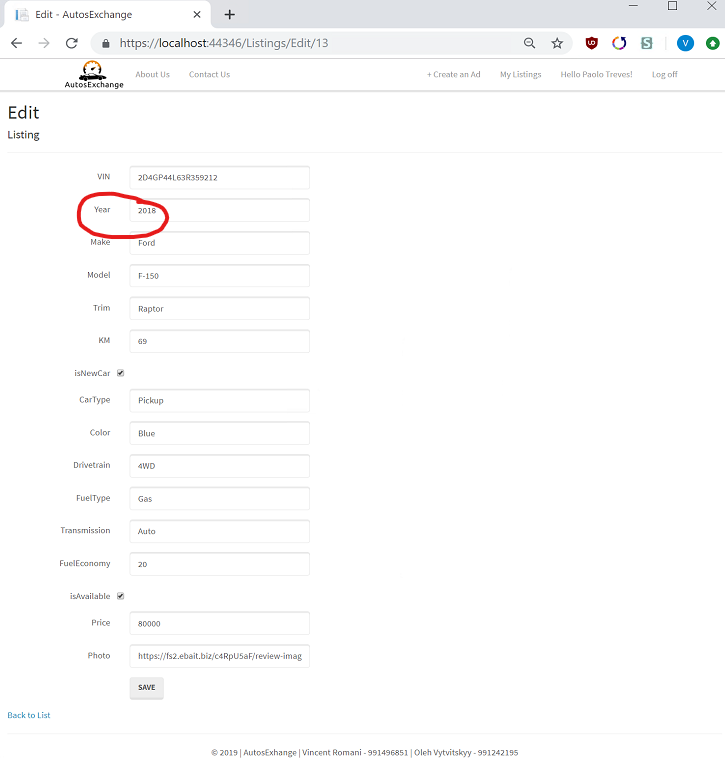
### Editing an Ad

In order to edit an ad, the user will first navigate tot heir listings like so:

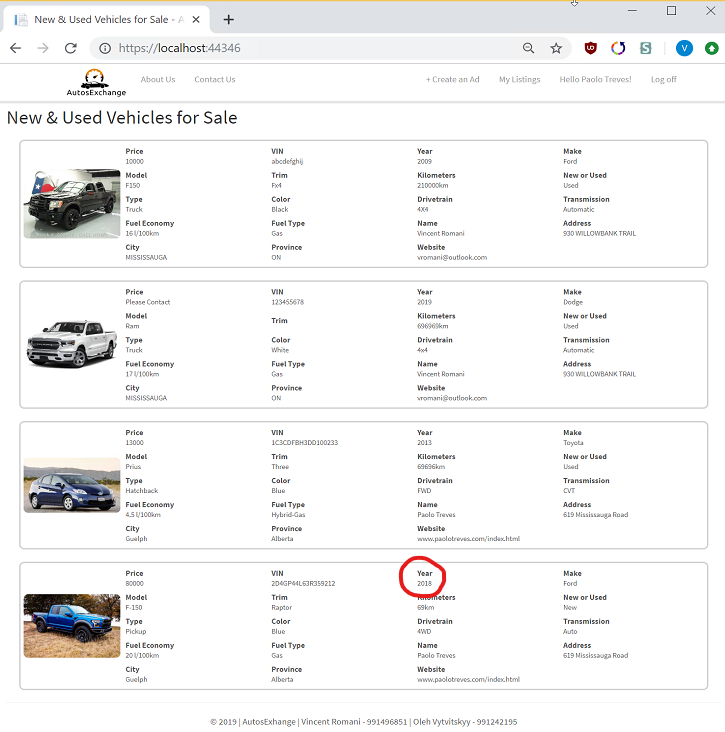


From there, the user has an **‘EDIT’** operation for each of their posted ads, which they must select:



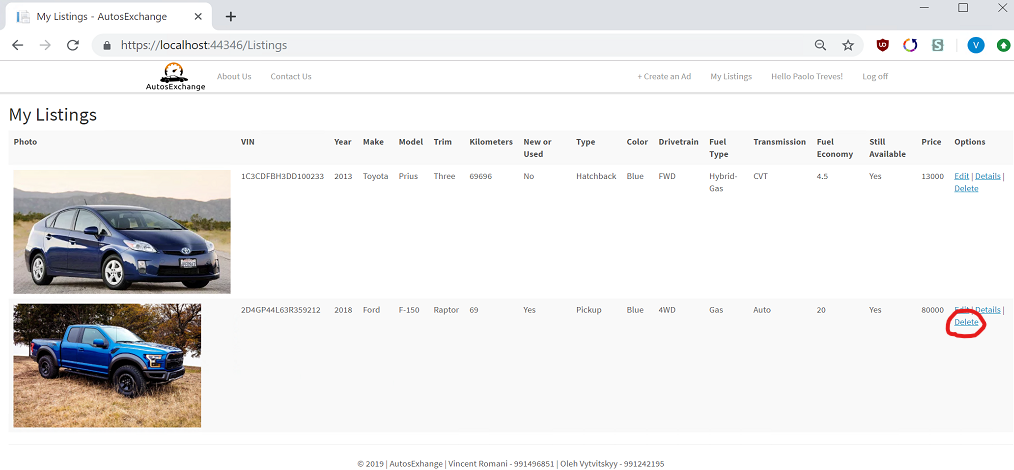


To illustrate functionality, we change the year of the Ford Raptor to 2018, and see the result displayed in the front page as so:

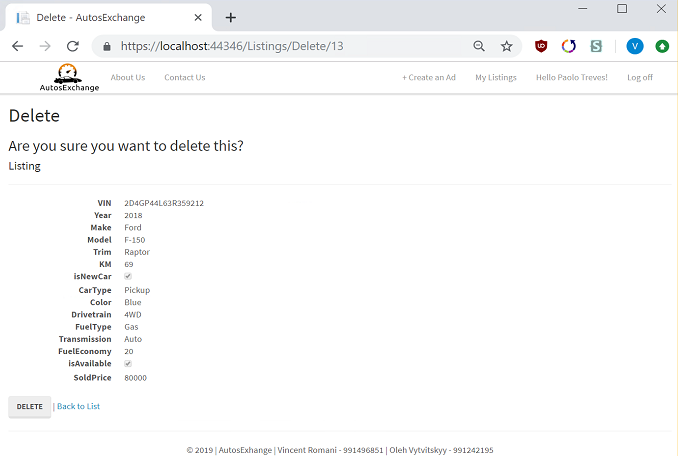


### Deleting an Ad

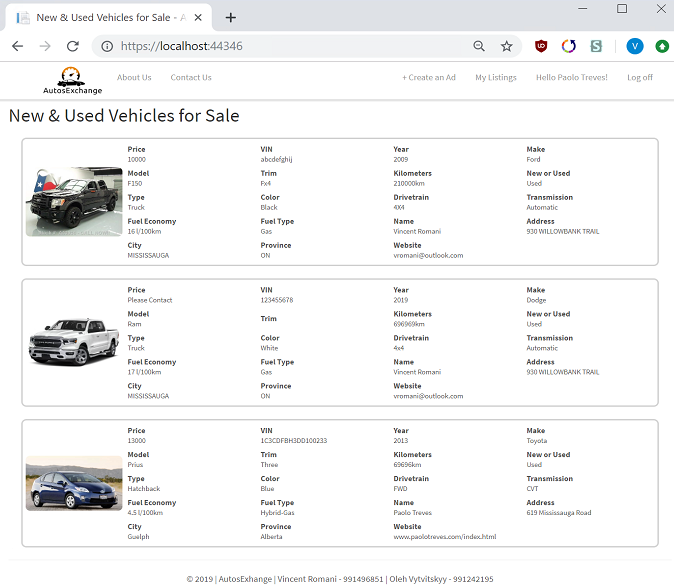
From the ‘My Listings’ page, the user has the option to delete a listing, we will attempt to delete the Ford Raptor as an example:



The user must confirm:

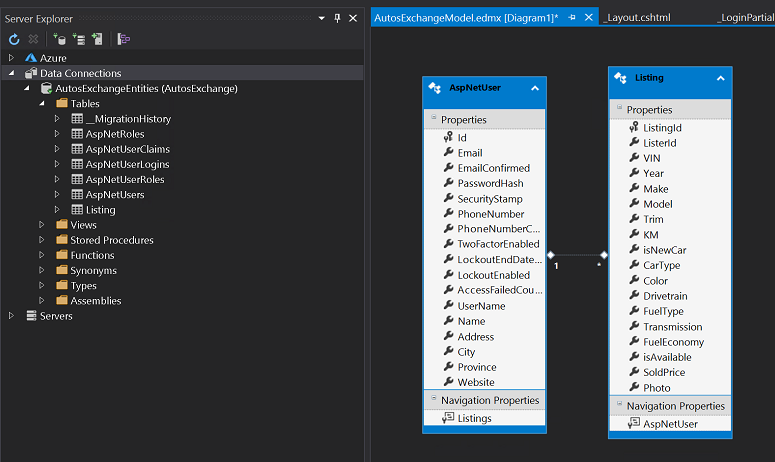


The Ford Raptor no longer appears as a vehicle listed for sale:



# Database Design & Data Used

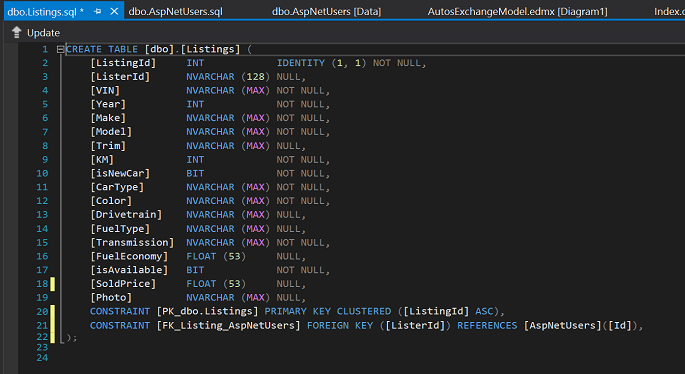
The database design for the project was localized into two tables. AspNetUser and Listing. The following screenshot shows the table structure in Visual Studio.



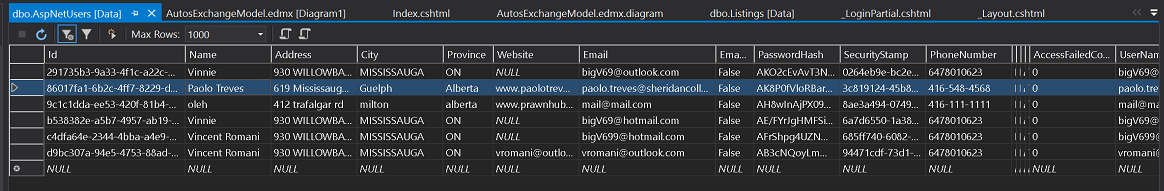
The data used for the listing table as shown from the workflow demo is stored in the Listings table:



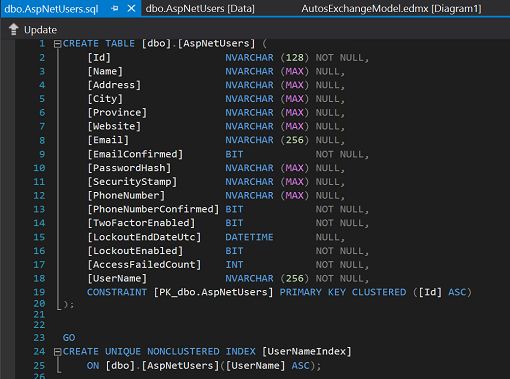
The table create statement for the table is shown here:



The other table used was AspNetUsers so we can store users and link them to accounts, the data used for the users that is currently present in the database is also from the workflow example:

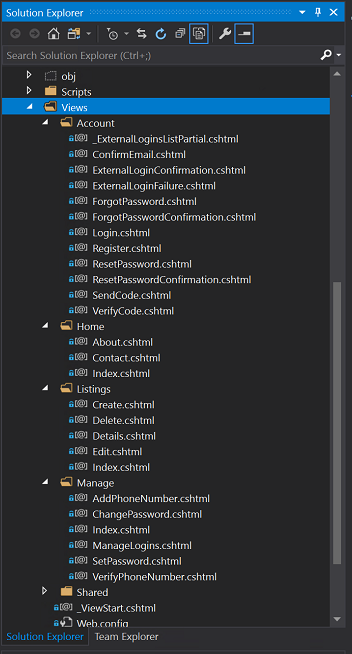
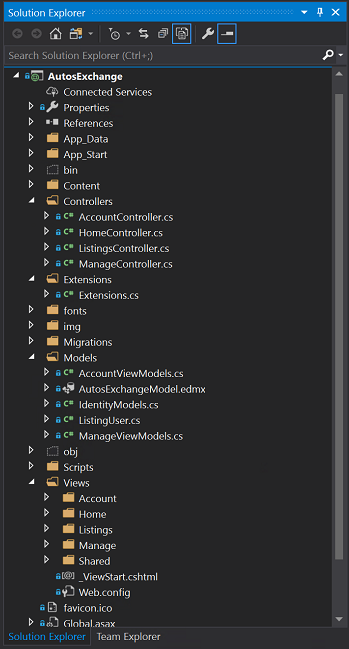


The table create statement for the table is shown here:



# Solution Explorer & Files

### Solution Explorer



### HomeController.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using AutosExchange.Models;

using AutosExchange.Extensions;

using Microsoft.AspNet.Identity;

namespace AutosExchange.Controllers

{

    public class HomeController : Controller

    {

        private ApplicationDbContext db = new ApplicationDbContext();

        public ActionResult Index()

        {

            IEnumerable<ListingUser> listings =

                from l in db.Listings

                join u in db.Users

                on l.ListerId equals u.Id

                select new ListingUser

                {

                    Name = u.Name,

                    Address = u.Address,

                    City = u.City,

                    Province = u.Province,

                    Website = u.Website,

                    ListingId = l.ListingId,

                    VIN = l.VIN,

                    Year = l.Year,

                    Make = l.Make,

                    Model = l.Model,

                    Trim = l.Trim,

                    KM = l.KM,

                    isNewCar = l.isNewCar,

                    CarType = l.CarType,

                    Color = l.Color,

                    Drivetrain = l.Drivetrain,

                    FuelType = l.FuelType,

                    Transmission = l.Transmission,

                    FuelEconomy = l.FuelEconomy,

                    isAvailable = l.isAvailable,

                    SoldPrice = l.SoldPrice,

                    Photo = l.Photo

                };

            return View(listings.ToList());

        }

        public ActionResult About()

        {

            ViewBag.Message = "Group Project for PROG37721";

            ViewBag.AdditionalMessage = "AutosExchange was built by Vincent Romani and Oleh Vytvitskyy using ASP.Net, Entity Frameworks, and SQLExpress";

            return View();

        }

        public ActionResult Contact()

        {

            ViewBag.Message = "Contact Us";

            return View();

        }

    }

}

### About.cshtml

@{

    ViewBag.Title = "About";

}

<h2>@ViewBag.Title</h2>

<h3>@ViewBag.Message</h3>

<p>@ViewBag.AdditionalMessage</p>

### Contact.cshml

@{

    ViewBag.Title = "Contact";

}

<h2>@ViewBag.Title.</h2>

<h3>@ViewBag.Message</h3>

<address>

    1430 Trafalgar Rd<br />

    Oakville, ON L6H 2L1<br />

    <abbr title="Phone">P:</abbr>

    (905) 845-9430

</address>

<div class="mapouter"><div class="gmap\_canvas">

    <iframe width="500" height="500" id="gmap\_canvas" src="https://maps.google.com/maps?q=sheridan%20college%20oakville&t=&z=15&ie=UTF8&iwloc=&output=embed" frameborder="0" scrolling="no" marginheight="0" marginwidth="0"></iframe>

</div>

<style>

    .mapouter {

        position: relative;

        text-align: right;

        height: 500px;

        width: 500px;

    }

    .gmap\_canvas {

        overflow: hidden;

        background: none !important;

        height: 500px;

        width: 500px;

    }

</style>

    </div>

<address>

    <strong>Support:</strong>   <a href="mailto:romanivi@sheridancollege.ca">romanivi@sheridancollege.ca</a><br />

    <strong>Marketing:</strong> <a href="mailto:vytvitso@sheridancollege.ca">vytvitso@sheridancollege.ca</a>

</address>

### Index.cshtml:

@model IEnumerable<AutosExchange.Models.ListingUser>

@{

    ViewBag.Title = "New & Used Vehicles for Sale";

}

<h2>@ViewBag.Title</h2>

<style>

    img{

    align-self: center;

    widows: 15vw;

    max-width: 150px;

    height:auto;

    max-height: 40vh;

    margin: 5px 5px;

    border-radius: 7px;

    transition: ease all .7s;

    }

    .bold{

    font-weight: bold;

    margin: 0px;

    font-size: 1.1em;

    }

    .fact{

    align-self: center;

    margin: 3px;

    font-size: 0.8em;

    transition: ease all .5s;

    }

    .mat-tab-body{

    border: grey 1px solid;

    border-radius: 8px;

    }

    .tile{

    border: lightgrey 2px solid;

    border-radius: 8px;

    display:flex;

    flex-direction: row;

    margin:20px;

    }

        .factContainer{

        display: grid;

        grid-template-columns: 1fr 1fr 1fr 1fr;

        grid-template-rows: 1fr 1fr 1fr;

        justify-content: center;

        flex-direction: column;

        margin: 5px 0px;

        width:100%;

    }

</style>

@foreach (var item in Model)

{

    if (item.isAvailable)

    {

<div class="tile">

    <img src="@item.Photo" alt="" style="max-height:200px; width:auto;" />

    <div class="factContainer">

        <div class="fact">

            <p class="bold">

                Price

            </p>

            @(item.SoldPrice.HasValue ? item.SoldPrice.ToString() : "Please Contact")

            </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.VIN)

            </p>

            @Html.DisplayFor(modelItem => item.VIN)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Year)

            </p>

            @Html.DisplayFor(modelItem => item.Year)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Make)

            </p>

            @Html.DisplayFor(modelItem => item.Make)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Model)

            </p>

            @Html.DisplayFor(modelItem => item.Model)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Trim)

            </p>

            @Html.DisplayFor(modelItem => item.Trim)

        </div>

        <div class="fact">

            <p class="bold">

                Kilometers

            </p>

            @(item.KM.ToString() + "km")

        </div>

        <div class="fact">

            <p class="bold">

                New or Used

            </p>

            @(item.isNewCar ? "New" : "Used")

        </div>

        <div class="fact">

            <p class="bold">

                Type

            </p>

            @Html.DisplayFor(modelItem => item.CarType)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Color)

            </p>

            @Html.DisplayFor(modelItem => item.Color)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Drivetrain)

            </p>

            @Html.DisplayFor(modelItem => item.Drivetrain)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Transmission)

            </p>

            @Html.DisplayFor(modelItem => item.Transmission)

        </div>

        <div class="fact">

            <p class="bold">

                Fuel Economy

            </p>

            @(item.FuelEconomy.ToString() + " l/100km")

        </div>

        <div class="fact">

            <p class="bold">

                Fuel Type

            </p>

            @Html.DisplayFor(modelItem => item.FuelType)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Name)

            </p>

            @Html.DisplayFor(modelItem => item.Name)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Address)

            </p>

            @Html.DisplayFor(modelItem => item.Address)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.City)

            </p>

            @Html.DisplayFor(modelItem => item.City)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Province)

            </p>

            @Html.DisplayFor(modelItem => item.Province)

        </div>

        <div class="fact">

            <p class="bold">

                @Html.DisplayNameFor(model => model.Website)

            </p>

            @Html.DisplayFor(modelItem => item.Website)

        </div>

        </div>

    </div>

    }

}

# Future Direction

Provided more time and adequate resources, we have some additional functionality that we would add to the website. However, given the limited timeframe, we could not implement the following:

1. **Clickable Listings**
   * Clicking an ad would bring up an additional page with more details on the vehicle. Right now all information is displayed on the Main Page
2. **Search functionality**
   * Ability to search based on area code, city, or vehicle
3. **Enums**
   * Limit input ability by implementing enums for things like transmission, vehicle type, etc.
4. **Vehicle Features**
   * Additional table with features for a specific vehicle that is for sale, could be displayed on the clickable ad that lists detailed information
5. **Login with Facebook/Social Media**
   * Create an account and allow user to login with their Facebook instead of email
6. **Multiple Photos**
   * Being able to add multiple photos to an ad, and the ability to upload a photo rather than providing a URL for the photo
7. **Next Button for Home Page**
   * Have it display 10 vehicles, then the next button brings you to the next page of listings
8. **Google Ads for revenue**
   * Need to make money some-how
9. **Additional Security**
   * Block off all URL domains and only allow access to URLs that a user is allowed to access